Getting Started Guide

Tmax 6 Fix#1

TMAXSOFT

저작권 공지

제한된 권리

상표 공지

오픈소스 소프트웨어 공지

본 제품의 일부 파일 또는 모듈은 다음의 라이선스를 준수합니다. : openssl-0.9.7.m, zlib-1.1.4, expat-2.0.0, netsnmp, DCE1.0, pthread, google-diff-match-patch, libevent, getopt

관련 상세한 정보는 제품의 다음의 디렉터리에 기재된 사항을 참고하시기 바랍니다. : ${INSTALL_PATH}/license/oss_licenses

안내서 이력

제품 버전	안내서 버전	발행일	비고
Tmax 6 Fix#1	3.1.1	2024-09-24	-
Tmax 6 Fix#1	2.1.2	2019-09-11	-
Tmax 6	2.1.1	2015-07-31	-

목차

용어 해설

트랜잭션 (Transaction)

하나의 완전한 일의 단위로 하나의 트랜잭션은 여러 가지 일을 포함한다.

Atomicity

하나의 완전한 일의 단위이다. 완전히 수행되는 경우와 수행되지 않는 2가지 경우만 존재한다.

Consistency

트랜잭션의 성공적인 수행 결과를 공유 자원에 갱신한다. 트랜잭션이 실패하는 경우 공유 자원을 원래 상태로 유지한다.

Isolation

트랜잭션의 영향을 받은 공유 자원의 변동사항은 트랜잭션이 commit 되기 전에는 다른 트랜잭션에 영향을 미치지 않는다.

Durability

트랜잭션의 결과가 commit 된 후에는 언제나 보존된다.

DTP (Distributed Transaction Processing)

하나의 트랜잭션에 여러 RM(Resource Manager)이 관여하여 처리한다.

전역 트랜잭션(Global Transactions)

하나 이상의 RM들을 하나의 일의 단위로 취급한다. 시스템에서 생성되는 일은 자동으로 commit 되어야 한다.

Commit

터미널에 나타나는 한 화면에 대응되는 개념이다.

Rollback

트랜잭션의 실패나 사용자 임의에 의해 트랜잭션의 결과를 처리 이전의 원래 상태로 복구한다.

TP-Monitor(Transaction Processing Monitor)

각종 프로토콜에서 동작하는 세션과 시스템 및 데이터베 이스 사이의 최소 처리단위인 트랜잭션을 감시하여 일관성 있게 보관 및 유지하는 역할을 하는 트랜잭션 관리 미들웨어이다.

다운사이징(DOWNSIZING)

중앙 집중식 메인프레임(Mainframe) 환경을 개방형 분산 시스템 환경으로 바꾸는 것이다.

미들웨어(Middleware)

분산 컴퓨팅 환경에서 단일 사용자 환경을 제공하고 이기종 간 시스템의 네트워크를 연결하 거나 클라이언트와 서버 간의 통신을 담당하거나 또는 컴퓨터와 컴퓨터의 연결을 담당하는 시스템 소프 트웨어이다.

WAS(Web Application Server)

웹에서 트랜잭션을 처리하고 이기종간 상호 통신 기능(J2EE)을 제공하는 서비스이다.

MOM(Messaging Oriented Middleware)

메시지를 큐라고 불리는 전달 중계소에 넣어 처리하고 큐에 의한 메시지 관리 기능을 제공(비동기적)하 는

서비스이다.

Database Access System

분산 환경에서 복수 개의 데이터베이스 서버들을 일관된 방법으로 이용할 수 있는 환경을 제공하는 서 비스이다.

RPC System

네트워크에서 다른 컴퓨터에 있는 프로그램을 실행(동기적)하는 서비스이다.

ORB(Object Request Broker)

클라이언트 객체가 ORB라는 소프트웨어 버스를 이용하여 원격지 서버의 메소드를 호출하는 기능을 제 공하는 서비스이다.

TMM (Tmax Manager)

Tmax 시스템을 운영 관리하는 핵심 프로세스로 Tmax 시스템의 모든 공유 정보와 CLL, CLH, TMS 및 AP(Application Program) 서버 프로세스를 관리한다.

CLL (Client Listener)

클라이언트와 Tmax의 연결을 담당하는 프로세스로 클라이언트 접속 관리를 위한 PORT Listener를 설 정해서 클라이언트로부터 요청을 받는다.

CLH (Client Handler)

클라이언트 핸들러이다. 클라이언트와 서버 사이를 중계하며, 서비스를 제공하는 업무처리 서버에 서 비스를 요청하고, 서버에 대한 연결 및 관리를 한다.

TMS (Transaction Management Server)

데이터베이스 관리 및 분산 트랜잭션 처리를 담당하는 프로세스로 데이터베이스 관련 시스템에서 동작 한다. XA 서비스에서 발생하는 commit/rollback을 RM(Resource Manager)에 전달한다.

TLM (Transaction Log Manager)

트랜잭션이 발생할 때 실제 CLH가 commit을 수행하기 전에 TLM을 통해서 트랜잭션 로그를 저장한다.

RQS (Reliable Queue Server)

Tmax 시스템의 디스크 큐(Disk Queue)를 관리하는 프로세스로 파일에 발생하는 읽기/쓰기를 수행한 다.

GW (Gateway Process)

여러 도메인으로 구분된 경우에 도메인 간의 통신을 담당한다.

Tmadmin (Tmax Administrator)

Tmax 관련 정보 모니터링 및 환경파일 변경 등을 관리한다.

RACD (Remote Access Control Daemon)

Tmax가 설치된 모든 도메인을 원격으로 통제한다.

TCS (Tmax Control Server)

CLH의 요청에 의해 비즈니스 로직을 처리하고 결과를 반환한다.

UCS (User Control Server)

CLH의 요청에 의해 비즈니스 로직을 처리하고 결과를 반환하면서 해당 프로세스가 control을 유지한다.

TIP (Tmax Information Provider)

시스템 환경 정보와 통계 정보를 확인하고, 시스템을 운용 및 관리한다.

TIM(Tmax Information MAP)

Tmax 시스템을 운용하는 핵심 정보로 Tmax에서 관리하는 공유 메모리를 말한다. TIM은 엔진 프로세스 중에 TMM 프로세스에 의해서 생성된다.

도메인 소켓(Domain Socket)

UNIX 도메인 소켓 통신 방식은 소켓 API를 수정없이 사용하며, 파일을 이용해서 내부 프로세스 간에 통신을 하는 방식이다.

SLM (System Load Management)

정의된 부하 비율로 분산 처리하는 방식이다.

DDR (Data Dependent Routing)

데이터 값에 따라 분산 처리하는 방식이다. 여러 노드에서 공통된 서비 스를 제공하면 데이터 범위에 따라 노드 간 라우팅을 할 수 있도록 지정한다.

DLM (Dynamic Load Management)

부하 비율에 따라 동적으로 처리 그룹을 선택하는 방식이다. 특정 노드에 부하가 집중되는 경우 Tmax 동적 부하 조절 방법에 따라 부하를 분산하여 전체 시스템의 처리량 을 증가시키고 처리 시간을 단축한다.

TCS (Tmax Control Server)

클라이언트가 요청하면 수동적으로 실행되는 프로세스로 대부분은 이 프로세스가 사용된다.

UCS (User Control Server)

호출자의 요청이 없어도 능동적으로 데이터를 전달할 수 있는 프로세스로 Tmax만의 고유 기능 이다.

POD (Processing On Demand)

클라이언트의 요청이 있을 때만 서버 프로세스가 기동되어 처리할 수 있도록 하는 방식이다.

RCA (Raw Client Agent)

멀티 스레드 방식으로 프로세스를 효율적으로 처리할 수 있는 다중 포트를 지원한다.

SCA (Simple Client Agent)

Non-Tmax 클라이언트/Tmax 클라이언트 모두 수용할 수 있는 다중 포트를 지원한다.

2PC(Two-phase commit) protocol

둘 이상의 동종 및 이종의 데이터베이스가 관련된 전역 트랜잭션에서는 트랜잭션의 속성을 보장하기 위해 2 단계(2 Phase) commit을 지원하여 데이터 무결성을 보장하고 API를 제공한다. 1단계 (Prepare Phase), 2단계 (Commit Phase)

RQ(Reliable Queue)

서비스 수행도중 장애 등으로 인하여 요청 내용이 사라지는 것을 방지하 여 서비스가 신뢰성있게 처리되도록 한다.

HMS(Hybrid Messaging System)

Tmax의 기능으로 Sender와 Receiver의 느슨한 결합(loosely coupled)을 위한 통신 매개체이며, Queue

방식과 Topic 방식을 지원한다.

WebT(Web Transaction)

클라이언트/서버 환경의 미들웨어 제품인 Tmax와 Java 애플리케이션 프로그 램 사이의 트랜잭션 서비스를 지원한다.

STRING 버퍼

NULL 값으로 끝나는 문자열로 따로 버퍼의 길이를 명시할 필요가 없다.

CARRAY, X_OCTET 버퍼

길이가 지정된 Byte 열로 구성된 버퍼로 보통 바이너리 타입의 데이터를 보낼 때 사용되며 데이터 교환 시에는 반드시 길이를 명시해야 한다.

STRUCT, X_C_TYPE 버퍼

C 언어의 구조체를 데이터 통신에 사용하고자 할 때 사용한다.

X_COMMON 버퍼

멤버 타입이 char, int, long으로 한정된 C 구조체이다.

FIELD 버퍼

필드키와 데이터 값을 한 쌍으로 관리하는 데이터 버퍼로 모든 원시 타입의 데이터를 담을 수 있다.

1. TP-Monitor 소개

본 장은 Tmax를 이해하기 위해 미들웨어와 TP-Monitor에 대한 개념과 기능에 대해 설명한다.

1.1. 개요

TP-Monitor(Transaction Processing Monitor)는 각종 프로토콜에서 동작하는 세션과 시스템 및 데이터베이스 사이의 최소 처리단위인 트랜잭션을 감시하여 일관성 있게 보관 및 유지하는 역할을 하는 트랜잭션 관리 미들웨어이다.

Tmax는 TP-Monitor 기반의 제품이며, 본 장에서는 Tmax를 이해하기 위한 기본 개념인 미들웨어와 TP-Monitor에 대해서 설명한다.

1.2. 미들웨어

중앙 집중식 메인프레임(Mainframe) 환경이 운영과 비용 등에서 여러 가지 문제가 나타나면서 업무별로 호스트를 분리하는 즉, 개방형 분산 시스템 환경으로 바꾸는 다운사이징(DOWNSIZING) 방법론이 도입되기 시작했다.

그러나 개방형 분산환경은 메인프레임에서 사용하던 업무를 여러 서버로 분산하여 관리하면서 서로 다른 운영체제 간의 통신이나 각 서버 프로그램 간의 연계, 호환성의 문제가 발생하게 되었다. 이러한 이유로 서로 다른 시스템이나 서버 프로그램, 네트워크 자원을 하나의 단일 사용자 환경으로 사용하려는 요구가 생겼다.

다음은 중앙 집중식 메인프레임 환경과 개방형 분산 시스템 환경의 문제점에 대해서 설명한 표이다.

- 중앙 집중식 메인프레임 환경의 문제점

구분	문제점
비용적 측면	• 고가의 도입 비용 • 유지보수 비용 과다 지출
운영적 측면	• 업무 프로세스보다는 메인프레임 환경이 최우선시 되므로 사용자 환경이 고려되지 않음
시스템 측면	• 이기종 간 통신의 어려움 • 다른 시스템으로 프로그램 이식 곤란 • 업무 확장의 어려움

- 개방형 분산 시스템 환경의 문제점

구분	문제점
비용적 측면	• 네트워크, DBMS 등 전문 기술 요구

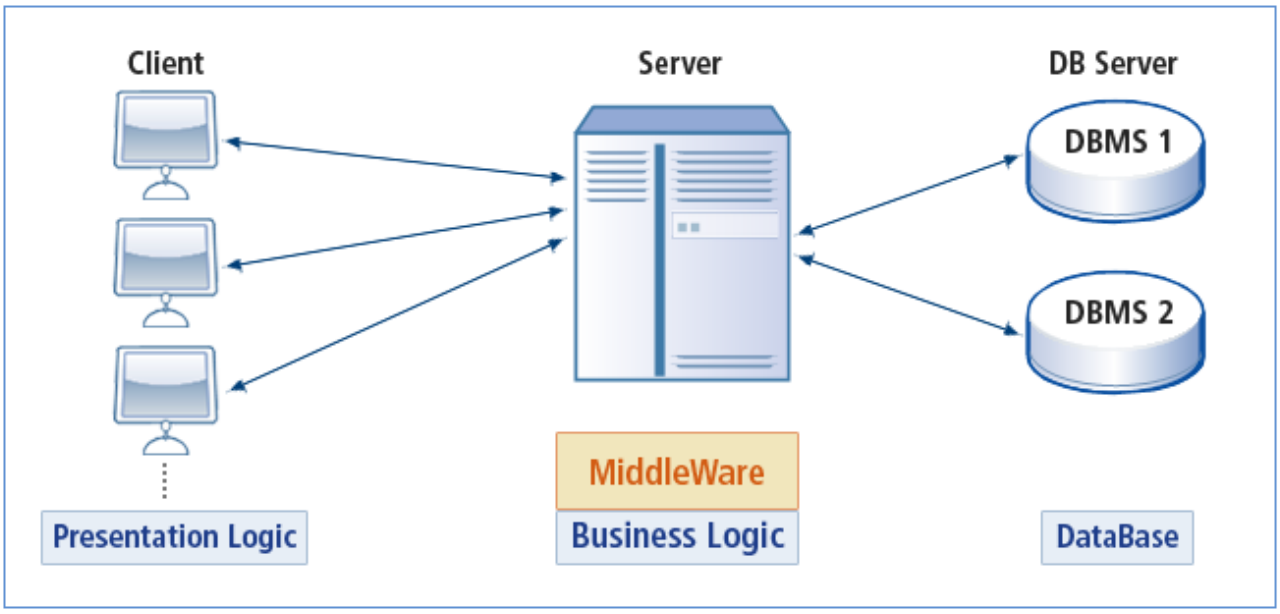
구분	문제점
운영적 측면	• 비효율적인 시스템 운영상의 문제 • 사용자 환경이 고려되지 않음 • 분산 환경에 따른 관리 문제 • 시스템의 관리 및 감독의 어려움 • 장애가 발생하는 경우 대처가 어려움 • 다른 서버 간 운영 방법 상이 • 복수 공급자에 따른 문제 • 사용자 증가에 따른 급격한 성능 저하 • 서버 간 부하 차이
시스템 측면	• 프로세스 관리 • 다양한 통신 방식 필요 • 시스템 보안 기능 취약 • 이기종 데이터베이스 및 전역 트랜잭션 처리의 어려움 • 이기종 서버 간 프로그램 이식성 문제 • 개발상의 문제(OS, 개발 언어 등이 너무 다양) • 미들웨어 한 종류로 이질적인 분산 환경에서 트랜잭션 처리 및 프로세스 관리의 기능을 제공

이러한 개방형 분산환경의 문제점을 해결하기 위해서 개발된 소프트웨어가 미들웨어이다.

미들웨어는 분산 컴퓨팅 환경에서 단일 사용자 환경을 제공하고 이기종 간 시스템의 네트워크를 연결하거나 클라이언트와 서버 간의 통신을 담당하거나 또는 컴퓨터와 컴퓨터의 연결을 담당하는 시스템 소프트웨어이다. 미들웨어는 이기종의 하드웨어나 프로토콜, 통신 환경 등을 하나로 연결해서 애플리케이션이나 운영 환경 사이에 원만한 통신이 가능하도록 해준다.

클라이언트와 데이터베이스 서버는 직접 통신을 하지 않고 미들웨어가 두 시스템 간의 통신을 연계한다. 미들웨어에는 업무에 필요한 비즈니스 로직을 가지고 있어서 클라이언트는 미들웨어와의 통신만 고려하고 서버 프로그램도 미들웨어와 통신만 고려하면 된다. 미들웨어를 통해서 클라이언트와 데이터베이스 서버는 하나의 단일 시스템 환경으로 구축이 가능해진다. 이기종 머신에서 사용하는 다수의 데이터베이스 환경에서도 시스템 통합을 보장할 수 있고 데이터의 호환성 및 일관성을 보장할 수 있다. 이러한 환경은 최소의 리소스로 최대의 성능을 제공할 수 있다.

다음은 미들웨어를 사용하여 클라이언트/서버 환경을 구축한 시스템 구성도이다.



미들웨어 동작 원리

미들웨어 제품은 각 제품의 용도와 목적에 따라 다음의 6가지로 나누어진다.

- TP-Monitor(Transaction Processing Monitor)

  이질적인 분산 환경에서 트랜잭션을 처리하고 각종 처리 절차를 관리하는 기능을 제공하는 서비스이다. Tmax는 미들웨어의 다양한 제품군 중에 TP-Monitor 제품군에 속한다.

- WAS(Web Application Server)

  웹에서 트랜잭션을 처리하고 이기종 간 상호 통신 기능(J2EE)을 제공하는 서비스이다.

- MOM(Messaging Oriented Middleware)

  메시지를 큐라고 불리는 전달 중계소에 넣어 처리하고 큐에 의한 메시지 관리 기능을 제공(비동기적)하는 서비스이다.

- Database Access System

  분산 환경에서 복수 개의 데이터베이스 서버들을 일관된 방법으로 이용할 수 있는 환경을 제공하는 서비스이다.

