

RTGS 프로토타입 시스템 PoC 테스트 1차 결과보고

2024. 8. 23.

I. 개요	1
II. 프로토타입 시스템 구축 내용	2
1. 기본 아키텍처 설계	2
2. 기본 업무절차 설계	3
3. 시스템 구축환경 구성	4
4. 프로그램 개발	8
5. 성능 테스트	17

IT전략국 IT운영부 RTGS시스템팀

- IT전략국은 금융결제국과 함께 실시간충액결제 방식의 신속자금이체를 24×365 연중무휴로 제공하는 **소액RTGS시스템 구축을 추진중***

* 「RTGS 방식 신속자금이체시스템 구축 종합계획」(결제정책팀-1169, 2023.12.28, 부총재보)
「소액RTGS시스템 구축을 위한 IT부문 기본계획」(결제시스템팀-3, 2024.1.2, 부총재보)

- 일평균 1.1억건 이상의 대량거래를 5초 이내 처리하는 신속성과 특정 IT센터의 전면 가동중단에도 서비스를 지속할 수 있는 수준의 복원력 확보가 주요 목표

- 소액RTGS시스템에 필요한 고성능과 고가용성은 기존의 전통적인 방식의 시스템 아키텍처에는 한계가 있으며, 아직 국내 코어뱅킹 시스템은 IT센터 동시 운영체계(Active-Active) 구현 사례가 부재*

* 「소액RTGS 구축관련 국내사례조사 결과보고」(결제시스템팀-1366, 2023.12.29, 부장)

- RTGS 방식 신속자금이체시스템을 운영중인 유럽, 미국, 호주 등의 중앙은행은 인메모리 기술을 활용하여 순차적으로 거래를 처리하고, 병행처리 방식의 Active-Active를 구현함으로써 처리속도와 복원력을 극대화*

* 「소액RTGS시스템 구축 관련 IT아키텍처 국외사례 조사결과」(RTGS시스템팀-48, 2024.4.29, 부총재보)
「이탈리아 중앙은행과의 정례 회의 결과 보고」(RTGS시스템팀-81, 2024.7.11, 국장)
「유럽 지급결제시스템 기술혁신 현황 및 시사점」(결제정책팀-631, 2024.7.18, 국장)

- RTGS시스템팀은 소액RTGS시스템 구성에 필요한 IT요소기술과 비즈니스 요구사항을 분석하고 국내외 사례를 검토하여 **당행의 소액RTGS시스템에 적합한 아키텍처를 수립하고 기술검증 방안을 마련하여 구축계획을 수립***

* 「소액RTGS시스템 구축을 위한 기본 아키텍처」(RTGS시스템팀-62, 2024.5.20, 부장)
「소액RTGS시스템 구축 관련 기술검증 방안」(RTGS시스템팀-69, 2024.5.31, 부장)
「소액RTGS시스템 기술검증을 위한 프로토타입시스템 구축계획」(RTGS시스템팀-78, 2024.7.1, 부장)

⇒ 본격적인 시스템 구축(2026~2030년)에 앞서 소액RTGS시스템의 아키텍처를 검증하기 위한 **프로토타입 시스템의 핵심 프로세스 개발환경을 구성하고 기본 프로그램 개발을 완료**

II

프로토타입 시스템 구축 내용

1

기본 아키텍처 설계

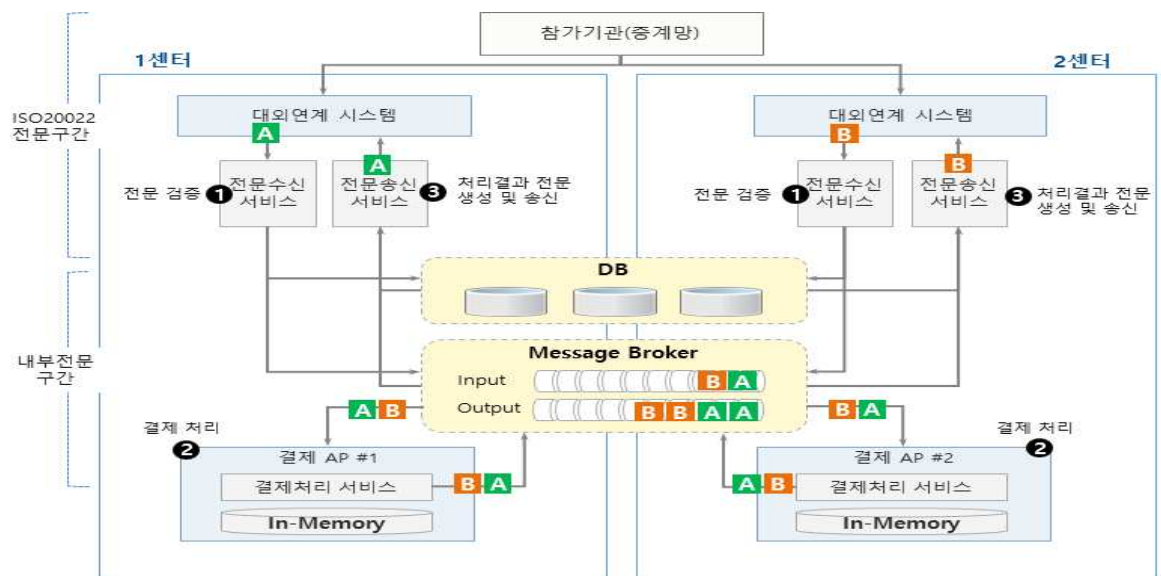
- 소액RTGS시스템의 기능과 성능을 검증하기 위하여 전문 송수신 및 저장 등을 담당하는 솔루션 부문과 전문 송수신과 결제처리를 수행하는 프로그램 개발 부문으로 구분하여 기본 아키텍처를 설계