- RPC System

  네트워크에서 다른 컴퓨터에 있는 프로그램을 실행(동기적)하는 서비스이다.

- ORB(Object Request Broker)

  클라이언트 객체가 ORB라는 소프트웨어 버스를 이용하여 원격지 서버의 메소드를 호출하는 기능을 제공하는 서비스이다.

1.3. TP-Monitor

초기에 대부분의 업무 시스템은 메인프레임 기반의 중앙 집중식 환경으로 구성되어 있었다. 중앙 집중식 환경은 비용이나 관리에 여러 가지 단점을 나타내기 시작했다. 이러한 중앙 집중식 환경의 문제점을 보완하기 위해 대두된 것이 개방형 분산 시스템이다.

그러나 개방형 분산 시스템 또한 시스템 운영이나 관리에 또 다른 문제가 나타났다. 이러한 문제점을 해결하기 위해서 도입된 소프트웨어가 미들웨어이다. 미들웨어 중에 TP-Monitor는 프로토콜에서 동작하는 세션과 시스템 및 데이터베이스 사이의 최소 처리 단위인 트랜잭션을 감시하여 일관성있게 보관 및 유지하는 역할을 하는 트랜잭션 관리 미들웨어이다.

다음은 TP-Monitor의 주요 기능이다.

- 애플리케이션 개발 용이

  복잡한 업무 프로세스를 개발할 때 데이터 중심에서 기능 중심으로 분산해서 개발한다. 기존 메인프레임 환경은 비즈니스 로직과 데이터 처리 로직이 혼재되어 개발되었다. 따라서 메인프레임 환경에서 애플리케이션의 개발은 상당히 어려웠다.

  그러나 TP-Monitor를 사용하면 미들웨어에는 비즈니스 로직을 구현하고 클라이언트 프로그램은 사용자에게 제공하는 모듈만 구현하면 된다. 그리고 데이터베이스 서버 프로그램은 데이터 관리에 대한 기능만 구현하면 된다. 이러한 기능의 분리는 애플리케이션의 개발을 쉽게 한다.

- 효율적인 애플리케이션 관리

  분산된 각 업무 시스템을 TP-Monitor를 통해서 관리하므로 각 애플리케이션을 효율적으로 관리할 수 있다.

- 이기종 DBMS 자원관리

  DBMS의 트랜잭션을 통합 관리하므로 이기종 데이터베이스 간의 자원 관리가 가능하다.

- 부하 조절

  최적의 자원 사용에 대해 관리하므로 부하를 분산하고 분산 트랜잭션을 지원한다.

- 수행 속도와 신뢰성

  적은 서버 자원으로 많은 클라이언트를 관리하여 DBMS의 오버헤드를 줄이고 응답시간을 향상시킨다.

2. Tmax 소개

본 장은 Tmax의 개념, 구조, 기능, 특징, 도입시 고려할 사항에 대해서 설명한다.

2.1. 개요

Tmax는 Transaction Maximization의 약어로 트랜잭션 처리 극대화를 의미한다. Tmax는 시스템의 분산 환경에서 이기종 컴퓨터 간의 트랜잭션 처리를 완벽히 보장하면서 부하를 분산시키고 에러가 발생하는 경우 적절한 조치를 담당하는 TP-Monitor이다.

트랜잭션의 특성을 지원하면서 사용자에게는 최적의 개발환경을 제공하고, 클라이언트/서버 환경에서 최적의 솔루션을 제공하며, 성능 개선은 물론 모든 장애에 완벽하게 대처한다.

Tmax는 분산 트랜잭션 프로세싱의 국제 표준인 X/Open DTP(Distributed Transaction Processing) 모델을 준수하고 국제 표준기구인 OSI(Open Systems Interconnection group)의 DTP 서비스에 대한 기능적 분산과 기능 구성 요소 간 API 및 시스템 인터페이스 정의에 따라 개발되었다. 또한, 분산 환경에서 이기종 간의 투명한 업무처리 및 OLTP(On-Line Transaction Processing)를 지원하고 트랜잭션 처리의 ACID(Atomic, Consistent, Isolated, Durable: Transaction Properties) 특성을 만족하게 한다.

Tmax는 투명한 업무처리를 통해 성능을 극대화하고, 중요도가 높은 기간계 업무를 완벽하게 처리하여 개발자와 시스템 관리자에게 최적화된 컴퓨팅 환경을 제공한다. 또한, 금융, 공공, 통신, 제조, 유통 등 전 산업 분야에서 하루 수천만 건의 트랜잭션이 발생하는 중요도 높은 시스템에서 부하를 조절하고 장애발생을 방지하여 고객 시스템의 안정성을 보장한다. 또한 Large-scale OLTP 애플리케이션 개발이나 항공 및 호텔, 병원, 국방 분야나 은행 온라인 업무나 신용카드 승인 업무, 고객 및 판매 관리 업무와 같은 다양한 사업분야와 업무에서 사용되고 있다.

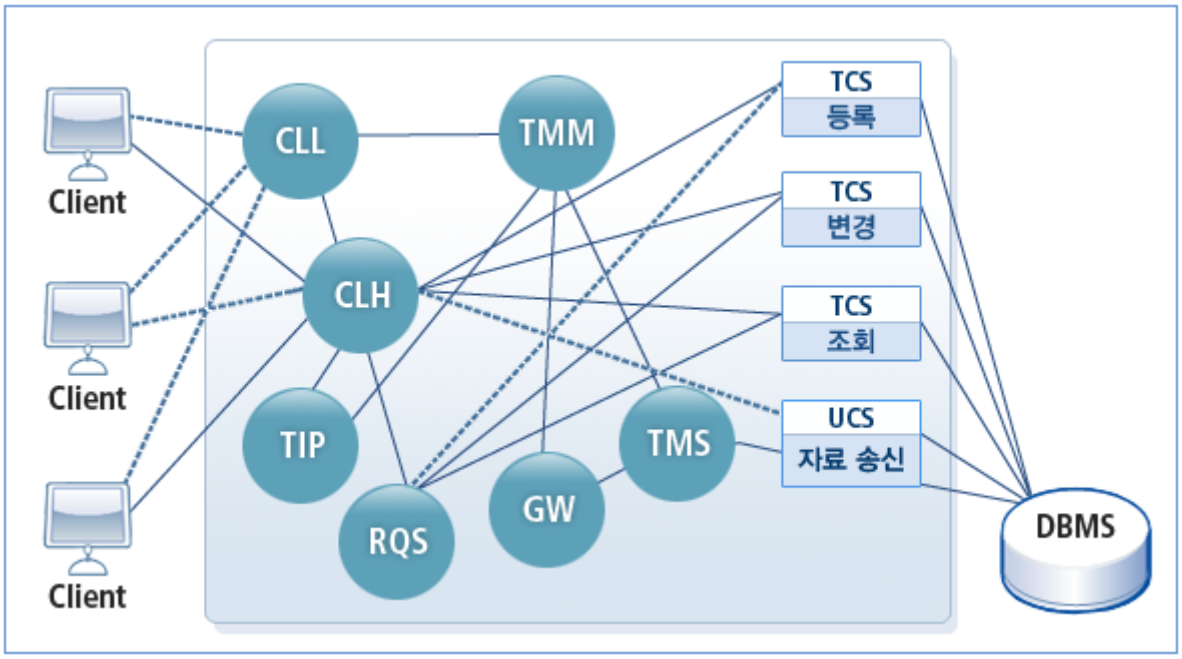
트랜잭션에 대한 자세한 사항은 Tmax Application Development Guide의 "트랜잭션"을 참고한다.

2.2. Tmax 구조

본 절에서는 시스템의 구성과 기능에 대해서 설명한다.

2.2.1. 시스템 구성

다음은 Tmax를 구성하는 시스템 구성에 대한 그림이다.



시스템 구성

- TMM(Tmax Manager)

  Tmax 시스템을 운영 관리하는 핵심 프로세스로 Tmax 시스템의 모든 공유 정보와 CLL(Client Listener), CLH(Client Handler), TMS(Transaction Management Server) 및 AP(Application Program) 서버 프로세스를 관리한다.

  다음은 TMM의 주요 기능에 대한 설명이다.

주요 기능	설명
공유 메모리 할당	설정한 환경 정보를 **cfl** 명령어로 컴파일한 바이너리는 엔진이 기동되는 시점에 TMM 프로세스가 공유 메모리(Shared Memory)에 로딩을 한다. TMM은 로딩된 운영 정보를 이용해서 시스템을 운영한다.
프로세스 관리	TMM은 모든 시스템에 대한 운영과 종료의 주체가 된다.
로그 관리	Tmax에서 발생하는 시스템 로그(slog)와 사용자 로그(ulog)를 관리한다.

- CLL(Client Listener)

  클라이언트와 Tmax의 연결을 담당하는 프로세스로 클라이언트 접속 관리를 위한 PORT Listener를 설정해서 클라이언트로부터 요청을 받는다.

- CLH(Client Handler)

  클라이언트 핸들러이다. 클라이언트와 서버 사이를 중계하며 서비스를 제공하는 업무처리 서버에 서비스를 요청하고, 서버에 대한 연결 및 관리를 한다. tpcall과 같은 함수를 통해 들어오는 클라이언트의 요청을 해당 서버에게 전달한다. 또한, XA 서비스 환경에서 XID 채번과 commit/rollback 요청을 중계한다.

- TMS(Transaction Management Server)

  데이터베이스 관리 및 분산 트랜잭션 처리를 담당하는 프로세스로 데이터베이스 관련 시스템에서 동작한다. XA 서비스에서 발생하는 commit/rollback을 RM(Resource Manager)에 전달한다.

- TLM(Transaction Log Manager)

  트랜잭션이 발생할 때 실제 CLH가 commit을 수행하기 전에 TLM을 통해서 트랜잭션 로그를 저장한다. 트랜잭션 로그는 tlog에 저장된다.

- RQS(Reliable Queue Server)

  Tmax 시스템의 디스크 큐(Disk Queue)를 관리하는 프로세스로 파일에 발생하는 읽기/쓰기를 수행한다.

- GW(Gateway Process)

  여러 도메인으로 구분된 경우에 도메인 간의 통신을 담당한다.

- Tmadmin(Tmax Administrator)

  Tmax 관련 정보 모니터링 및 환경파일 변경 등을 관리한다.

- RACD(Remote Access Control Daemon)

  Tmax가 설치된 모든 도메인을 원격으로 통제한다.

- TCS(Tmax Control Server)

  CLH의 요청에 의해 비즈니스 로직을 처리하고 결과를 반환한다.

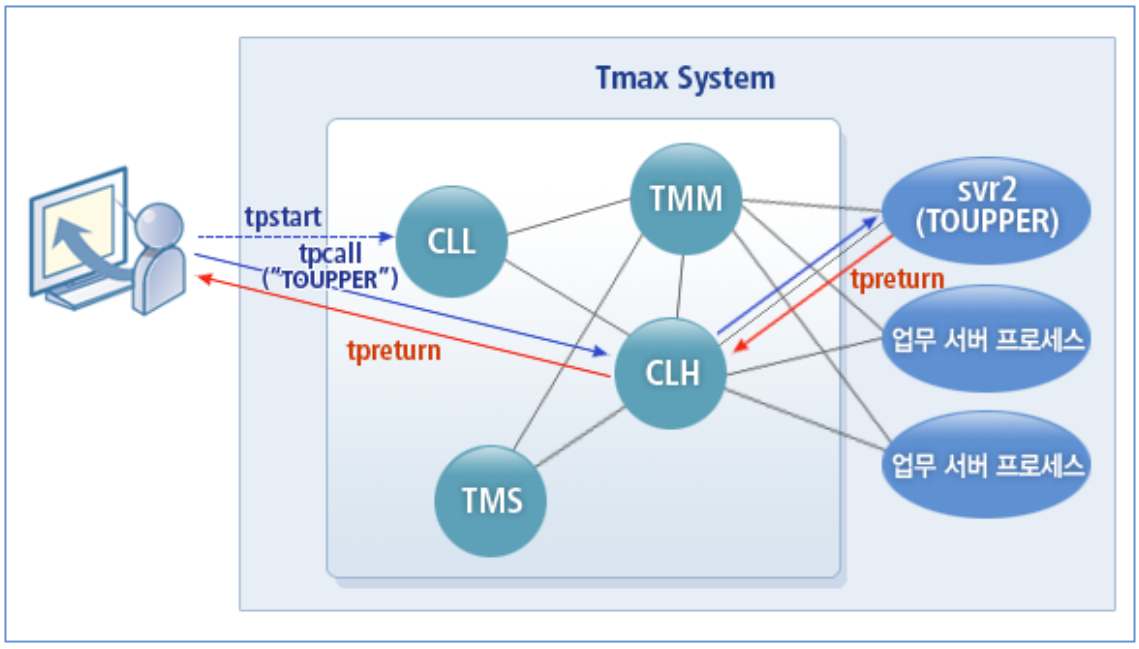
- UCS(User Control Server)

  CLH의 요청에 의해 비즈니스 로직을 처리하고 결과를 반환하면서 해당 프로세스가 control을 유지한다.

- TIP(Tmax Information Provider)

  시스템 환경 정보와 통계 정보를 확인하고, 시스템을 운용 및 관리한다. (boot/down only)

다음은 Tmax 시스템이 서비스를 수행하는 과정을 보여주는 그림이다.



Tmax 시스템 서비스 수행

다음은 Tmax 시스템이 서비스를 수행하는 절차에 대한 설명이다.

1. 클라이언트가 tpstart하면 CLL 프로세스가 연결 요청을 수행하고 내부 동작을 통하여 CLH 프로세스로 연결 처리된다.
2. 서비스 요청이 있을 경우 CLH 프로세스에서 모든 서비스를 처리한다.
3. CLH 프로세스는 클라이언트의 서비스 요청(tpcall)을 수신하고, 요청 서비스를 분석하며, 업무 서버 프로세스(svr2)에 서비스를 매핑한다.
4. 업무 서버 프로세스(svr2)는 서비스 처리 후 CLH에 결과를 반환(tpreturn)한다.
5. CLH는 서비스 처리 결과를 수신해서 클라이언트에 전송한다.

2.2.2. TIM

TIM(Tmax Information MAP)은 Tmax 시스템을 운용하는 핵심 정보로 Tmax에서 관리하는 공유 메모리를 의미한다. TIM은 엔진 프로세스 중에 TMM 프로세스에 의해서 생성된다.

TIM은 역할에 따라서 다음과 같이 4개로 구분된다.

- Tmax 시스템 설정 정보

  Tmax 환경파일인 <tmconfig.m> 파일을 관리하기 위해 공유 메모리에 로딩하고 필요한 경우에 참조한다.

- Tmax 시스템 운영 정보

  Tmax 시스템 운영에 대한 정보를 관리한다. 시스템에 장애가 발생했을 경우의 대응 방법, 부하 분산을 위한 Load Balancing에 대한 기준 정보, 여러 장비로 구성된 시스템의 경우에는 각 서비스에 접근하기 위한 Naming 서비스 정보와 애플리케이션 위치 정보 등을 관리한다.

- 애플리케이션 상태 정보

  시스템에 로딩되는 애플리케이션의 프로세스 상태(Ready/Not Ready/Running 등)를 관리한다.

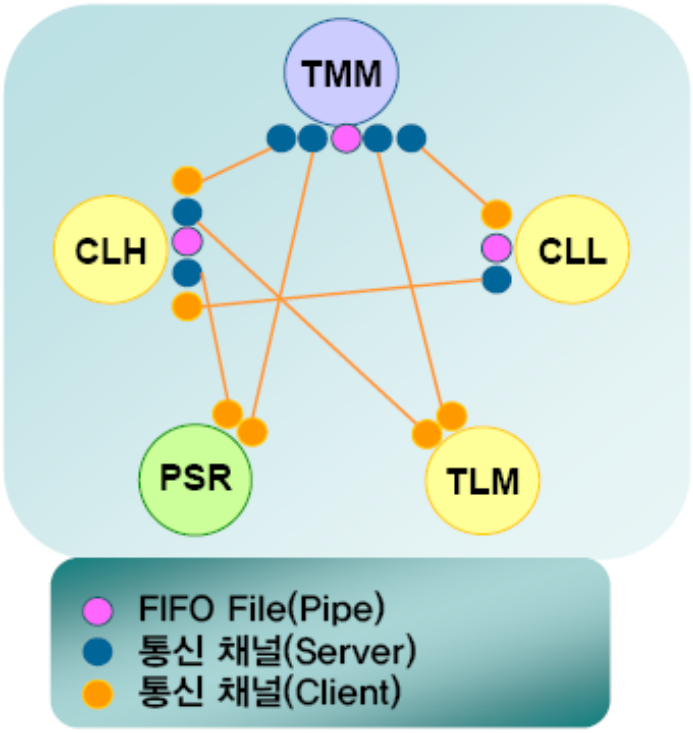
- 분산 트랜잭션 정보

  RM과 통신의 중계 역할 데이터베이스 운영 정보 및 트랜잭션 처리를 위한 채번 정보를 관리한다.