검증대상 구성요소

구 분	구 성 요 소	기 능
솔루션	대외연계	참가기관과의 전문(ISO20022) 송수신(이체 요청접수, 결과송부)
	데이터베이스	참가기관과 송수신 전문원장 저장(IT센터간 중복체크 및 동기화)
	메시지 브로커	IT센터간 전문 직렬화 처리(단계별 작업내역 큐에 저장)
	인메모리 저장소	계좌잔액, 기관정보 등을 저장, 신속결제처리
프로그램	전문수신	전문을 파싱, 검증, 변환하고 메시지 브로커로 전달(병렬처리)
	전문송신	메시지 브로커의 처리 결과에 따라 송신 전문 생성(병렬처리)
	결제처리	인메모리 저장소를 통해 결제요청 처리(순차처리)

- 구성요소는 다중의 IT센터에 동등하게 배치하여 동시 운영하며, 데이터 베이스와 메시지 브로커는 정합성을 위해 단일 클러스터로 구성하고 장애에 대비하여 데이터 사본을 분산저장

검증대상 아키텍처 개념도



2 기본 업무절차 설계

- 검증을 위한 기본 아키텍처에서는 하나의 거래를 여러 세부 처리단계로 분리하여 독립적으로 처리함으로써 병목현상을 축소
 - 1개 센터내 최대 성능을 확인하고, 2개 IT센터간 계좌잔액, 거래내역 등 데이터 정합성과 성능, 복원력 등을 시험

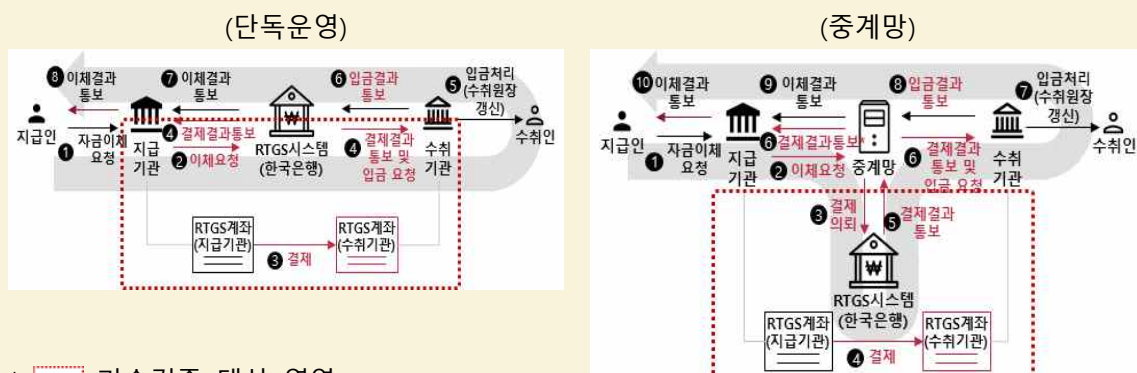
프로토타입 시스템 기술검증 목표

구분	시나리오
부하 모델	송신기관1개, 수신기관1개를 설정하여 균등하게 부하 생성(API 개발) 적정부하부터 최대부하(4,000~20,000 TPS)까지 Ramp-up한 후 10분간 유지
성능 점검	적정부하 및 최대부하 환경에서 처리량, 응답시간, 자원사용률 확인 서버 노드 및 센터 간 데이터 정합성 검증(순서, 잔액, 건수 대사)
복원력 확인	적정부하 환경에서 노드 중지(프로세스 강제종료, 네트워크 절체) 중지 노드 재기동(중지 노드수를 늘려가며 반복 테스트)

- 지급결제 거래과정 중 핵심 영역인 ①전문수신(결제의뢰 수신), ②결제처리, ③전문송신(결제결과 통보) 등 3단계를 대상으로 구현
 - 소액RTGS시스템 구축 모델에 따른 중계기관의 유무와 상관없이 결제의뢰 전문을 처리하는 단계는 동일하므로 어떤 모델로 구축하더라도 기술검증 결과를 활용 가능