2.2.3. 소켓 통신

Tmax는 UNIX 도메인 소켓(Domain Socket) 통신 방식을 사용한다. UNIX 도메인 소켓 통신 방식은 소켓 API를 수정없이 사용하며, 파일을 이용해서 내부 프로세스 간에 통신을 하는 방식이다. 이는 Port 기반의 외부 통신 방식과는 달리 파일을 이용한 내부 통신 방식으로 더욱 안정적이고, 속도가 빠르다는 장점이 있다. 파일을 통한 통신 방식과 커널에서 메시지를 관리한다는 특징은 FIFO(Named Pipe)와 비슷하나, FIFO와 달리 양방향 통신이 가능하여, 다수의 클라이언트/서버 환경을 구축하기가 쉽다.

다음은 CLL, CLH, TLM이 UNIX 도메인 소켓 통신을 하는 과정에 대한 그림이다. PSR로 표시된 애플리케이션 서버는 CLL, CLH, TLM과 연결되어 진다.



도메인 소켓 통신

2.3. Tmax 기능

다음은 Tmax의 주요 기능에 대한 설명이다.

- 프로세스 관리

  Tmax는 3-tier 기반의 클라이언트/서버 환경을 제공하여 클라이언트마다 생성되는 서버의 업무처리 프로세스 수를 조절하여 시스템을 최적의 환경으로 활용할 수 있도록 관리한다.

- 트랜잭션 관리

  분산 트랜잭션 처리될 때 2PC(2 Phase commit)를 지원하여 데이터 무결성을 보장하고 간단한 몇 가지 함수(tx_begin, tx_commit, tx_rollback 등)를 제공하여 전역 트랜잭션을 용이하게 한다. 또한, MultiThread 방식의 트랜잭션 매니저를 제공하여 적은 자원 대비 높은 효율성을 제공하고 동적 로깅으로 에러가 발생하는 경우 신속히 대응하므로 Recovery/Rollback에 의한 안정성을 보장한다. 모든 트랜잭션은 중앙에서 관리하므로 트랜잭션에 대한 스케줄링이나 관리가 쉽다.

- 부하 조절

  Tmax는 다음과 같은 3가지 방식을 통해 부하 조절 기능을 제공하여 전체 시스템의 처리량을 증대시키고 처리시간을 단축한다.

  - SLM(System Load Management)
  - DDR(Data Dependent Routing)
  - DLM(Dynamic Load Management)

- 장애 대책

하드웨어적인 장애가 발생하는 경우 Load Balancing을 통한 장애 대책과 서비스 백업에 의한 정상 운영이 가능하다. 서버 프로세스가 다운되는 소프트웨어적인 장애가 발행하는 경우 프로세스가 재시작되므로 중단 없는 서비스가 제공된다.

- Naming 서비스

  Naming 서비스를 통해 분산 시스템 내의 서비스 위치 정보를 이름으로 제공하여 간단하고 쉽게 서비스 호출이 가능하고 위치 투명성을 제공한다.

- 프로세스 제어

  Tmax는 다음과 가은 3가지 방식의 데이터 전달 프로세스를 제공한다. 이에 대한 자세한 내용은 프로세스 제어를 참고한다.

  - TCS(Tmax Control Server)
  - UCS(User Control Server)
  - POD(Processing On Demand)

- RQ(Reliable Queue) 기능 제공

  서비스 수행 중 장애 등으로 요청 내용이 사라지는 것을 방지하는 디스크 큐(Disk Queue) 처리를 통해서 데이터를 보존하고 신뢰성 있는 처리를 보장한다.

- 보안 기능

  Tmax는 Fiffie-Hellman 알고리즘 바탕의 데이터 보호 기능을 제공하고 UNIX에서 제공하는 보안기능을 포함하여 다음의 3단계 보안기능을 지원한다.

  - 1단계: 시스템 접속 인증

    전체 Tmax 시스템(도메인)에 대한 단일 암호를 설정하고 이 암호를 등록한 클라이언트에 한해서만 Tmax 시스템 접속을 허락한다.

  - 2단계: 사용자 인증

    Tmax 시스템에 등록되어 있는 사용자 ID에 한해서만 인증을 거쳐 Tmax에서 제공하는 서비스 사용을 허락한다.

  - 3단계: 서비스 접근 인증

    특별히 보안이 요구되는 서비스에 대하여, 권한이 있는 사용자에게만 해당 서비스를 지원한다. Tmax 4부터 지원한다.

    보안기능에 대한 자세한 내용은 Tmax Application Development Guide의 "보안 시스템"을 참고한다.

- 편리한 API 및 다양한 통신방식 지원

  동기형 통신(Synchronous), 비동기형 통신(Asynchronous), 대화형 통신(Conversational), 요구 재요청(Request Forwarding), 알람 통지(Notify), 메시지 동시 전달(Broadcasting)과 같은 다양한

통신방식을 지원하며, 이를 위해 편리한 API를 제공한다.

- 시스템 관리와 자원 관리
  - 프로세스 상태, 서비스 큐잉 상태, 서비스 처리 건수, 평균 서비스 처리시간과 같은 전체 시스템 진행 상황을 모니터링할 수 있고 시스템 상태 및 큐 관리 시스템 통계 분석 및 보고서 작성이 가능하다.
  - 애플리케이션과 데이터베이스를 통합하여 관리하므로 자원의 효율적인 관리가 가능하다.
- 멀티 도메인 및 다양한 게이트웨이 서비스 제공

  장거리 분산 시스템의 상호 데이터를 교환하고, 다른 플랫폼 기반의 시스템 간에 손쉽게 연동할 수 있으며, SNA CICS, SNA IMS, TCP CICS, TCP IMS, OSI TP 등과 같은 다양한 게이트웨이 모듈을 지원한다. 도메인 간 트랜잭션 서비스 처리 및 라우팅 기능이 가능하다.
- 다양한 클라이언트 에이전트 제공

  2-tier 시스템에서 3-tier 시스템으로 변환이 쉬운 다양한 에이전트를 제공한다.

구분	설명
RCA(Raw Client Agent)	MultiThread 방식으로 프로세스를 효율적으로 처리할 수 있는 다중 포트를 지원한다.
SCA(Simple Client Agent)	Non-Tmax 클라이언트/Tmax 클라이언트 모두 수용할 수 있는 다중 포트를 지원한다.

- 다양한 개발 방식 지원

구분	설명
RDP(Real Data Processor)	UDP 통신 데이터 유형을 사용해서 Tmax 시스템을 경유하지 않는 직접적인 데이터 전달이 가능하도록 지원한다.
Window 제어	다중 Window 설정을 위한 클라이언트 라이브러리(WinTmax Library)를 제공하여 동시 다발적인 데이터 수용이 가능하다.

- 확장지원
  - 웹과의 연동

    클라이언트/서버 환경과 웹 환경을 Java Applet/Servlet, PHP 등을 통해 연동함으로써 빠른 응답시간을 보장하고, 시스템 성능을 향상시킬 수 있다. Tmax에서는 서비스 연동을 쉽게 하기 위해 **WebT**를 제공한다. WebT에 대한 자세한 내용은 "Tmax WebT User Guide"를 참고한다.
  - 메인프레임 연동

    IBM 등 레거시 시스템에 존재한 애플리케이션 서비스를 Host-link를 이용하여 클라이언트/서버 환경의 애플리케이션 서비스와 동일하게 접근할 수 있다.
  - 다른 미들웨어 전환 용이

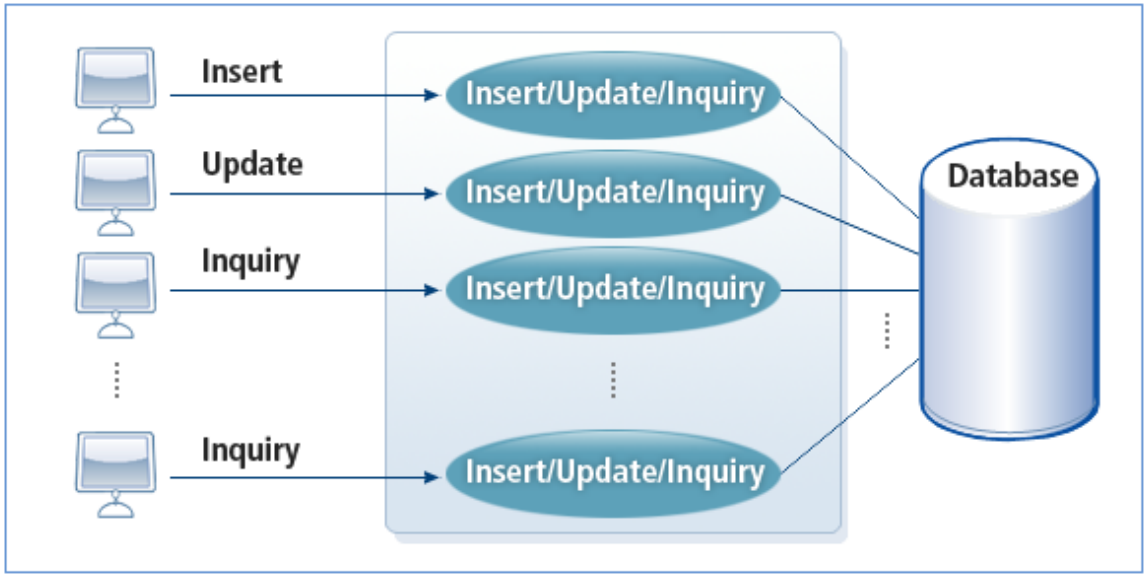
    다른 미들웨어(Tuxedo, TopEnd, Entera)로 개발된 시스템도 소스 수정 없이 Tmax로 쉽게 전환이 가능하며, 더욱 향상된 성능과 수준 높은 기술 서비스 및 비용 절감의 효과를 기대할 수 있다.

2.3.1. 프로세스 관리

기존의 2-tier 기반의 클라이언트/서버 환경은 애플리케이션 개발의 가장 일반적인 방법으로 하나의 클라이언트에 서버 프로세스가 하나씩 생성되는 방식이다. 즉, 클라이언트의 수가 증가하면 서버 프로세스의 수도 함께 증가한다. 이러한 클라이언트와 서버 프로세스의 증가는 프로세스 생성에 필요한 시간과 파일 및 데이터베이스의 open/close에 많은 시간이 소요되고, 하나의 서버는 연결된 클라이언트에서만 사용이 가능하므로 서버 프로세스의 현저한 사용 저하와 많은 관리 비용의 문제가 발생했다.

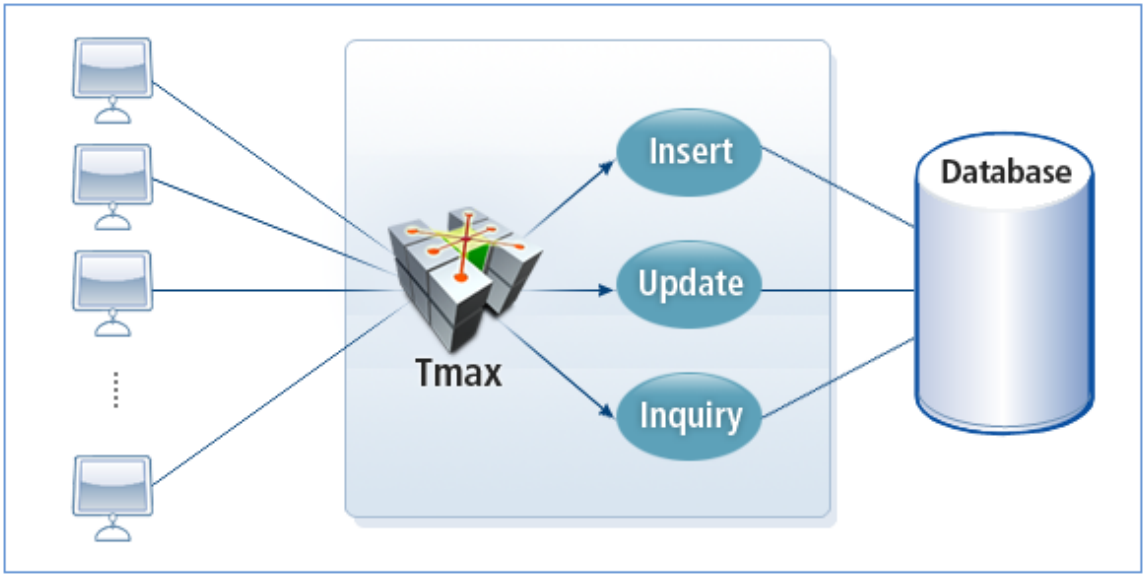
2-tier 기반의 클라이언트/서버 환경은 이러한 문제를 해결하기 위해 미들웨어인 TP-Monitor를 이용하여 3-tier로 구성한 클라이언트/서버 구조를 도입하였다. TP-Monitor인 Tmax는 생성되는 프로세스의 수를 조절하고, idle 중인 서버 프로세스를 스케줄링하며, 서버 프로세스의 Tuning을 통한 최적의 시스템을 운영한다.

다음은 2-tier 클라이언트/서버 구조를 나타내는 그림이다.



2-tier 클라이언트/서버 구조

다음은 3-tier 클라이언트/서버 구조를 나타내는 그림이다.



3-tier 클라이언트/서버 구조

3-tier 환경은 2-tier 환경보다 성능, 확장성, 관리 기능, 장애 대책 측면에서 월등한 시스템으로 구축할 수 있고, 내부에서 프로세스 관리 모듈을 통해 안정적 서비스를 제공한다. 3-tier 환경에서 Tmax의 내부 프로세스 관리 모듈은 생성되는 프로세스 수 조절하고 Idle 중인 서버 프로세스 스케줄링한다. 또한 서버 프로세스의 Tuning을 통한 최적의 시스템을 운영한다.

다음은 2-tier와 3-tier의 클라이언트/서버 환경에 대한 비교이다.

- 적용분야

2-tier 환경	3-tier 환경
• 소규모 부서의 단위별 업무 • 단일 서버 • 소수 사용자(대략 50 미만) • 배치 업무 분야	• 전사적 업무 • 다중 서버 • 대규모 사용자(50명 이상) • OLTP성 업무 및 대량 거래 처리

- 장점

2-tier 환경	3-tier 환경
• 프로그램 개발기간 단축(테스트 용이) • 초기 도입비용 절감 • 단순한 프로그램 개발 용이	• 애플리케이션 개발의 모듈화 • 이기종 H/W 및 데이터베이스 환경에서 상호 연동 가능 • 시스템 자원 최대한 활용하여 성능 향상 • 확장 용이 • 시스템 관리, 부하 분산, 장애 대책 추가 보안 기능

- 단점

2-tier 환경	3-tier 환경
• 거래량 증가될 때 급격한 성능 저하 • 다른 플랫폼 및 데이터베이스 환경에서 상호 연동 불가능 • 확장의 어려움 • 시스템 관리, 부하 분산, 장애 대책 추가 보안 기능 구현 불가능	• 프로그램 개발 시 장시간 소요(클라이언트/서버 결합 테스트) • 초기 도입 비용 증가 • 단순한 프로그램도 클라이언트/서버 분리 개발

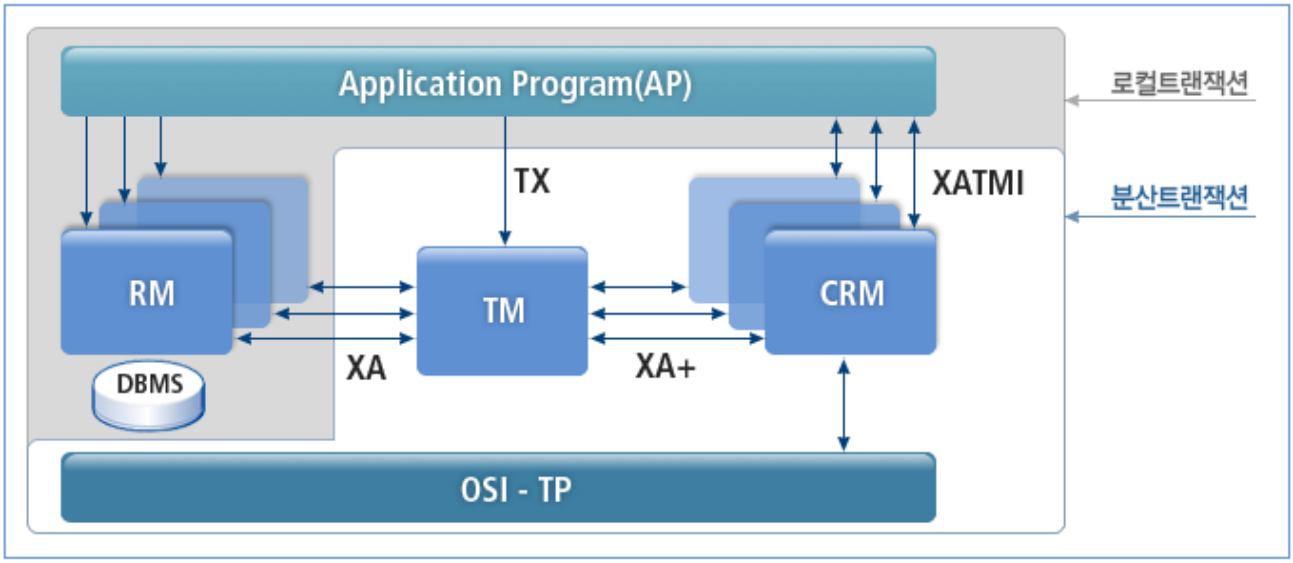
2.3.2. 분산 트랜잭션

트랜잭션은 논리적으로 하나의 단위로 처리함으로써 다양한 자원들을 일괄적으로 활용할 수 있으며 분산된 자원 간의 자료 무결성을 유지할 수 있다. 분산 트랜잭션은 네트워크상의 시스템 간의 트랜잭션을 의미하며, 트랜잭션의 ACID(Atomic, Consistent, Isolated, Durable: transaction properties) 특성을 만족하게 해야 한다. Tmax 시스템은 이기종 또는 동종 복수 DBMS 간의 분산된 트랜잭션에서 ACID를 보장한다.