<참고1>

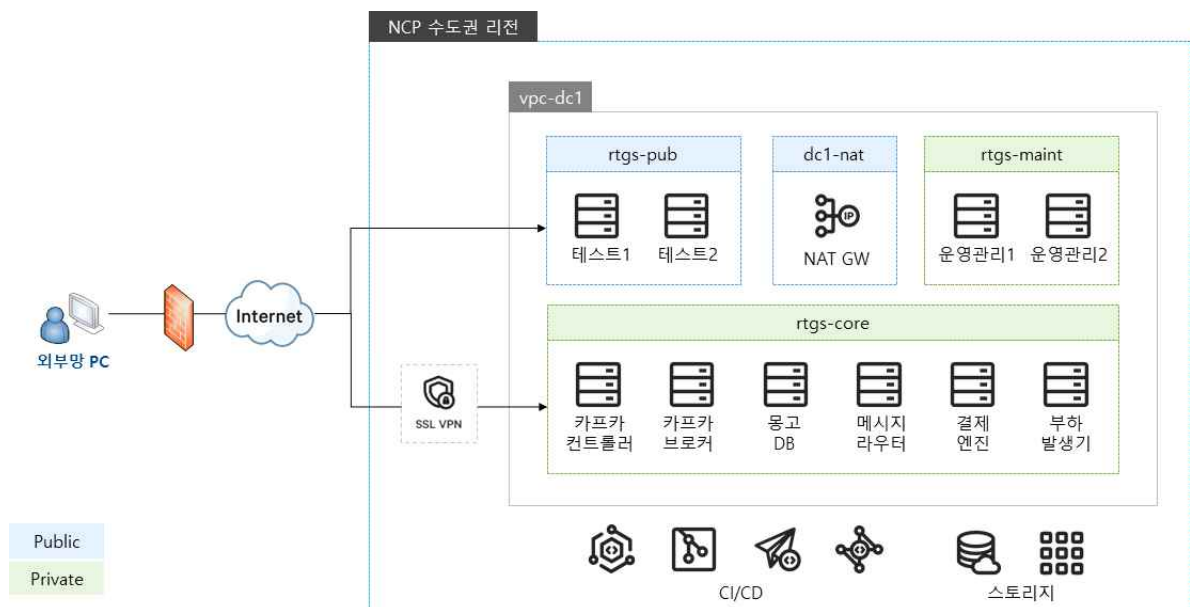
구축 모델별 전문 흐름



3 시스템 구축환경 구성

- IT기획팀의 클라우드 연구플랫폼(네이버 공공클라우드)을 활용하여 프로토타입 시스템 개발 및 테스트를 수행할 수 있는 구축환경을 구성
 - 현재는 프로그램 개발 단계이므로 각 구성요소를 단일 서버로 구성하였으며, 성능 및 복원력 테스트시 서버를 추가하여고가용성 구성할 계획
 - 보안성 확보를 위하여 업무처리 서버는 인터넷과 직접 연결되지 않도록 프라이빗 서브넷에 구성하고 SSL VPN을 통해 접속

프로토타입 시스템 구축환경



- 서버는 가상서버(KVM 기반)로 생성하여 평소에는 최소 사양으로 운영하고 성능 테스트 등 필요시 솔루션에서 요구하는 권장 사양으로 변경*
 - * 서버 사양(CPU, 메모리)을 정해진 유형(최소 2core/4GB, 최대 64core/256GB) 내에서 수시로 변경이 가능
- 운영체제는 실제 운영환경과 유사하게 구성하기 위하여 Kafka, MongoDB 등 주요 솔루션용 서버에 레드햇 계열의 centos를 설치하고 AP 및 관리 도구용 서버에 편의성이 높은 ubuntu를 설치

프로토타입 서버 구성 내역

서브넷	호스트명	IP 주소	OS	vCPU (core)	MEM (GB)	SSD (GB)	HDD (GB)	용도
rtgs-pub	pub-tst1	192.168.20.6 211.188.35.146	centos-7.8	2	4	50		테스트1
	pub-www1	192.168.20.7 175.45.221.158	ubuntu-22.04	2	4	50		테스트2
rtgs-core	core-kctrl1	192.168.10.11	centos-7.8	2	4	50		카프카컨트롤러
	core-kbrk1	192.168.10.21	centos-7.8	2	4	50	100	카프카브로커
	core-mg1	192.168.10.51	centos-7.8	2	4	50 100		몽고DB
	core-in1	192.168.10.81	ubuntu-22.04	2	4	50		전문수신
	core-settl1	192.168.10.101	ubuntu-22.04	2	4	50		결제처리
	core-perf1	192.168.10.201	ubuntu-22.04	2	4	50		성능 테스트
rtgs-ops	ops-maint1	192.168.100.11	ubuntu-22.04	2	4	50		운영관리1
	ops-maint2	192.168.100.12	ubuntu-22.04	2	4	50		운영관리2

* 성능 정보 화면(프로메테우스) : <http://192.168.100.11:9090>
 시각화 화면(그라파나) : <http://192.168.100.11:3000>
 Kafka 관리도구(UI for 카프카) : <http://192.168.100.12:8080>
 헤이즐캐스트 관리도구 : <http://192.168.100.12:8081>

<참고2>

Kafka 상용버전(Confluent)의 권장 사양

Confluent Component	운영 환경 권장 사양				최소 구성 사양		
	CPU(Core)	Memory	Disk	수량(고가용성 제공)	CPU(Core)	Memory	Disk
ZooKeeper	6	16 GB	512 GB (SSD 권장)	3 node	3	4 GB	100 GB (SSD)
Kafka Broker	24	64 GB	12 TB (24 X 1 TB HDD disk. RAID 10 권장)	4 node (온라인 업그레이드/패치, 장애를 고려하여 4 node를 권장)	4	16 GB	1 TB (HDD, RAID 10 권장)
Confluent Control Center (Web UI Admin/User Portal)	12	32 GB (JVM default 6 GB)	300 GB (SSD 권장)	1 node (관리 컴포넌트로 고가용성 제외)	4	16 GB	250 GB (HDD, RAID 10 권장)
Schema Registry	4	16 GB	100 GB (HDD)	2 node	2	4 GB	50 GB (HDD)
REST Proxy	16	32 GB(최소 2GB + Producer당 64MB + Consumer당 16MB 권장)	100 GB (HDD)	2 node	16	32 GB(최소 2GB + Producer당 64MB + Consumer당 16MB 권장)	50 GB (HDD)
ksqlDB	8	64GB	300 GB (SSD 권장)	2 node	4	20 GB	100 GB (SSD)
Kafka Connect (Connect Worker)	12	64 GB (커넥터에 따라 0.5 - 4 GB heap size 조정 필요)	100 GB (HDD)	2 node	4	16 GB	50 GB (SSD)
사이징 기준 예상 처리량(Throughput)	운영 환경 권장 사양을 적용시 예상 최대 처리량: * Broker 4 node 기준: - 예상 최대 처리량 0.5GBps - 예상 최대 가용 스토리지: 12.8TB(3별 복제 기준)						
비고	* 최적화된 사이징을 위해서는 Confluent Professional Service 를 이용하여 요구사항, 환경 분석 및 사이징/아키텍처 디자인 등의 컨설팅 서비스를 적극 권장합니다. * Disk의 용량은 Usable size를 의미 합니다. ** Disk type SSD로 표시된 경우 520MB/s의 Throughput을 권장합니다. ** Kafka Broker의 Single volume Throughput은 125MB/s로 RAID 10 구성으로 600MB/s의 Throughput을 권장합니다. ** 목표 처리량 및 인프라 환경에 따라 구성에 보다 낮은 사양의 DISK로 구성 및 운영도 가능합니다. *** NIC는 10GE 기준 *** VM 구성 시 각 node는 Host별 1대씩 배치를 권장합니다. 특히 Broker Node의 경우 성능 저하가 발생될 수 있기 때문에 동일 호스트 배치되지 않는 구성이 반드시 필요합니다. **** REST Proxy의 메모리는 Producer당 64MB, Consumer당 16MB 추가로 산정하여 사이징을 적용하는 것을 권장합니다.						

□ 핵심 프로세스 구현을 위한 오픈소스 솔루션으로 Kafka, MongoDB, Hazelcast를 설치

○ 솔루션은 무료로 사용 가능한 커뮤니티 라이선스를 이용하고 최신 안정화 버전*을 채택

* 오픈소스 특성상 패치 및 버전 업그레이드가 빠른 속도로 이루어지고 있어 주기적으로 릴리스 상황을 확인할 필요

오픈소스 솔루션 채택 목록

제조사	제품명	라이선스	버전
몽고DB	MongoDB Community Edition	SSPL	7.0.12
아파치 재단	Apache Kafka	Apache 2.0	3.7.0
헤이즐캐스트	Hazelcast Community Edition	Apache 2.0 & Hazelcast Community License	5.4.0

○ 성능 향상을 위하여 솔루션 공식 매뉴얼에서 권장하는 구성을 최대한 적용

— Kafka 브로커의 데이터 저장용 파일시스템은 상대적으로 높은 IOPS를 제공하는 HDD 스토리지로 별도 구성하고 마운트시 noatime 설정

— MongoDB 서버는 ulimit 증가, THP 및 스왑 비활성화, 디스크 스케줄러 변경 등을 적용

□ 서버 및 JVM 등 성능 정보를 실시간으로 확인할 수 있도록 오픈소스를 활용하여 가시성 확보

프로토타입 시스템 가시성 구성도



오픈소스	수행 기능
Node exporter	시스템 성능 정보
Spring Boot (Micrometer)	JVM 성능 정보
Promtail	로그파일 정보
Prometheus	성능 메트릭 수집
Grafana loki	로그파일 수집
Grafana	시각화 화면 제공

시각화 대시보드 화면



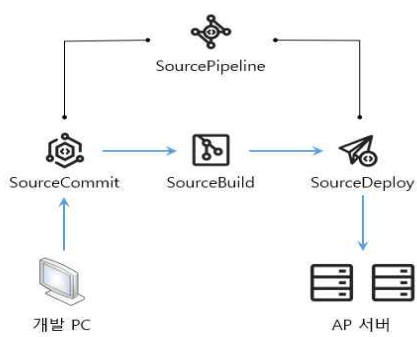
4

프로그램 개발

개발 환경구성

- ☐ 지급지시 전문 및 결제를 처리하는 프로그램은 회계결제시스템에서 사용 중인 Java 언어를 사용하여 구현
- ☐ 개발 편의성을 위해 오픈소스 프레임워크인 스프링부트*를 이용
- * 스프링부트는 카프카 버전 호환성 고려하여 최신 안정화 버전인 3.3.1을 사용
- ☐ 클라우드 연구플랫폼에서 제공하는 서비스를 이용하여 CI/CD 파이프라인을 구성
- ☐ 레파지토리에 프로그램 소스코드가 커밋되면 빌드 및 배포, 프로그램 재시작 절차를 자동으로 수행

프로토타입 시스템 프로그램 개발환경



구분	세부 내용
프레임워크	스프링부트 3.3.1
자바	Open JDK 17
빌드	Gradle 8.8 / SourceBuild
개발도구	VS Code
형상관리	Git / SourceCommit
레파지토리	input-processor, settlement-engine
Branch	master, develop

샘플 전문 작성

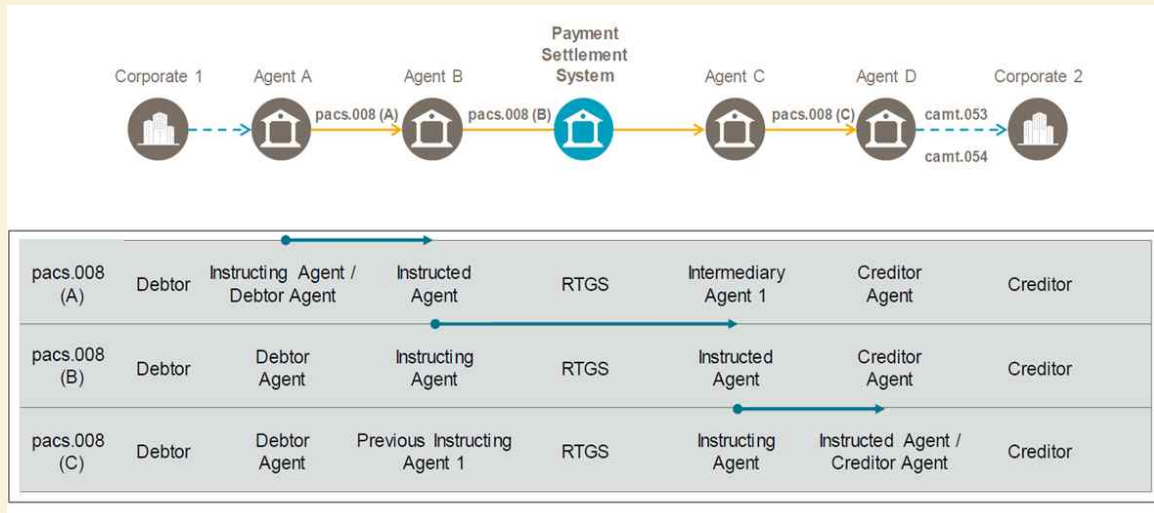
- BAH, Business Message로 구성된 pacs.008, pacs002 샘플 및 XSD 작성
 - 결제의뢰 전문은 pacs.008, 결제결과 통보 전문은 pacs.002를 사용하고 항목 값은 한은금융망의 전문 설계를 참조
 - 건당 전문 크기는 pacs.008이 4.34KB, pacs.002가 2.79KB
 - 프로토타입 시스템에서는 이 중에 송신기관, 수신기관, 전문번호, 전문유형, 생성시간 항목 등을 적용
 - 전문번호는 영업일(8자리), 기관코드(4자리), 일련번호(8자리)를 조합하여 총 22자리로 구성