구분	설명
원자성(**A**tomicity)	트랜잭션이 수행되거나 전혀 수행되지 않는(all or nothing) 2가지 경우만 존재해야 한다.
일관성(**C**onsistency)	트랜잭션의 성공적인 수행 결과를 하나의 일관된 상태에서 다른 일관된 상태의 공유 자원에 갱신한다.
고립성(**I**solation)	공유 자원을 바꿀 때 트랜잭션의 결과가 commit될 때까지 트랜잭션 외부로 나타나지 않는다.
영속성(**D**urability)	하위 시스템이나 media 오류를 거친 트랜잭션 commit의 결과를 바꾼다. 성공적으로 완료된 트랜잭션의 결과는 영구적으로 반영된다.

Tmax의 분산 트랜잭션 관리는 기본적으로 국제 표준인 X/Open의 DTP 모델을 준수한다. X/Open DTP 모델을 따라 AP(Application Program), TM(Transaction Manager), RM(Resource Manager), CRM(Communication Resource Manager)로 구성되어 있으며, X/Open DTP 모델을 준수하는 표준 함수의 집합체인 ATMI 함수를 지원한다. 또한 X/Open DTP를 준수하는 서로 다른 종류의 DBMS 사이에 발생하는 트랜잭션을 묶어서 처리한다.

다음은 X/Open DTP의 구조이다.



X/Open DTP의 구조

- AP(Application Program)

  DT(Distributed Transaction)의 Boundary를 제공한다.

- RM(Resource Manager)

  데이터베이스 등의 리소스에 접근 기능을 제공한다.

- TM(Transaction Manager)

  DT별로 ID를 생성하여 진행을 관리하고 완료 및 실패하는 경우 복구 기능을 제공한다.

- CRM(Communication Resource Manager)

분산 AP 간의 통신을 제어한다.

- OSI-TP(Open System Interconnection-Transaction Processing)

  서로 다른 TM 영역과의 통신을 담당한다.

분산 트랜잭션을 처리하는 경우 2단계(2 Phase) commit을 지원하여 데이터 무결성을 보장하고 API를 제공하여 전역 트랜잭션을 용이하게 한다. 분산 트랜잭션 관리란, 여러 지역으로 분산된 환경에서 다수의 이기종 하드웨어 플랫폼 및 데이터베이스를 이용하여 실행하는 트랜잭션의 관리를 의미한다.

- 2PC(Two-phase commit) protocol

  둘 이상의 동종 및 이종의 데이터베이스가 관련된 전역 트랜잭션에서는 트랜잭션의 속성을 보장하기 위해 2PC를 사용한다. 2PC(Two-phase commit)는 둘 이상의 데이터베이스가 연동할 때 ACID 속성을 완전히 보장하기 위해 2단계 처리를 하는 것을 말한다.

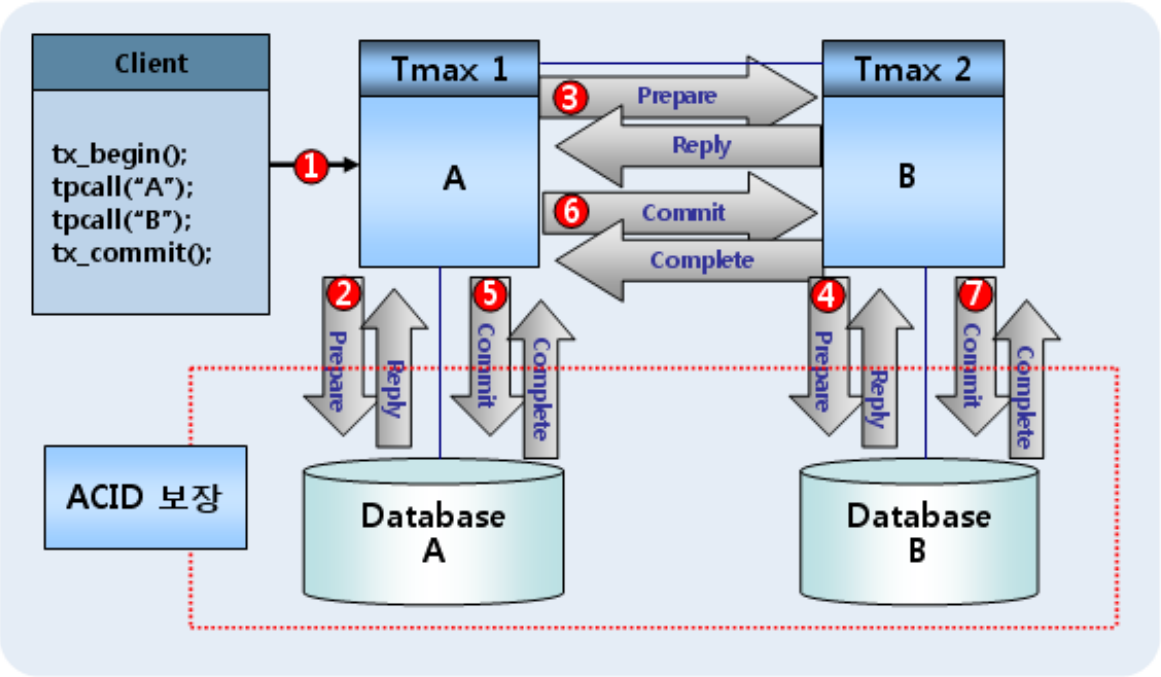
  - 1단계 : Prepare Phase

    트랜잭션에 관련된 모든 데이터베이스에 트랜잭션을 처리할 준비가 되었는지 확인한다. 모든 데이터베이스로부터 준비가 완료되면 신호를 전달한다. 하나의 분산 트랜잭션으로 묶여 있는 각각의 데이터베이스나 네트워크, 서버 등이 commit이나 rollback을 할 수 있는지를 체크하고 데이터베이스를 준비시키는 단계이다.

  - 2단계 : Commit Phase

    모든 데이터베이스로부터 정상 신호를 받았으면 commit이 된다. 하나라도 비정상 신호를 받았다면 rollback를 처리하여 전역 트랜잭션을 완료한다. commit 메시지를 모든 노드에 보내고 각 노드에서는 RM으로 commit 요청을 수행하는 단계이다. 단, Commit Phase에 참여하는 모든 노드가 Commit Complete가 완료되어 tx_commit()을 요청한 쪽으로 정상 처리되었음을 알리는 시점에 데이터 변경 작업이 완료된다는 점에 주의한다.

  다음은 2PC(Two-phase commit) 동작 과정을 보여주는 그림이다.



2PC(Two-phase commit) 동작

- 전역 트랜잭션(Global Transaction)

  다수의 이기종 하드웨어 및 데이터베이스를 하나의 논리적인 단위(트랜잭션)로 취급하여 작업한다. 둘 이상의 동종 또는 이기종 시스템상에 존재하는 DBMS와 관련된 전역 트랜잭션 처리 시 2단계(2 Phase) commit을 지원하여 데이터 무결성을 보장함으로써 분산 트랜잭션을 완벽하게 처리한다. Tmax 시스템은 매우 간단한 함수(tx_begin, tx_commit, tx_rollback 등)를 제공하여 전역 트랜잭션을 용이하게 지원한다. 전역 트랜잭션 처리될 때 노드 간 통신은 클라이언트 핸들러를 통해 이루어진다.

  다음은 전역 트랜잭션의 동작에 대한 설명이다.

  - 1단계 : Prepare Phase

    분산 트랜잭션을 일으킨 노드(전역 Coordinator)가 분산 트랜잭션에 참석한 노드에 대해 commit이나 rollback을 수행해도 되는지를 확인하는 단계이다.

  - 2단계 : Commit Phase

    분산 트랜잭션에 참여한 노드가 전역 Coordinator에게 분산 commit해도 된다는 응답(Prepare 되었음)을 받고 트랜잭션을 commit한다. 어떤 노드라도 Prepare가 안되었다는 응답이 있으면 트랜잭션을 rollback한다.

- Recovery / Rollback

  트랜잭션이 실패하면 현재 사용 중인 RM의 내용이 바뀌었다 하더라도 이전의 내용으로 회복한다.

- 트랜잭션 중앙관리

  노드들이 물리적으로 멀리 떨어져 있더라도 노드 간에 발생하는 트랜잭션에 대하여 중앙에서 관리 및 통제한다.

- 트랜잭션 스케줄링

우선순위를 고려한 동시성 제어 기법을 지원한다.

2.3.3. 부하 조절

Tmax는 다양한 부하 조절 기능을 제공하여 전체 시스템의 처리량을 증대시키고 처리 시간을 단축한다. Tmax는 SLM(System Load Management)에 의한 부하 조절, DDR(Data Dependent Routing)에 의한 부하조절, DLM(Dynamic Load Management)에 의한 부하조절을 제공한다.

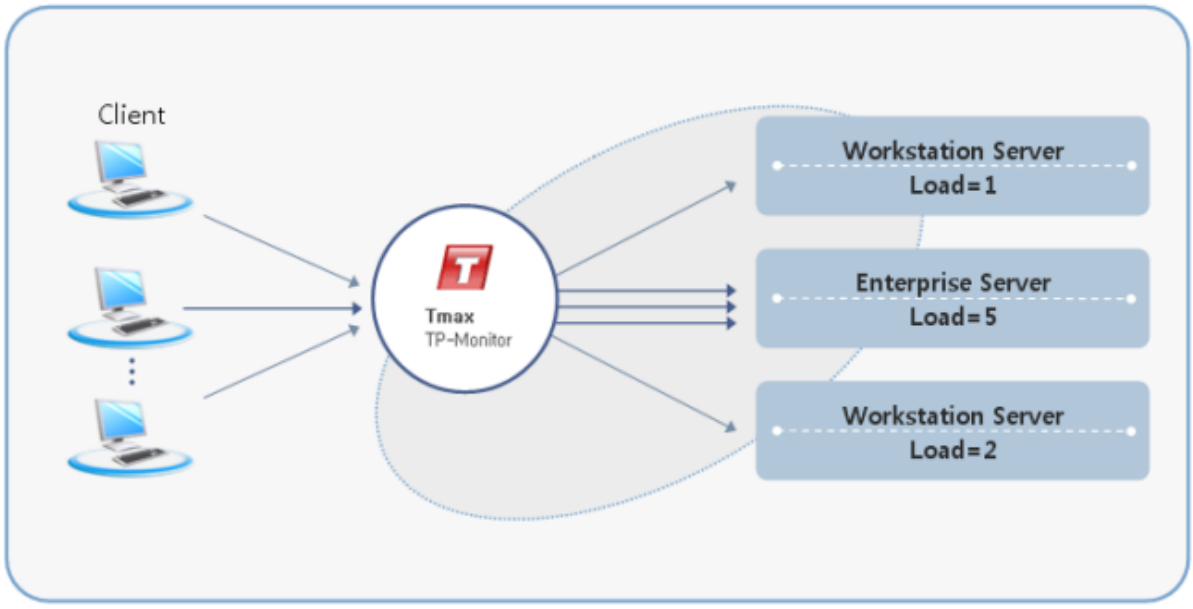
SLM에 의한 부하 조절

SLM(System Load Management)은 정의된 부하 비율로 분산 처리하는 방식이다. 하드웨어 성능에 따라 고유의 작업 처리량을 설정해서 한 노드의 서비스 요청량이 작업 처리량을 초과하면 다른 노드로 서비스 연결을 전환하는 방식이다. 노드마다 처리량을 다르게 설정하여 부하를 처리할 수 있다.

SLM은 다음과 같은 동작 방식으로 각 프로세스에서 처리된다.

1. 클라이언트의 요청을 CLH가 수신한다.
2. CLH는 TIM을 통해서 SLM 서비스인지를 판단해서 서버 그룹별 처리 건수를 확인한다.
3. CLH가 적절한 서버 그룹에 스케줄링을 한다.

다음은 SLM에 의해 부하를 조절하는 프로세스의 예로 Node 1, Node 2, Node 3에 작업 처리량을 각각 1:5:2로 설정해 놓으면 Node 1에서 1개의 작업을 처리하고 다음 작업은 Node 2, 다음 작업은 Node 3에서 처리한다. 이후 연속적인 3개의 작업은 Node 2에서 모두 처리한다.



SLM에 의한 부하 조절

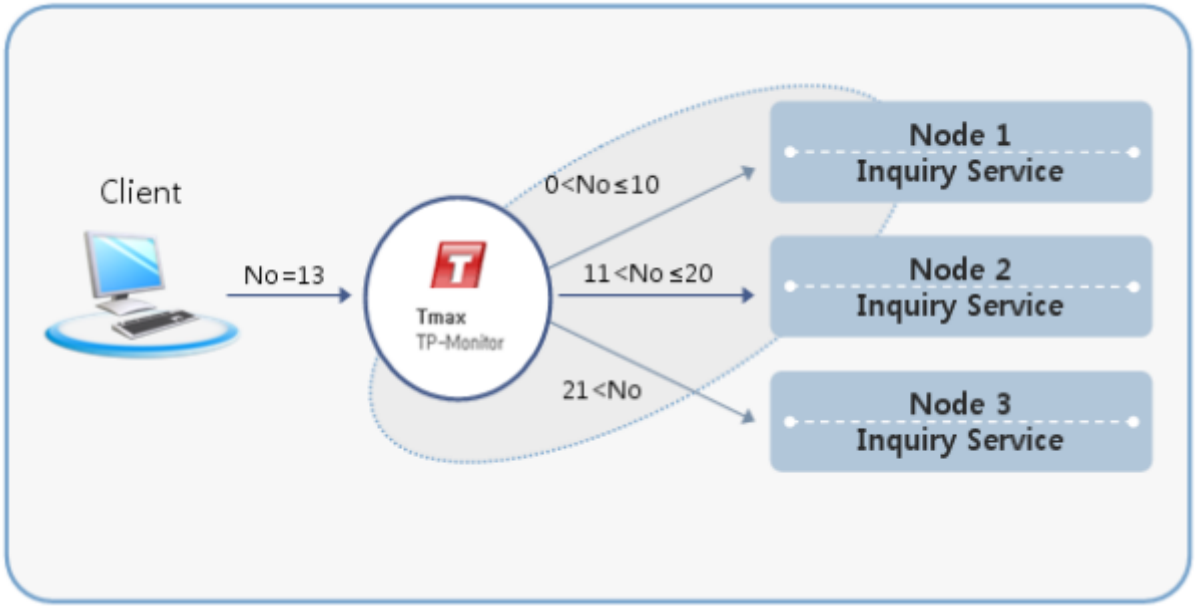
DDR에 의한 부하조절

DDR(Data Dependent Routing)은 데이터 값에 따라 분산 처리하는 방식이다. 여러 노드에서 공통된 서비스를 제공하면 데이터 범위에 따라 노드 간 라우팅을 할 수 있도록 지정한다. 입력된 필드의 값을 확인하여 적절한 서버 그룹으로 해당 서비스를 요청한다.

DDR은 다음과 같은 동작 방식으로 각 프로세스에서 처리된다.

1. 클라이언트의 요청을 CLH가 수신한다.
2. CLH는 TIM을 통해서 DDR 서비스인지를 판단해서 CLH가 구분 필드값을 확인한다.
3. CLH가 정의된 서버 그룹에 스케줄링을 한다.

다음은 DDR에 의해 부하를 조절하는 프로세스의 예로 연령층 별로 각각 다른 노드에서 고객조회 서비스를 제공한다면 Node 1에서는 0~10세까지, Node 2에서는 11~20세까지, 그리고 그 이외의 연령은 Node 3에서 처리되도록 한다.



DDR에 의한 부하조절

DLM에 의한 부하조절

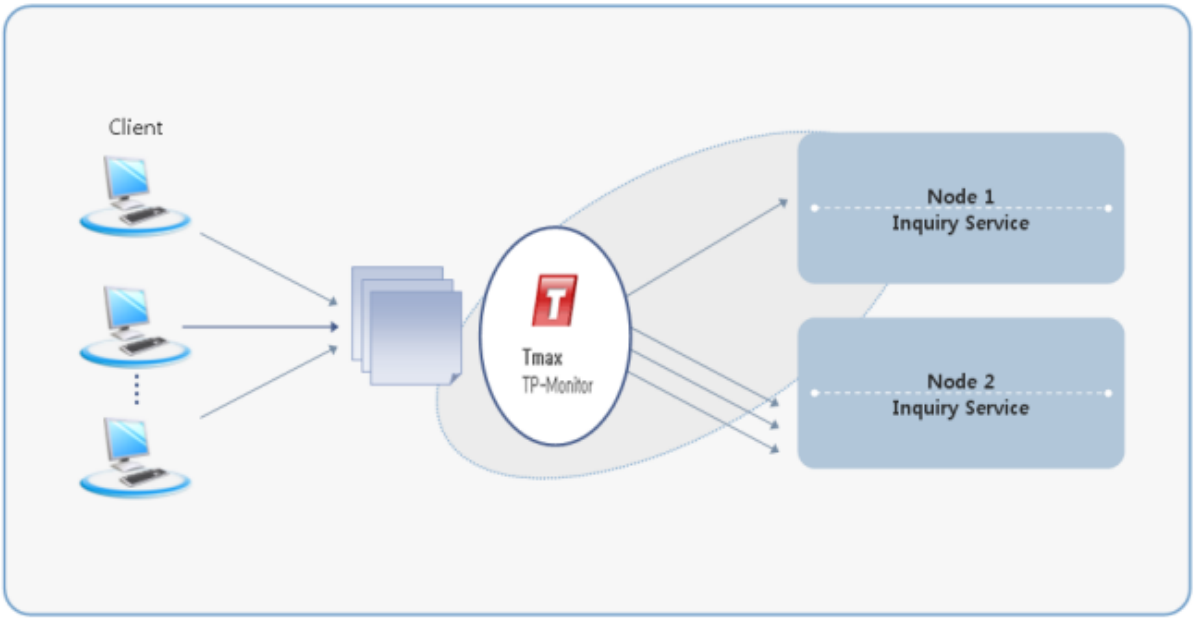
DLM(Dynamic Load Management)은 부하 비율에 따라 동적으로 처리 그룹을 선택하는 방식이다. 특정 노드에 부하가 집중되는 경우 Tmax 동적 부하 조절 방법에 따라 부하를 분산하여 전체 시스템의 처리량을 증가시키고 처리 시간을 단축한다. 시스템의 부하는 기동되어 있는 프로세스의 큐잉 상태로 판단된다.

Tmax 시스템은 프로세스별로 메모리 큐를 관리하여 요청된 서비스에 대해서 현재 매핑 가능한 프로세스가 없는 경우에는 메모리 큐에 저장한다. 메모리 큐에 쌓인 거래의 건수가 시스템의 부하를 의미한다.

DLM은 다음과 같은 동작 방식으로 각 프로세스에서 처리된다.

1. 클라이언트 요청을 CLH가 수신한다.
2. CLH는 TIM을 통해서 DLM 서비스인지를 판단해서 CLH가 서버별로 큐잉 건수를 확인한다.
3. CLH가 임계치에 도달하면 다음 서버 그룹에 스케줄링을 한다.

다음은 DLM에 의해 부하를 조절하는 프로세스의 예로 Node 1, Node 2에서 서비스를 동일하게 제공하고, Node 1에 서비스 요청이 집중되는 경우 Tmax에서 제공하는 동적 분산 알고리즘에 의해 부하를 분산하여 처리한다.



DLM에 의한 부하조절

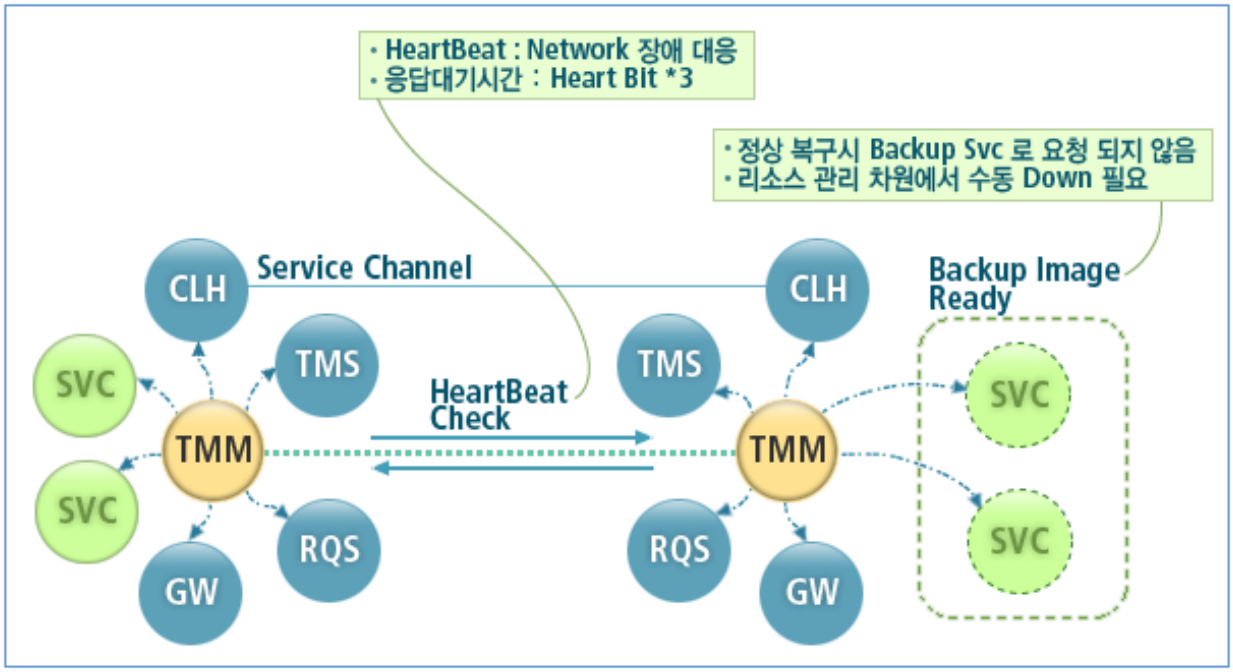
2.3.4. 장애 대책

Tmax는 시스템 자원의 높은 가용성을 보장하기 위하여 머신, 네트워크, 시스템, 서버 프로세스 등의 장애가 발생하는 경우 장애 대책을 통하여 중단없는 서비스를 제공할 수 있다. 장애는 하드웨어적인 장애와 소프트웨어적인 장애로 구분할 수 있다.

하드웨어 장애

하드웨어적인 장애가 발생하는 경우 Load Balancing을 통한 장애 대책과 서비스 백업에 의한 정상 운영이 가능하다. 서버 프로세스가 다운되는 소프트웨어적인 장애가 발생할 때에도 프로세스가 재시작되어 중단없이 서비스가 제공된다.

Tmax의 시스템 구성은 각 노드가 서로 다른 노드를 감시하는 Peer-to-Peer 방식이다. 따라서, 아무리 많은 노드라 해도 동일한 조건에서 즉각적으로 장애에 대응할 수 있다.

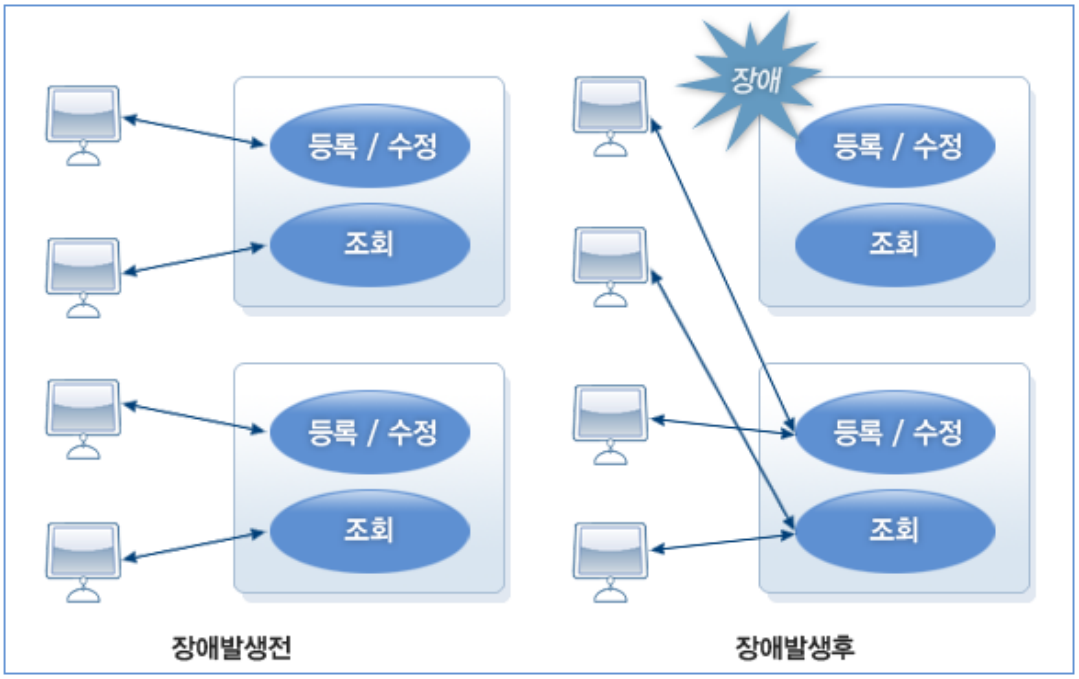


하드웨어 장애

하드웨어 장애는 다음과 같은 2가지의 장애 대책이 있다.

- 부하 조절에 의한 장애 대책

  하나의 서비스가 여러 노드에서 제공되는 경우에는 한 노드에서 장애가 발생하더라도 다른 노드에서 계속적인 서비스를 제공한다. 클라이언트에서 백업 노드로 재연결하고 서비스를 요청한다.



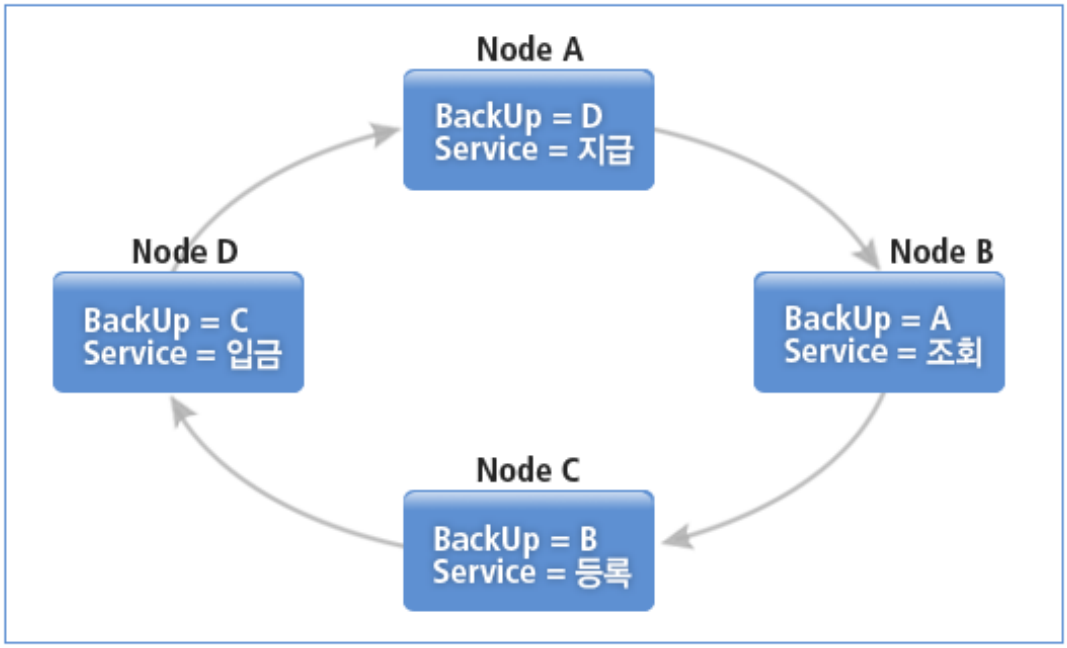
부하 조절에 의한 장애 대책

- 서비스 백업에 의한 장애 대책

  노드 장애가 발생하는 경우 다른 노드에서 미리 준비된 백업 프로세스를 동작시켜 서비스를 처리한다.

◦ 장애 발생 전(정상 운영)

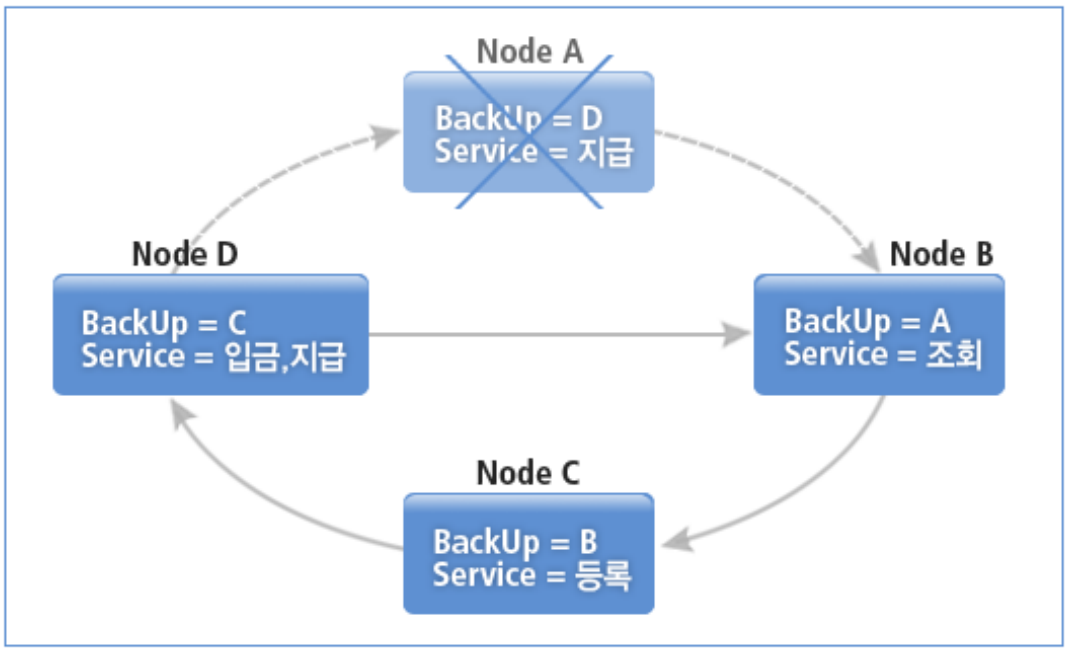
모든 노드에 대한 장애를 완벽히 지원(Peer-to-Peer 방식)



서비스 백업에 의한 장애 대책-장애 발생 전

◦ 장애 발생 후(정상 운영)

장애가 발생될 때 지정된 백업 노드에서 중단 없는 서비스 제공

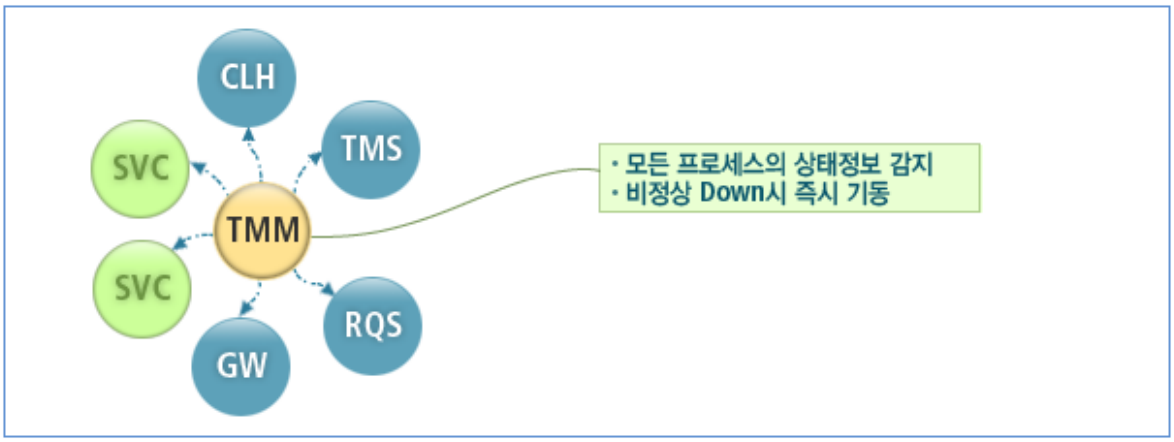


서비스 백업에 의한 장애 대책-장애 발생 후

소프트웨어 장애

프로그램 내부적인 버그에 의해서 혹은 사용자의 실수로 인하여 비정상적으로 서버 프로세스가 다운됐을 경우 자동으로 재시작될 수 있는 기능을 제공한다. 그러나 TMS, CAS, CLH와 같은 시스템 프로세스들은 제한 없이

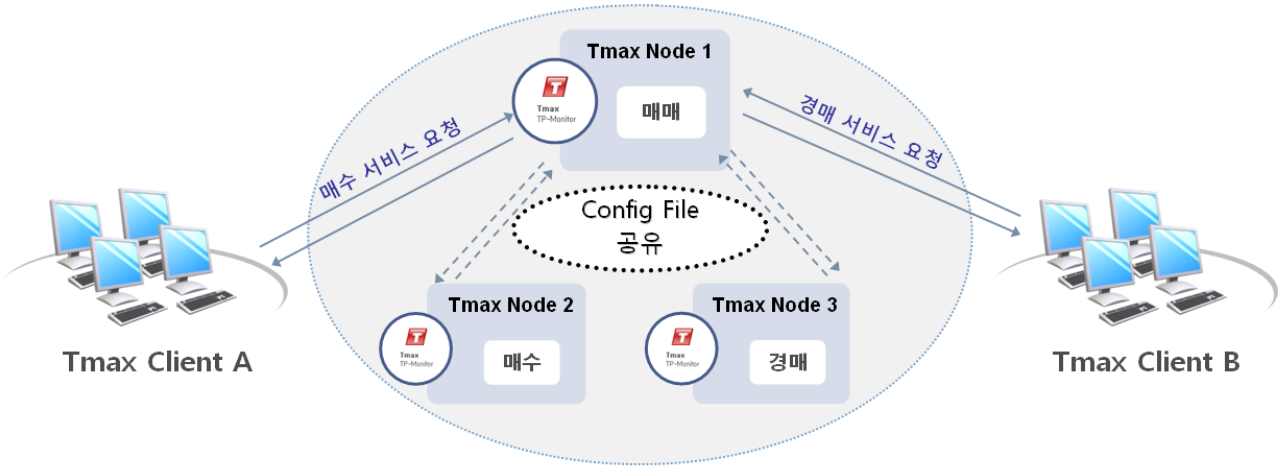
무한으로 재시작되고 이 프로세스들이 비정상적으로 종료되었을 때에는 처리 중이던 서버 프로세스도 같이 다운될 가능성이 있으므로 주의해야 한다.