ISO 20022 전문 주요항목

전 문	메시지 항목	XML 태그
업무헤더 (BAH)	송신기관 코드 수신기관 코드 거래고유번호 전문유형 생성시간	<Fr><FIld><FinInstnId><ClrSysMmbld><Mmbld> <To><FIld><FinInstnId><ClrSysMmbld><Mmbld> <MizMsgldr> <MsgDefldr> <CreDt>
결제의뢰 (pacs.008)	송신기관 코드 수신기관 코드 의뢰인 성명 의뢰인 계좌번호 이체기관 코드 수취기관 코드 수취인 성명 수취인 계좌번호 결제유형 수수료제외 이체금액 이체일자 생성시간	<InstgAgt><FinInstnId><ClrSysMmbld><Mmbld> <InstdAgt><FinInstnId><ClrSysMmbld><Mmbld> <Dbtr><Nm> <DbtrAcct><Id><Othr><Id> <DbtrAgt><FinInstnId><ClrSysMmbld><Mmbld> <CdtrAgt><FinInstnId><ClrSysMmbld><Mmbld> <Cdtr><Nm> <CdtrAcct><Id><Othr><Id> <ChrgBr> <IntrBkSttlmAmt> <IntrBkSttlmDt> <CreDtTm>
결과통보 (pacs.002)	송신기관 코드 수신기관 코드 결제상태 결제시스템 참조번호 생성시간	<InstgAgt><FinInstnId><ClrSysMmbld><Mmbld> <InstdAgt><FinInstnId><ClrSysMmbld><Mmbld> <TxSts> <ClrSysRef> <CreDtTm>

<참고4>

RTGS 관련 ISO 20022 전문 구성



구 분	구 성	내 용
Business Application Header (head.001)		The Business Application Header (BAH) is a mandatory component of the CBPR+ Business Message. The Business Application Header enables both application routing rules and logic without having to read the Business Document.
FI to FI Customer Credit Transfer (pacs.008)		The pacs.008 has two core sets of nested elements: Group Header which contains a set of characteristics that relates to all individual transaction Credit Transfer Transaction Information which contains elements providing information specific to the individual credit transfer transaction.
FI to FI Payment Status Report (pacs.002)		The Financial Institution To Financial Institution Payment Status Report message is sent by an instructed agent to the previous party in the payment chain. It is used to inform this party about the positive or negative status of an instruction. It is also used to report on a pending instruction

전문 수신 서비스

- HTTP를 통해 ISO 20022 전문(XML)을 수신하여 유효성을 검사하고 항목값을 파싱하여 객체를 생성
 - 자바에서 XML을 다루기 위한 API는 크게 3가지로 구분되며 기능 및 성능을 고려하여 유효성 검사에는 SAX Parser를 사용하고 실제 항목값 파싱에는 StAX Parser를 사용하여 구현
 - XML API는 스레드 안전(thread-safe)하지 않으므로 싱글톤 객체로 구현하지 않도록 유의
 - 유효성 검사는 XSD* 스키마를 이용하여 수행
- * XML Schema Definition : XML 문서의 구조 및 해당 문서가 포함할 수 있는 적법한 요소와 속성을 명시
- XML 처리 성능은 전문 수신 서비스의 전체 성능에 직접적으로 영향을 미치므로 최적의 처리 방안을 모색할 필요

<참고3>

XML Parser 비교

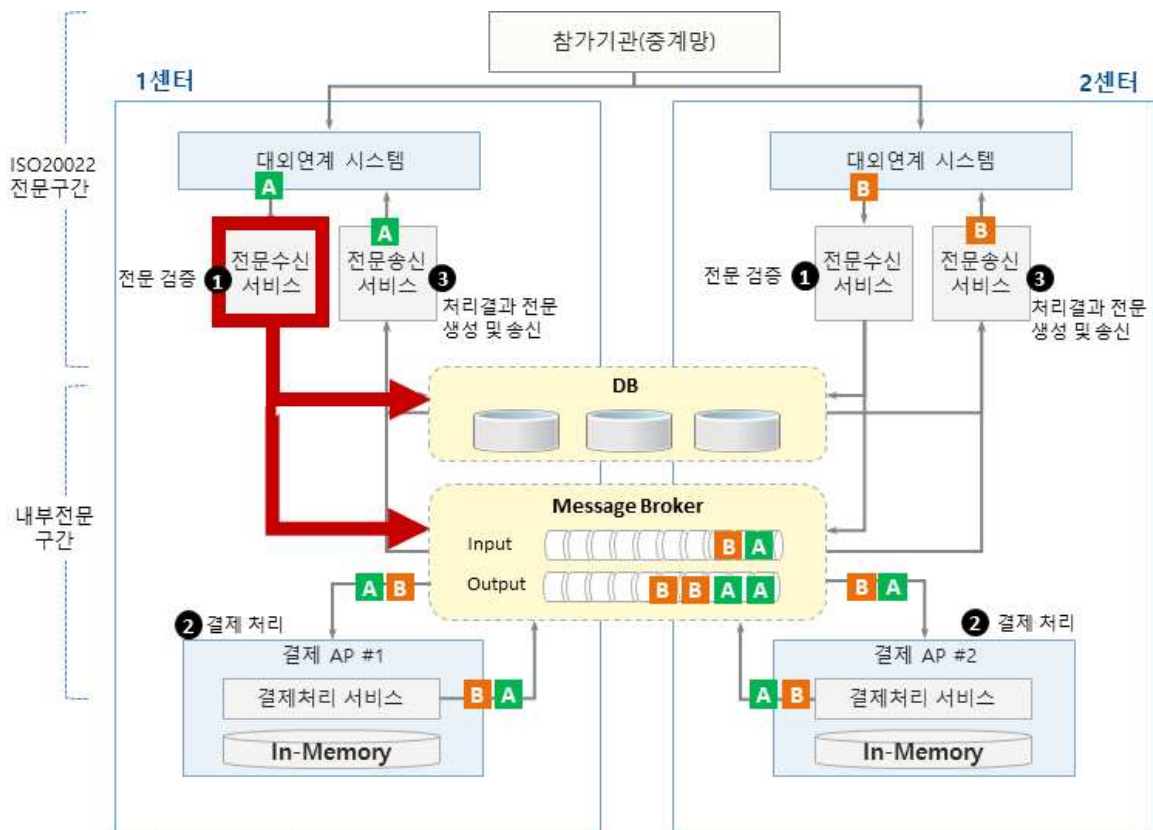
특징	DOM	SAX	StAX
API Type	In memory tree	Push, streaming	Pull, streaming
Ease of Use	High	Medium	High
XPath Capability	Yes	No	No
CPU and MEM Efficiency	Varies	Good	Good
Forward Only	No	Yes	Yes
Read XML	Yes	Yes	Yes
Write XML	Yes	No	Yes
CRUD	Yes	No	No