소프트웨어 장애

2.3.5. Naming 서비스

Tmax에서 제공하는 Naming 서비스는 간단 명료한 서비스 호출이 가능하도록 하여 위치 투명성(Location Transparency)을 보장한다. Tmax는 Naming 서비스를 통해 분산 시스템 내의 서비스 위치 정보를 이름으로 제공하여 간단하고 용이한 서비스 호출이 가능하고 위치 투명성을 제공한다. Naming 서비스는 간단 명료한 서비스 호출이 가능하고 서비스명만 호출하여 해당하는 서비스를 받으므로 프로그래밍이 쉽다. 클라이언트는 서버의 주소를 알지 못해도 서비스명을 통해 서버의 정보를 얻을 수 있다.



Naming 서비스

2.3.6. 프로세스 제어

Tmax에서는 3가지 종류의 서버 프로세스를 지원한다.

- TCS(Tmax Control Server)

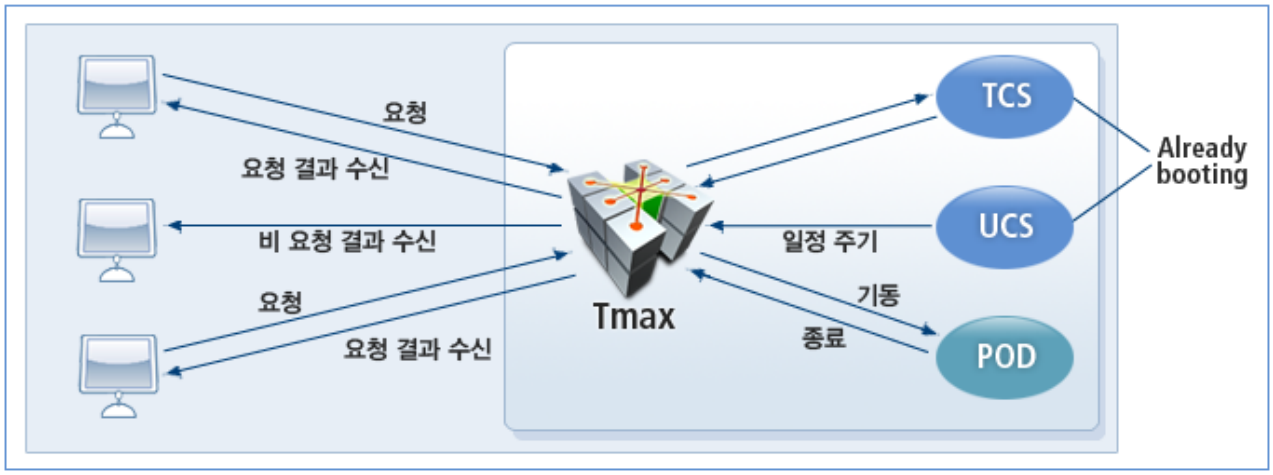
클라이언트가 요청하면 수동적으로 실행되는 프로세스로 대부분은 이 프로세스가 사용된다. 클라이언트의 요청을 처리할 서버는 미리 부팅된 상태이어야 한다. 클라이언트의 요청 프로세스를 처리하는 전형적인 처리 방법이다. 호출자의 요청을 Tmax 핸들러로부터 수신하여 일을 처리하고 그 결과를 Tmax에서 반환하고서 다음 요청을 기다린다.

- UCS(User Control Server)

  UCS는 호출자의 요청이 없어도 능동적으로 데이터를 전달할 수 있는 프로세스로 Tmax만의 고유 기능이다. 클라이언트의 요청을 처리할 서버는 미리 부팅된 상태이어야 한다. 클라이언트의 요청 없이 클라이언트에게 지속적으로 데이터를 보낼 수도 있으며, 동시에 TCS 프로세스와 동일하게 호출자의 요청을 받아들여 업무를 처리할 수도 있다. 즉, UCS는 TCS와 동일하게 클라이언트의 요청을 처리하면서 자발적이며 능동적으로 업무를 처리할 수 있는 기능이 부여된 프로세스이다.

- POD(Processing On Demand)

  POD는 클라이언트의 요청이 있을 때만 서버 프로세스가 기동되어 처리할 수 있도록 하는 방식이다. 클라이언트의 요청 때만 기동하고 업무 수행 후에는 종료되는 프로세스로 요청이 많지 않은 업무에 적합한 형태이다.

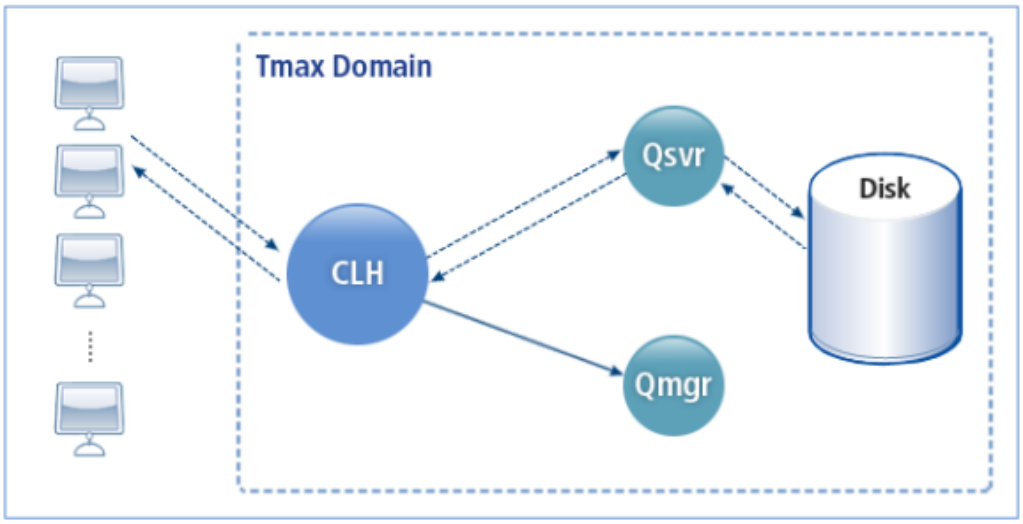


프로세스 유형

서버 프로세스 제어에 대한 자세한 내용은 Tmax Application Development Guide의 "서버 프로그램"을 참고한다.

2.3.7. RQ 기능

Tmax의 RQ(Reliable Queue)는 서비스 수행도중 장애 등으로 인하여 요청 내용이 사라지는 것을 방지하여 서비스가 신뢰성있게 처리되도록 한다. 많은 시간이 필요한 작업이라든지 반드시 신뢰성이 보장되어야 하는 업무의 경우 요청된 업무를 일단 디스크에 저장하고 처리한다. 시스템 장애 혹은 다른 치명적인 요인이 발생한 때에도 시스템 복구 후 해당 업무를 정상적으로 처리하는 방식이다.



RQ 프로세스

tpenq() 함수의 반환값을 통하여 요청된 서비스가 정확히 디스크에 저장되었는지 확인할 수 있다. 큐잉 업무처리는 Queue manager(Qmgr)가 독립적으로 담당함으로써 다른 업무처리에는 전혀 영향을 주지 않는다. tpdeq() 함수를 사용하여 Queue manager의 처리결과에 대한 정상 유무를 확인할 수 있다.

2.3.8. 보안 기능

Tmax는 Fiffie-Hellman 알고리즘 바탕의 데이터 보호 기능을 제공하고 UNIX에서 제공하는 보안기능을 포함하여 5단계 보안기능을 지원한다.

2.3.9. 시스템과 자원 관리

시스템 관리(System Management)

프로세스 상태, 서비스 큐잉 상태, 서비스 처리 건수, 평균 서비스 처리 시간과 같은 전체 시스템 진행 상황을 모니터링할 수 있고 시스템 상태 및 큐 관리 시스템 통계 분석 및 보고서 작성이 가능하다.

시스템 관리는 정적 시스템 관리, 동적 시스템 관리, 모니터링 및 운영(Administration) 기능을 지원한다.

- 정적 시스템 관리

  사용자의 환경에 따라 Tmax 시스템을 구성하는 서비스, 서버, 서버 그룹, 노드, 도메인 등에 대한 전반적인 시스템 환경을 설정한다.

- 동적 시스템 관리

  Tmax가 동작 중에도 각 구성요소를 항목별로 변경할 수 있다.

항목	설명
도메인(Domain)	서비스 타임아웃 시간, 트랜잭션 타임아웃 시간, 노드(machine) live check 시간 등을 변경한다.

항목	설명
노드(Node)	메시지 큐의 타임아웃 시간을 설정한다.
서버 그룹(Server group)	노드별 load 값, 부하 조절 방식 등을 변경한다.
서버(Server)	Max queue count, 큐잉 시 서버 start count, 서버 restart count, 대 서버 개수, 서버 우선순위 등을 변경한다.
서비스(Service)	서비스별 우선순위, 서비스 타임아웃 시간 등을 변경한다.

- 모니터링 및 운영(Administration) 기능
  - 동적환경 설정 변경이 가능하다.
  - 각종 자료를 출력하고 서버의 트랜잭션 처리량, 서비스별 처리 건수, 평균 처리 시간 등의 각종 통계 정보를 제공한다.

자원 관리(Resource Management)

애플리케이션과 데이터베이스의 통합 관리를 통해서 자원의 효율적인 관리가 가능하다.

기존 시스템에서는 전체 시스템에 대한 관리가 이루어지지 못해서 자원의 낭비가 발생해서 분산 시스템은 더 확실한 관리 기능이 요구되었다. Tmax는 전체의 분산 시스템에 대한 중앙 집중식의 모니터링 기능을 통해 애플리케이션을 관리한다.

단일 애플리케이션에 동종의 데이터베이스 혹은 이기종의 데이터베이스를 함께 사용하는 경우 Tmax는 다양한 종류의 자원을 애플리케이션 차원에서 통합관리가 가능하다.

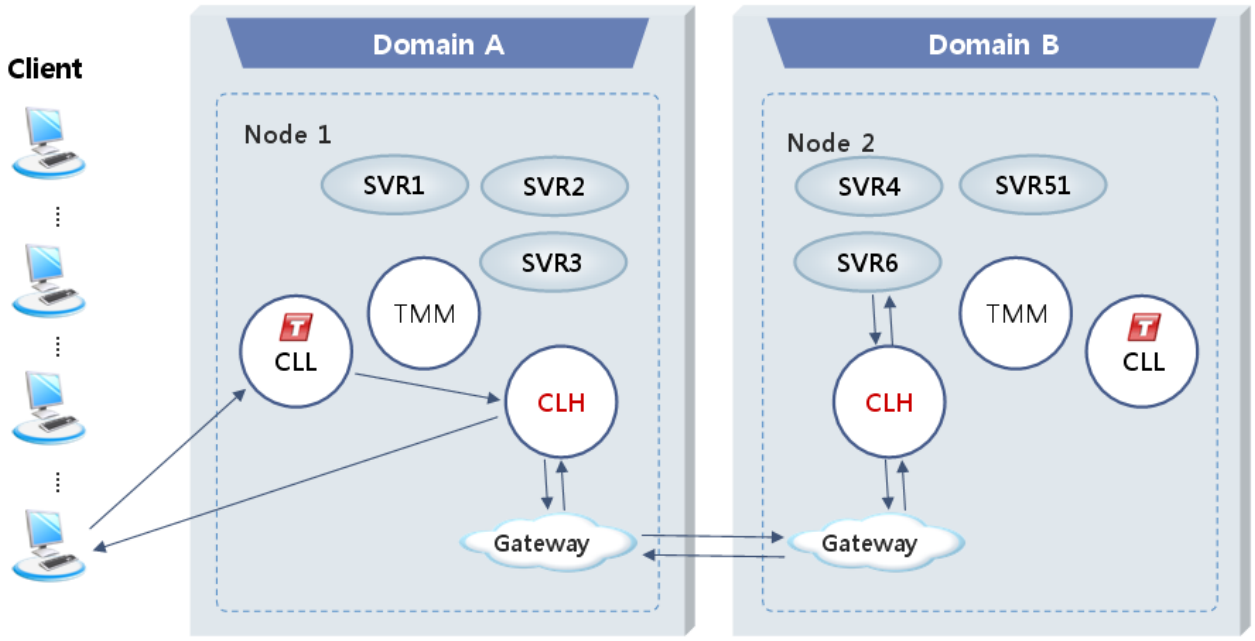
2.3.10. 멀티 도메인 및 다양한 게이트웨이 서비스 제공

Tmax 도메인은 Tmax에서 독립적으로 관리(start/down)되는 최상위 단위로 지역별 또는 업무별로 시스템을 여러 개의 도메인으로 분산 관리하는 때에도 도메인 게이트웨이를 통해 도메인 간의 연동뿐만 아니라 멀티 도메인 2PC 등의 기능을 지원한다.

Tmax는 장거리 분산 시스템의 상호 데이터를 교환하고, 다른 플랫폼 기반의 시스템 간 손쉬운 연동을 지원하며, SNA CICS, SNA IMS, TCP CICS, TCP IMS, OSI TP 등과 같은 다양한 게이트웨이 모듈을 지원한다. 도메인 간 트랜잭션 서비스 처리 및 라우팅 기능이 가능하다.

여러 노드를 한 도메인에서 관리할 때 발생하는 전체 노드 관리의 어려움, 노드 간 통신 트래픽 급증 등의 문제점을 해결할 수 있다. 또한, 어느 시스템에서든지 요청된 서비스의 처리가 가능하다. 멀티 도메인 환경에서 서비스 방식은 도메인 간 서비스 처리, 도메인 간 라우팅 , 도메인 간 트랜잭션으로 이루어진다.

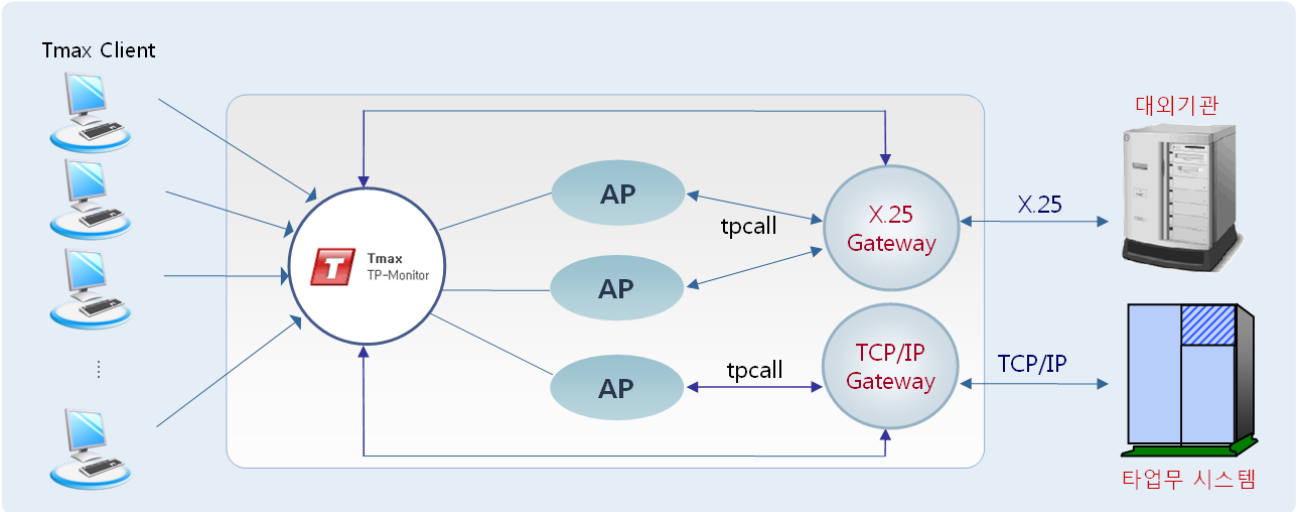
다음은 멀티 도메인 구성에 따른 서비스 호출 흐름에 대한 설명이다. 멀티 도메인은 게이트웨이를 통해 연결된다. 도메인 게이트웨이는 서로 연결을 맺을 때 서버 또는 클라이언트로 동작한다. 게이트웨이 간 기동 순서는 없다.



멀티 도메인 서비스 흐름

Tmax는 TCP/IP, X.25, SNA 등 다양한 게이트웨이를 제공해서 타업무 시스템/대외기관과의 연동을 쉽게 해준다. 게이트웨이는 효과적인 통신을 가능하게 하고 업무 로직과 연계 모듈 분리를 통해 관리의 편의성을 제공한다. 게이트웨이는 Tmax에 의해 관리되므로 자동으로 장애 복구가 가능하고 관리자에 의한 별도 관리가 필요없다.

다음은 게이트웨이의 동작에 대한 설명으로 클라이언트가 서비스를 요청하면 해당 서비스가 있는 도메인으로부터 그 처리 결과를 응답받는 것을 보여준다. 이때 양쪽 도메인에서는 게이트웨이를 통해 트랜잭션 서비스나 값에 따라 도메인 간에 라우팅할 수 있다.



Tmax 게이트웨이 동작

2.3.11. 다양한 클라이언트 에이전트 제공

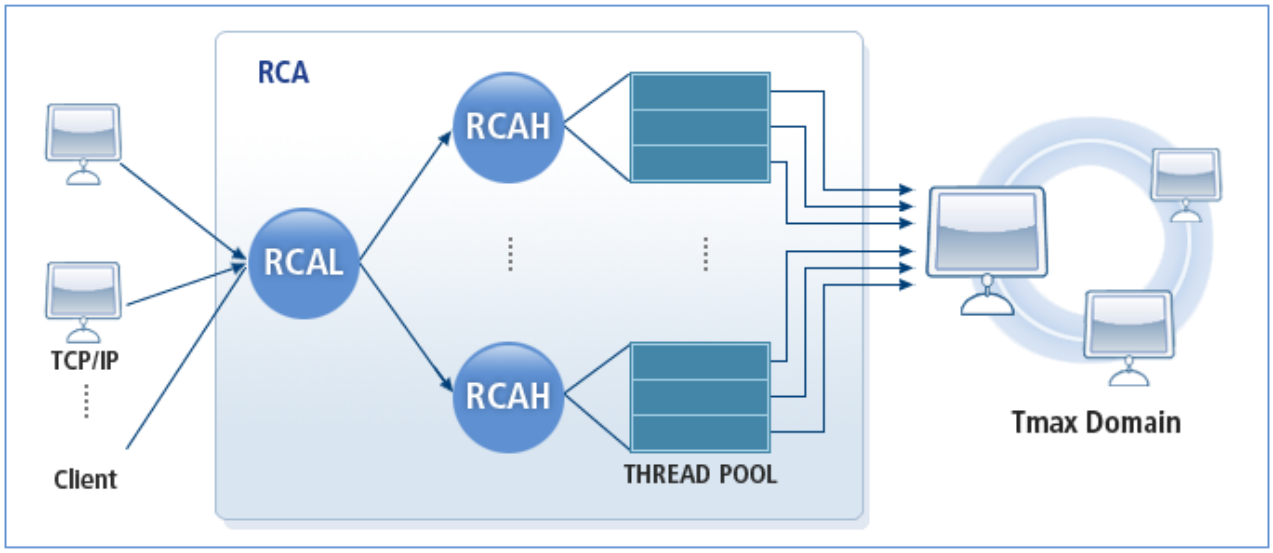
2-tier 시스템에서 3-tier 시스템으로 변환이 용이하도록 다양한 에이전트를 제공한다.

RCA(Raw Client Agent)

RCA는 Tmax 클라이언트 라이브러리를 사용할 수 없는 기존 통신 프로그램과 TCP/IP 소켓으로 연결하여 Tmax 시스템에서 제공하는 서비스를 이용할 수 있도록 지원하는 에이전트로 리모트(REMOTE) 모드 또는 로컬(LOCAL) 모드의 서비스를 지원한다. 기존 Tmax 클라이언트 라이브러리처럼 하나의 클라이언트 라이브러리는 하나의 클라이언트 프로그램에서만 사용한다. 그러나 RCA는 Multi Thread 방식으로 작성되어 하나의 Thread가 하나의 Tmax 클라이언트로 동작한다. 또한, 다양한 형태의 클라이언트를 지원하기 위해 최대 32개까지의 멀티 포트 지정이 가능하다.

RCA는 사용자의 접속을 제어하는 RCAL, 사용자의 로직과 함께 생성되는 RCAH를 통해 안정적인 장애대책이 가능하고 관리 툴인 rcastat와 rcakill을 제공한다.

다음은 RCA의 구조이다.



RCA 구조

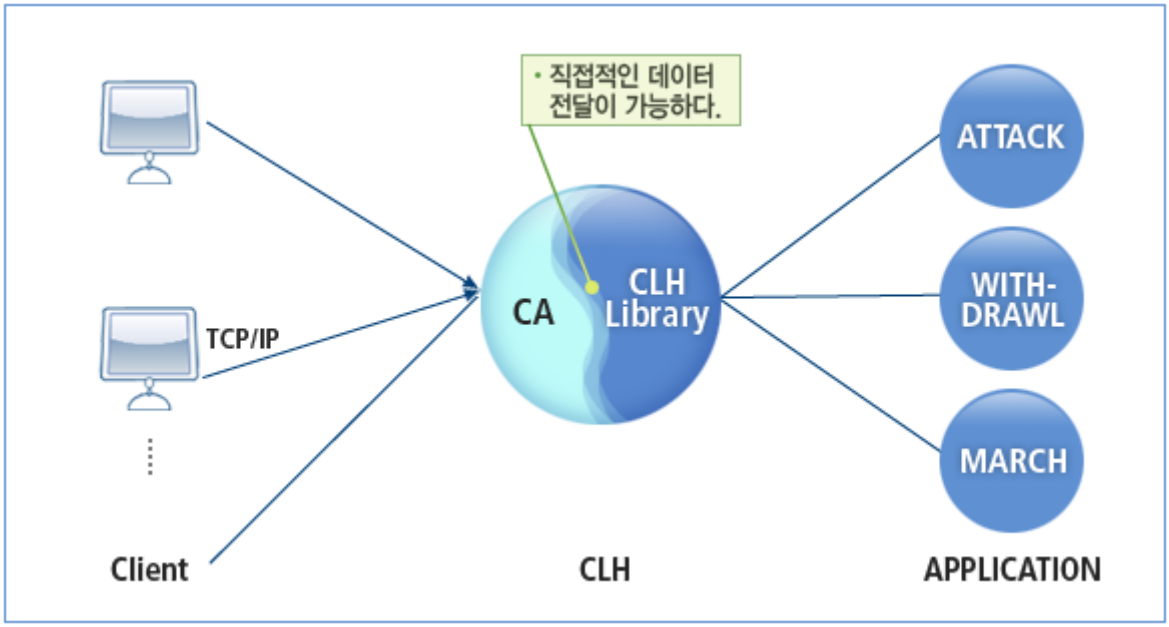
SCA(Simple Client Agent)

SCA는 Tmax 클라이언트 라이브러리를 사용할 수 없는 기존의 통신 프로그램과 TCP/IP 통신으로 연결하여 Tmax 시스템에서 제공하는 서비스를 이용할 수 있도록 지원하는 에이전트이다. SCA는 customizing routine과 CLH 라이브러리로 구성된다. CLH는 CLH 라이브러리와 링크된다.

클라이언트와 접속 및 해제, 데이터의 송수신에 해당하는 네트워크에서의 처리 부분은 시스템 내부적으로 처리된다. 개발자는 클라이언트와 미리 정의된 포트 번호를 Tmax 환경파일에 설정하여 클라이언트와 접속할 수 있으며, 수신된 데이터와 송신할 데이터를 개발자의 의도에 맞도록 수정 및 보완할 수 있다.

SCA는 다양한 형태의 클라이언트를 지원하기 위해 최대 8개까지 포트 지정이 가능하다. SCA 모듈은 CLH 모듈과 직접적인 데이터의 전달이 가능하고 Non-Tmax 클라이언트/Tmax 클라이언트를 동시에 수용할 수 있다. SCA의 서비스 방식은 TCP/IP raw 소켓을 이용하는 방법과 Tmax 클라이언트 라이브러리를 이용하는 방법으로 나뉜다.

다음은 SCA 중 하나인 CA(Client Agent)를 이용한 서비스 호출 형태를 보여주는 그림이다.



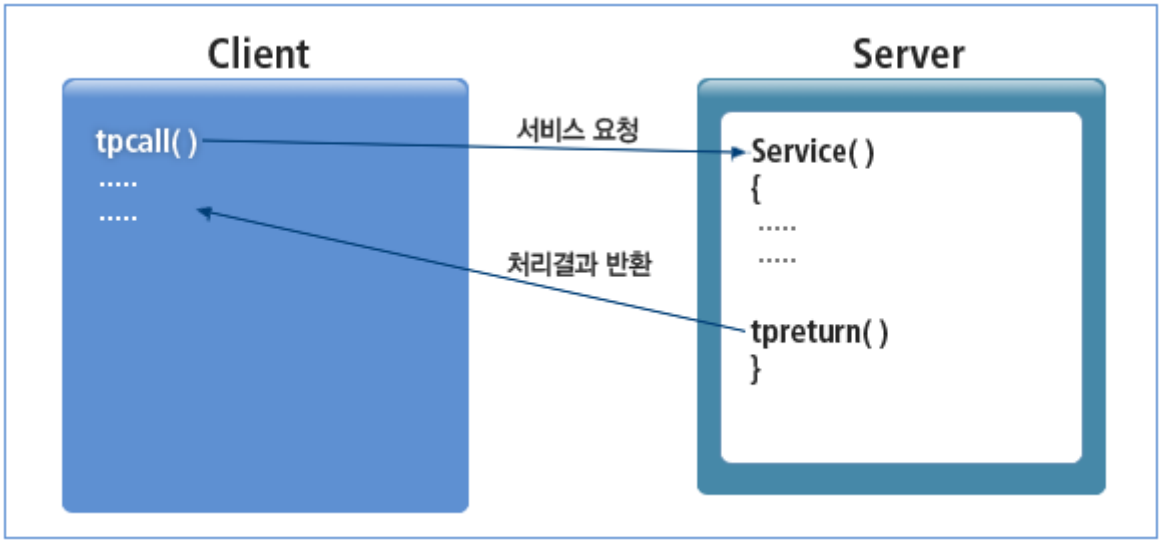
CA를 이용한 서비스 호출 형태

2.3.12. 다양한 통신 방식 지원

클라이언트에서 서버로 통신을 요청하는 다음과 같은 4가지 방식을 지원한다.

- 동기형 통신

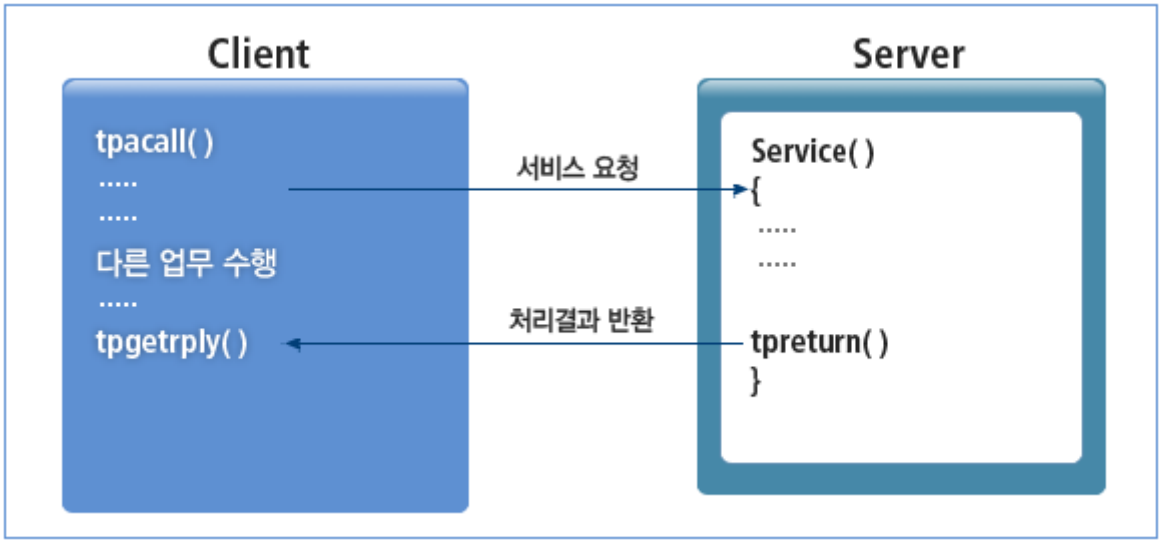
  클라이언트에서 서버로 요청을 보내고 서버에서 응답이 올 때까지 블로킹(blocking)되어 기다린다.



동기형 통신

- 비동기형 통신

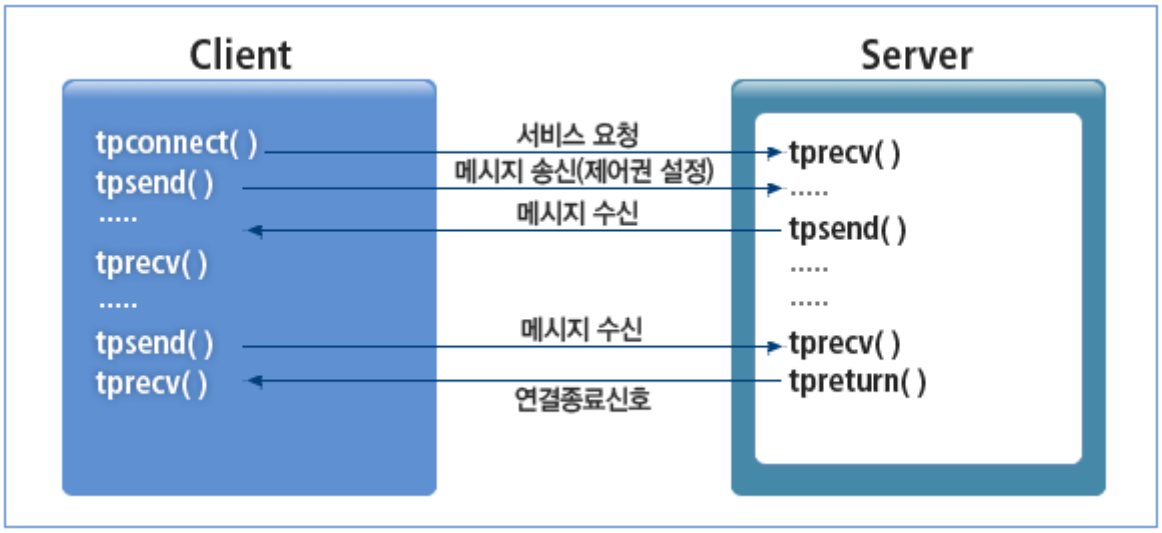
  클라이언트가 서버로 요청을 보낸 후 서버에서 응답이 올 때까지 기다리지 않고 다른 일을 처리할 수 있다.
  응답을 받고자 할 경우 함수를 사용해서 서버로부터 응답을 받는다.



비동기형 통신

- 대화형 통신

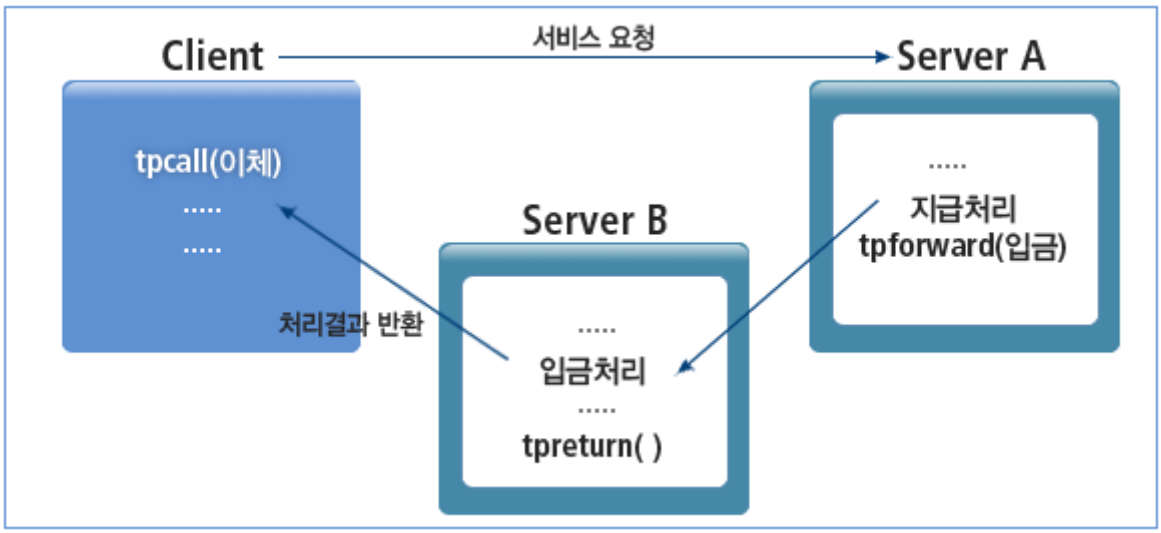
  클라이언트가 서버로 요청을 보내는 경우 서비스를 연결하고 서비스 요청을 보낸다. 메시지를 받을 때는 대화형 통신 함수를 이용해서 받는다. 클라이언트/서버 간 로컬 통신을 통해 control을 주고받으면서 메시지를 송수신한다. control을 가진 쪽이 메시지를 송신할 수 있다. 통신이 이루어졌을 때 connection descriptor가 반환되며, 반환된 connection descriptor가 메시지 전달을 확인하는 데 사용된다.