- 전문 파싱까지 정상적으로 완료되면 전문의 원본을 MongoDB에 저장하고 결제에 필요한 필수항목*을 Kafka의 Input 토픽으로 전송

* 결제처리 등 시스템 내부에서는 크기가 큰 참가기관 전문을 그대로 사용하지 않고 필요한 항목만 추출하여 최대한 작은 크기의 전문으로 변환함으로써 처리성능을 향상

프로토타입 시스템에서 참가기관 전문(pacs.008)의 크기는 4.34KB인 반면 Kafka에 저장되는 크기는 75Bytes

전문수신 서비스 영역



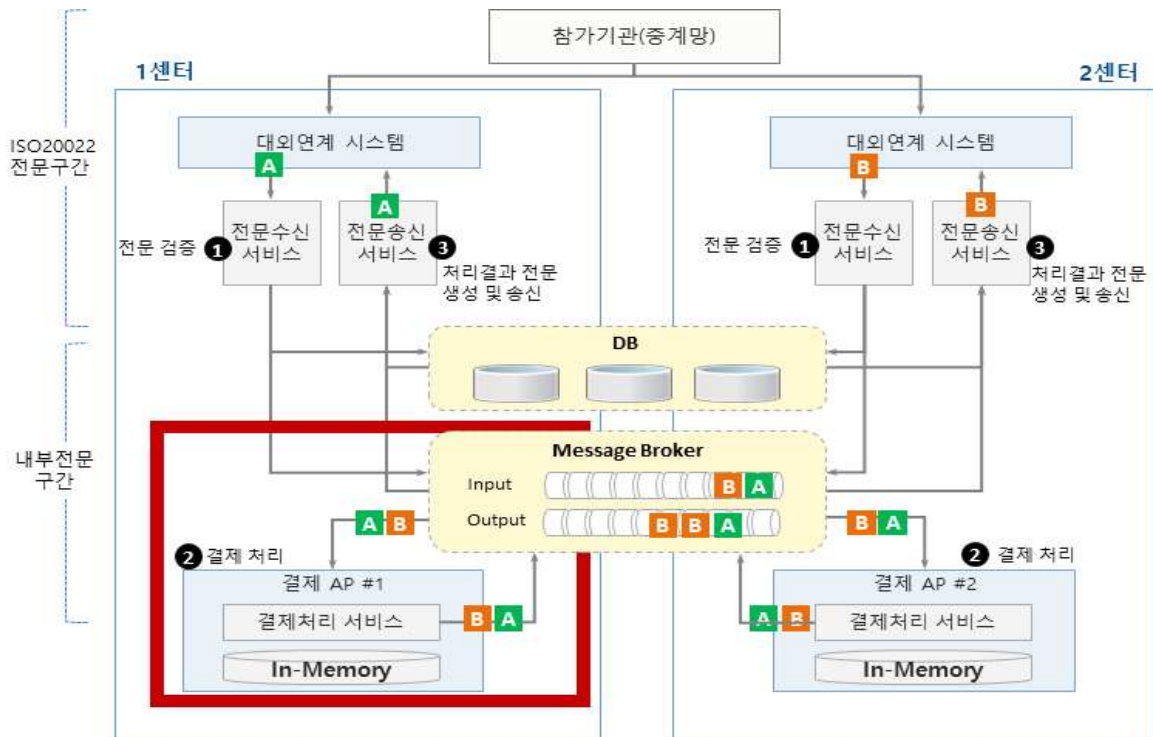
전문수신 프로그램

구분	프로그램	내용
common	AgentCode.java	참가기관 코드
	Command.java	메시지 구분
dto	AppHdr.java	ISO 20022 전문 헤더 객체
	CdtTrfTxInf.java	ISO 20022 전문 본문 객체
	GrpHdr.java	ISO 20022 전문 본문 객체
	Envelope.java	ISO 20022 전문 객체
	InputDocument.java	몽고DB 전송용 객체
	InputMessage.java	카프카 메시지 전송용 객체
src	InputProcessorApplication.java	전문수신 main 소스
	KafkaProducerConfig.java	카프카 프로듀서 사용을 위한 설정 - 부트스트랩 서버, 시리얼라이저 등
	KafkaProducerService.java	카프카 프로듀서를 이용하여 토픽에 메시지 전송
	RtgsController.java	전문 수신 처리 컨트롤러 - /pay/customer : pacs.008 전문 처리 - /pay/gen : 카프카 input 토픽 입력 - /test/pay/customer-parse : XML 파싱 테스트
	XMLCreator.java	(테스트용) XML 파일 생성
	XMLReader.java	(테스트용) XML 파일 읽기
	XMLService_dom.java	(테스트용) XML 파일 파싱 비교용 소스1 DOM : XML 문서 전체를 메모리에 로드하여 트리 구조로 변환하고 랜덤 액세스
	XMLService_mix.java	(테스트용) XML 파일 파싱 비교용 소스2 StAX : XML 문서 일부를 메모리에 로드하여 이벤트 기반으로 스트리밍 처리
	XMLService.java	XML 처리 서비스 - XSD 파일을 읽어서 validator 생성후 문법검사 - XML의 요소들을 파싱하여 객체 생성
resources	pacs.008.xml	ISO20022(008) 전문
	pacs.002.xml	ISO20022(002) 전문
	head.001.001.03.xsd	BAH 전문 스키마
	pacs.008.001.11.xsd	pacs.008 전문 스키마
	pacs.002.001.13.xsd	pacs.002 전문 스키마
	rtgs.pacs.008.xsd	(테스트용)
	rtgs.xsd	ISO 20022 전문 스키마
	application.properties	스프링부트 애플리케이션 설정
log	inpu-processor.log	애플리케이션 로그
	jetty-access.log	WAS 로그

결제 처리 서비스

- 결제 처리 서비스는 Kafka의 Input 토픽에 보관된 결제의뢰를 차례대로 가져와 결제를 수행하고 처리결과를 다시 Kafka의 Output 토픽으로 전송
 - 결제처리 성능 향상을 위하여 원장 데이터는 Hazelcast를 이용해 메모리에서 관리하고 트랜잭션을 적용하여 정합성을 확보
- 서비스 재기동, 장애 등에 대비하여 인메모리 데이터를 주기적으로 디스크에 저장
 - 결제 처리 서비스는 시작시 디스크에 저장된 스냅샷을 읽어와 원장 데이터를 복원하고 해당 데이터의 기준 시점(Kafka offset)으로 이동하여 결제 처리를 재개
 - Hazelcast 유료 버전에서는 데이터 저장 및 복원 기능(Map persistence)을 제공하나 무료 버전에서는 사용이 불가하여 자체 개발

결제처리 서비스 영역



결제처리 프로그램

구분	프로그램	내용
common	AgentCode.java	참가기관 코드
	Command.java	메시지 구분
dto	InputMessage.java	카프카 메시지(Input 토픽) 수신용 객체
	OutputMessage.java	카프카 메시지(Output 토픽) 전송용 객체
src	SettlementEngineApplication.java	결제처리 main 소스
	HazelcastConfig.java	헤이즐캐스트 사용을 위한 설정 - 트랜잭션 매니저
	HazelcastService.java	헤이즐캐스트 관련 서비스 - 디스크에서 스냅샷 로드 - 카프카 컨슈머 오프셋 설정 - 스냅샷이 없는 경우 초기 잔액 설정 - 결제 처리
	KafkaProducerConfig.java	카프카 프로듀서 사용을 위한 설정
	KafkaProducerService.java	카프카 프로듀서를 이용하여 토픽에 메시지 전송
	KafkaConsumerConfig.java	카프카 컨슈머 사용을 위한 설정
	KafkaConsumerService.java	카프카 컨슈머 Input 토픽에서 메시지를 읽어서 HazelcastService를 이용하여 결제를 처리하고 KafkaProducerService를 이용하여 결과를 Output 토픽으로 전송(싱글 쓰레드 : concurrency=1)
resources	application.properties	스프링부트 애플리케이션 설정
log	inpu-processor.log	애플리케이션 로그
	jetty-access.log	WAS 로그

모니터링

- 전문 수신 및 결제처리 서비스의 처리결과는 Kafka와 Hazelcast 관리도구를 통해 저장된 데이터와 처리시간 등을 확인 가능

Kafka 관리화면

UI for Apache Kafka 83b5a60 v0.7.2

Dashboard

RTGS

Brokers

Topics

Consumers

Topics / input

Overview Messages Consumers Settings Statistics

Seek Type: Offset Offset Partitions: All items are selected. Key Serde: String Value Serde: String Clear all Submit Oldest First

Q Search + Add Filters

DONE 31 ms 47 KB 500 messages consumed

Offset	Partition	Timestamp	Key	Value
3679962	0	2024. 8. 12. 15시 7분 3초		{\"command\": \"SETTLE\", \"senderCode\": \"1051\", \"receiverCode\": \"1020\", \"amount\": 100}
3679963	0	2024. 8. 12. 15시 7분 3초		{\"command\": \"SETTLE\", \"senderCode\": \"1051\", \"receiverCode\": \"1020\", \"amount\": 100}
3679964	0	2024. 8. 12. 15시 7분 3초		{\"command\": \"SETTLE\", \"senderCode\": \"1051\", \"receiverCode\": \"1020\", \"amount\": 100}
3679965	0	2024. 8. 12. 15시 7분 3초		{\"command\": \"SETTLE\", \"senderCode\": \"1051\", \"receiverCode\": \"1020\", \"amount\": 100}

Timestamp: 2024. 8. 12. 15시 7분 3초
Timestamp type: CREATE_TIME

Key Serde: Size: 0 Bytes

Value Serde: String
Size: 75 Bytes

Hazelcast 관리화면

HAZELCAST

dev 5.4

Menu

Console

SQL

Members: 1

Administration: 0

WAN Replication: 0

Scripting: 0

Healthcheck: 0

Namespaces: 0

Storage: 0

Maps: 2/2

Replicated Maps: 0

Caches: 0

MultiMaps: 0

Lists: 0

Sets: 0

PN Counters: 0

Flake ID Generators: 0

Stream Processing: 0

Compute: 0

24/08/12 09:21 PM 5.4.1

RESET ZOOM

2024-08-12, 15:06 → 2024-08-12, 15:09

2024-08-12, 15:06 → 2024-08-12, 15:09

Map Statistics (In-Memory Format: BINARY)

RESET TIME 1 minute ago → now

Default View

Member	Entries	Gets	Puts	Removals	Sets
192.168.10.101:...	3	1	0	0	30,002
TOTAL	3	1	0	0	30,002

5 성능 테스트

- 프로토타입 시스템의 성능을 검증할 오픈소스 테스트 도구를 검토한 결과 성능이 우수하고 스크립트 작성이 편리한 k6를 선정
- 상용제품인 LoadRunner의 대안으로 jmeter가 널리 사용되고 있으나 XML을 채택하여 스크립트 작성이 비효율적이며 서버당 생성 가능한 부하가 낮음
- 최근에는 jmeter의 단점을 개선한 locust와 k6가 개발되어 주요 소프트웨어 업체에서 사용중

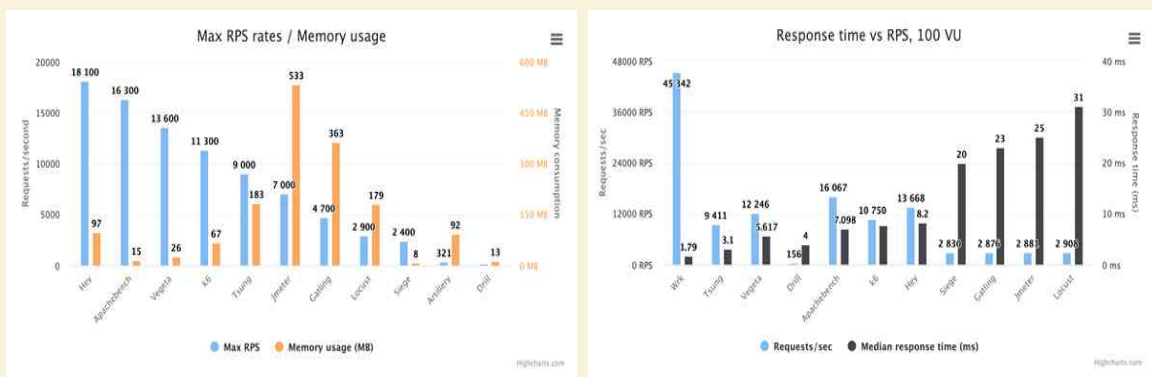
성능 테스트 도구 비교

구분	jmeter	ngrinder	locust	k6
제조사 ¹⁾	아파치 재단	네이버	Jonatan Heyman	그라파나 랩스
라이선스	Apache 2.0	Apache 2.0	MIT	AGPL 3.0
개발언어	Java	Java	Python	Go
스크립트 언어	XML	Groovy, Jython	Python	Javascript
GUI	지원	지원	지원	미지원 ²⁾
성능	낮음	낮음	보통	높음

주 : 1) 주요 관리자 또는 최초 개발자
2) 시각화 도구인 그라파나와 연동 가능

<참고5>

주요 성능 테스트 도구 성능



- k6를 사용하여 총 50,000건(가상유저 40개)의 이체요청을 실행한 결과, 건당 처리시간은 평균 17.64ms 소요되었으며 초당 2,204건을 처리

성능 테스트 도구(K6) 실행 결과

```
ncloud@core-perf1:~/k6$ k6 run --vus=40 --iterations=50000 --out influxdb=http://192.168.100.11:8086/k6 script.js
```



```
execution: local
script: script.js
output: InfluxDBv1 (http://192.168.100.11:8086)

scenarios: (100.00%) 1 scenario, 40 max VUs, 10m30s max duration (incl. graceful stop):
  * default: 50000 iterations shared among 40 VUs (maxDuration: 10m0s, gracefulStop: 30s)

data_received.....: 5.5 MB 240 kB/s
data_sent.....: 222 MB 9.8 MB/s
http_req_blocked.....: avg=4.75µs min=1.44µs med=2.37µs max=6.29ms p(90)=4.33µs p(95)=5.41µs
http_req_connecting.....: avg=1.45µs min=0s med=0s max=6.21ms p(90)=0s p(95)=0s
http_req_duration.....: avg=17.64ms min=11.07ms med=17.37ms max=103.09ms p(90)=21.11ms p(95)=23.5ms
  { expected_response:true }...: avg=17.64ms min=11.07ms med=17.37ms max=103.09ms p(90)=21.11ms p(95)=23.5ms
http_req_failed.....: 0.00% ✓ 0 ✗ 50000
http_req_receiving.....: avg=47.47µs min=13.57µs med=31.34µs max=9.97ms p(90)=63.25µs p(95)=86.43µs
http_req_sending.....: avg=47.42µs min=21.59µs med=38.64µs max=6.78ms p(90)=62.53µs p(95)=77.7µs
http_req_tls_handshaking.....: avg=0s min=0s med=0s max=0s p(90)=0s p(95)=0s
http_req_waiting.....: avg=17.54ms min=11ms med=17.28ms max=102.96ms p(90)=21ms p(95)=23.38ms
http_reqs.....: 50000 2204.740248/s
iteration_duration.....: avg=18.12ms min=81.95µs med=17.84ms max=110.38ms p(90)=21.61ms p(95)=24.07ms
iterations.....: 50000 2204.740248/s
vus.....: 40 min=40 max=40
vus_max.....: 40 min=40 max=40

running (00m22.7s), 00/40 VUs, 50000 complete and 0 interrupted iterations
default ✓ [=====] 40 VUs 00m22.7s/10m0s 50000/50000 shared iters
```