대화형 통신

- 전달형
  - 유형 A

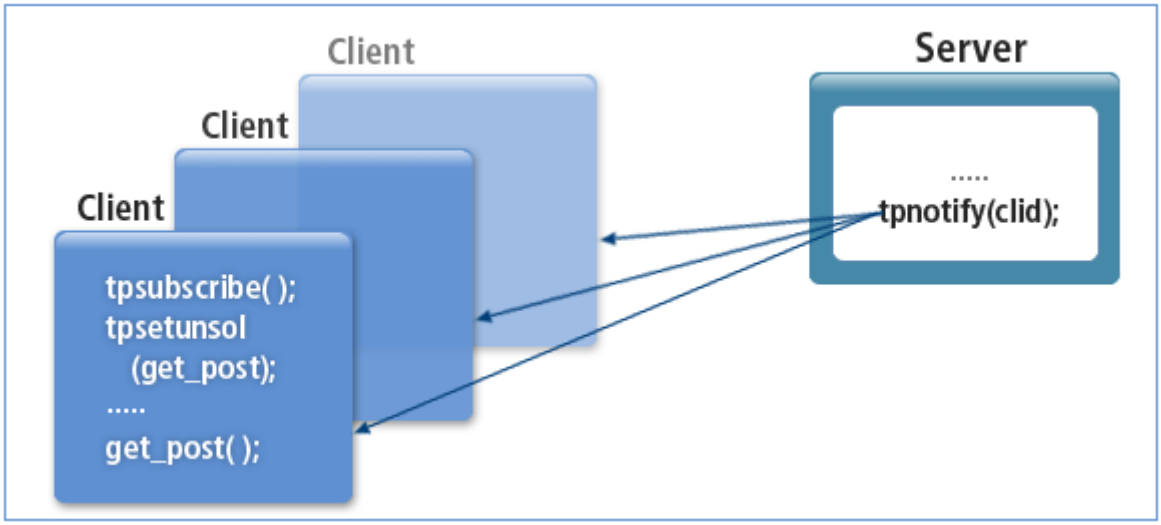
    비즈니스 로직을 모듈화하며 단계적인 서비스 처리가 가능한 유형으로 각 모듈에 대한 사용 효율을 높인다. 문제 분석 및 수정할 경우에 체계적인 접근을 가능하게 하고 동기형과 비동기형을 혼합하여 사용할 수 있다.



전달형-유형 A

◦ 유형 B

대외 기관과 같은 기존의 레거시 시스템과 연동하는 유형으로 서버 프로세스가 블로킹(blocking)되지 않고 지속적으로 전달되는 서비스 요청을 처리된다. 레거시 시스템에서 전달되는 응답을 최종적으로 호출자에게 전달할 수 있는 형태의 서비스를 지원한다.



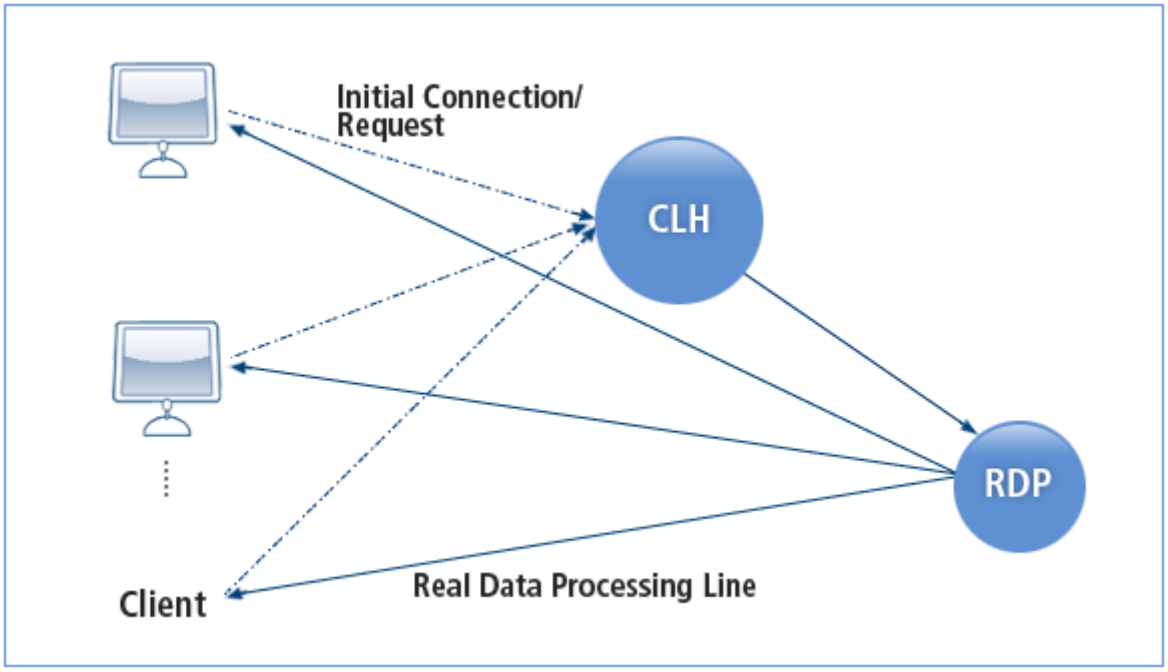
전달형-유형 B

2.3.13. 다양한 개발 방식 지원

RDP(Real Data Processor)

실시간으로 지속적으로 변하는 데이터를 보다 빠르게 처리하기 위해 CLH(Client Handler)를 거치지 않고 RDP에서 클라이언트에게 직접적으로 데이터를 전달할 수 있다.

따라서 RDP를 이용하게 되면 CLH의 부하를 크게 낮출 수 있기 때문에 Tmax 시스템 전체적인 성능을 높일 수 있다. RDP는 UDP 데이터 유형에 대해서만 지원하고 있다. RDP는 UCS 서버 프로세스 형태로 구성된다.



클라이언트와 RDP의 직접적인 데이터 전송

Window 제어

Window 기반의 클라이언트 프로그램에서 사용할 수 있는 편리한 라이브러리를 제공한다. WinTmax 라이브러리와 tmaxmt 라이브러리 2가지 종류가 있으며 기본적으로 Thread 기반으로 동작한다.

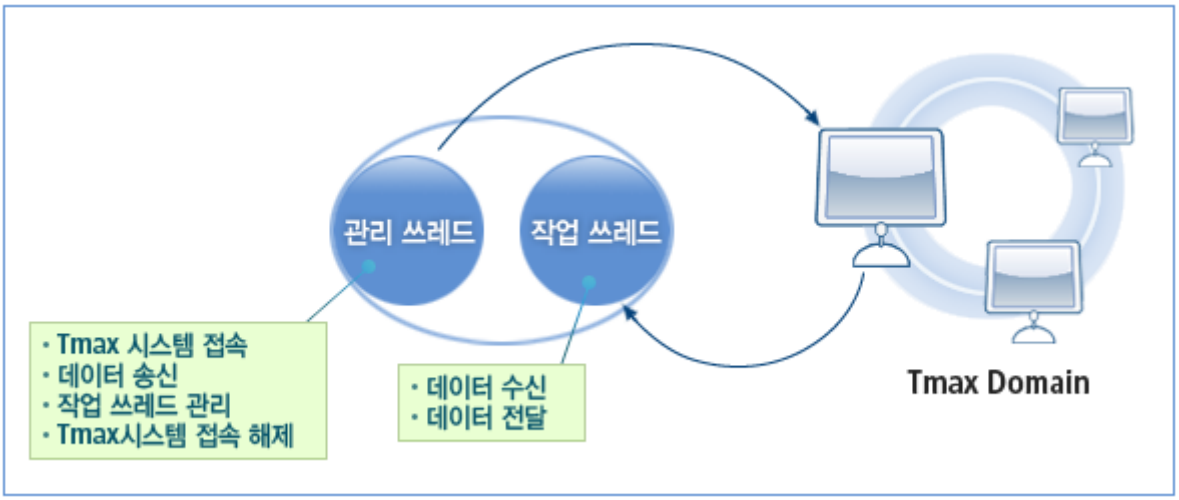
- WinTmax

  Window 기반의 클라이언트 라이브러리로 다중 Window를 설정할 수 있도록 설계되어 있다.

  WinTmax 라이브러리는 내부적으로 관리 Thread와 작업 Thread로 구성된다. 다중 Window 설정이 가능하고 256개까지의 Window 설정이 가능해서 동시 다발적으로 발생하는 데이터 처리에 유용하다.

구분	설명
관리 Thread	Tmax 시스템과의 접속, 데이터의 송신, 작업 Thread 관리, Tmax 시스템과의 접속 해제의 기능을 수행한다.
작업 Thread	데이터의 수신과 특정 Window에 데이터 전달의 기능을 수행한다.

  다음은 WinTmax 라이브러리 구조를 보여주는 그림이다.



WinTmax 라이브러리 구조 및 기능

- tmaxmt

  클라이언트 프로그램의 Thread화가 가능하도록 개발되어 있다. 개발자는 Thread를 사용하여 프로그램을 만들어야 하며 각각의 Thread에서 지정된 함수(WinTmaxAcall(), WinTmaxAcall2())를 사용하여 데이터를 송수신할 수 있다. 각각의 함수는 내부적으로 Thread를 생성하고, 서비스를 호출하며 그 결과를 받아 지정된 Window나 함수로 데이터를 전달한다.

  WinTmaxAcall은 지정된 Window에 데이터를 SendMessage를 사용하여 전달한다. 반면 WinTmaxAcall2는 지정된 Callback 함수에 데이터를 전달한다.

2.3.14. 안정적 메시지 전달

Tmax는 HMS(Hybrid Messaging System)를 통해서 안정적인 메시지 전달을 지원한다.

Tmax HMS는 다음과 같은 특징을 갖는다.

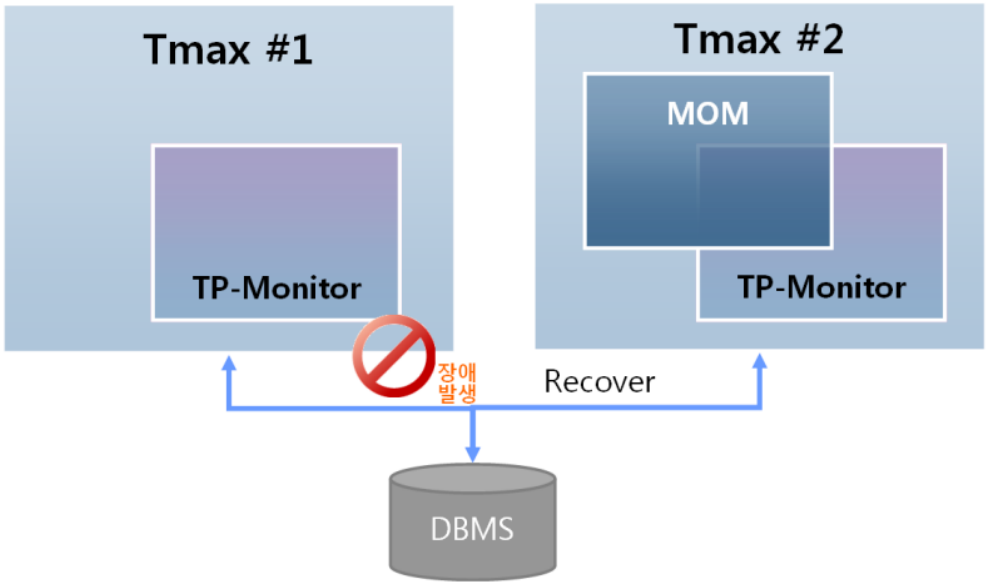
- Hybrid 아키텍처

  TP-Monitor(Tmax)와 MOM(Messaging Oriented Middleware)이 공존하고, MOM과 RM 사이의 2PC 기능을 제공한다. Tmax HMS는 Tmax의 TP-Monitor 기능에 추가하여 MOM 기능을 제공하기 위한 모듈로, 모듈 간 연계가 유연한 구조를 제공한다. HMS는 TMS와 TP-Monitor(Tmax 기본 기능)와 MOM이 공존하는 구조를 갖는다.

- 신뢰성 있는 장애복구

  Persistent 메시지의 저장을 위해 DBMS를 사용한다.

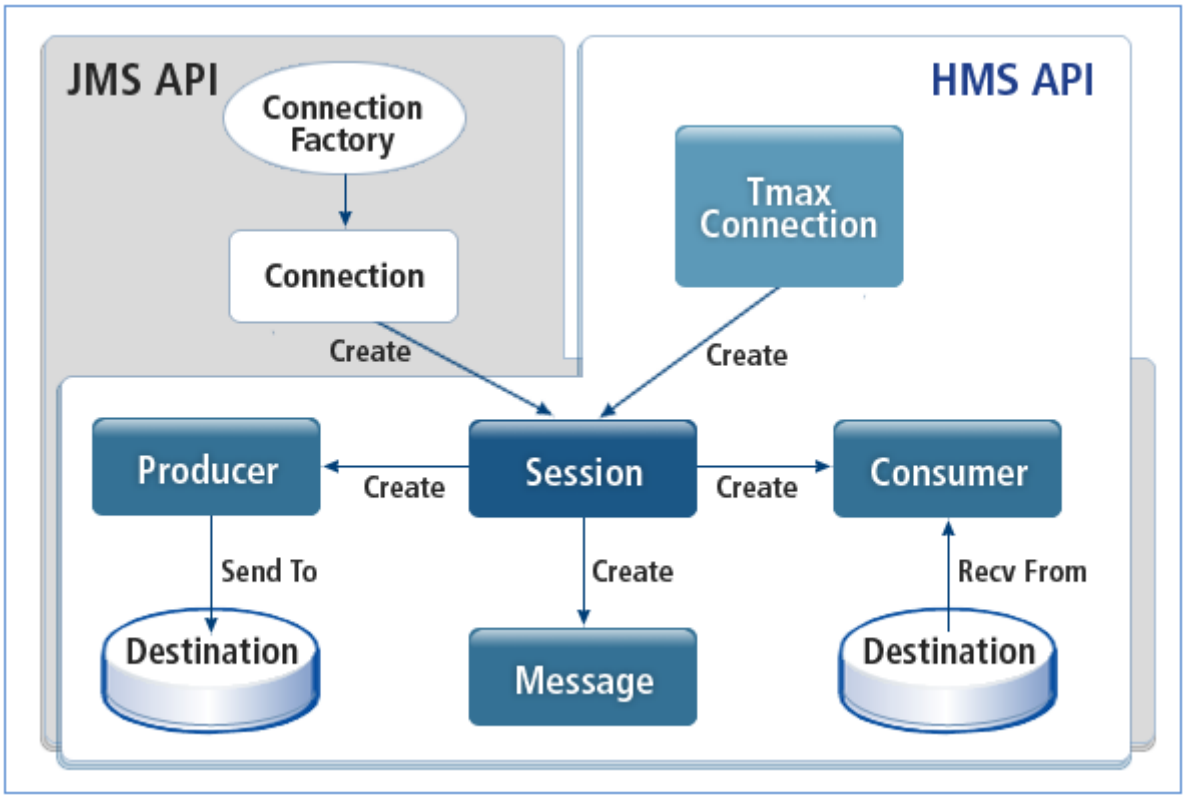
- 높은 가용성
  - 백업노드를 지정하여 장애에 유연하게 대처한다. Tmax HMS는 스토리지를 이용하여, 장애복구 및 신뢰성 있는 메시지 전달을 보장한다. 장애복구의 신뢰성 보장을 위해 DBMS를 사용하고 장애가 발생하면 Persistent 메시지를 DBMS로부터 복구한다.
  - Active-Standby의 HA(High Availability) 구성을 통해 시스템 장애에 대응한다.



HMS의 높은 가용성

- JMS 메시징 모델 수용

  JMS 메시징 모델을 수용하여 P2P, Publish/Subscribe를 지원하고 JMS 스펙과 유사한 C API 제공하여 사용하기 쉽다.

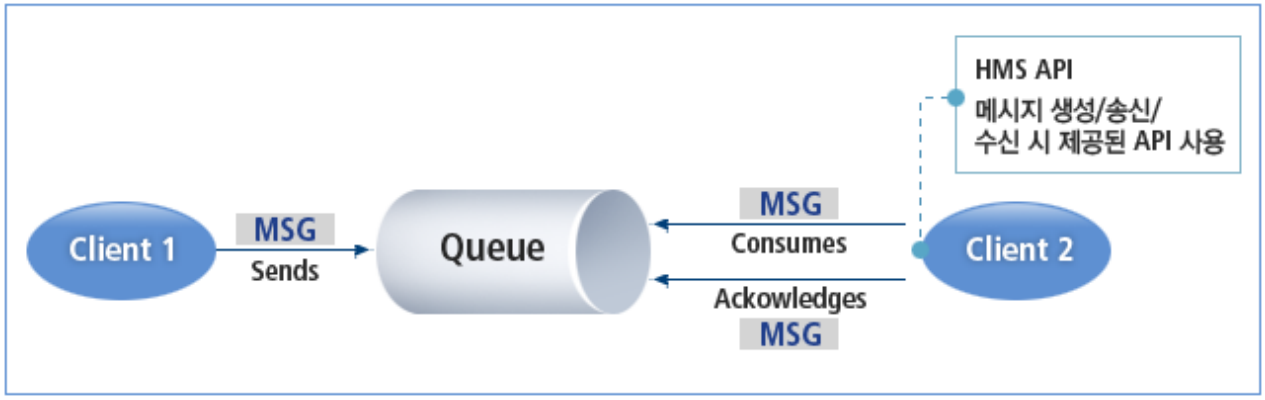


JMS의 API 호환성

HMS는 Tmax의 기능으로 Sender와 Receiver의 느슨한 결합(loosely coupled)을 위한 통신 매개체이며, Queue 방식과 Topic 방식을 지원한다.

- Queue 방식(Point-to-Point)

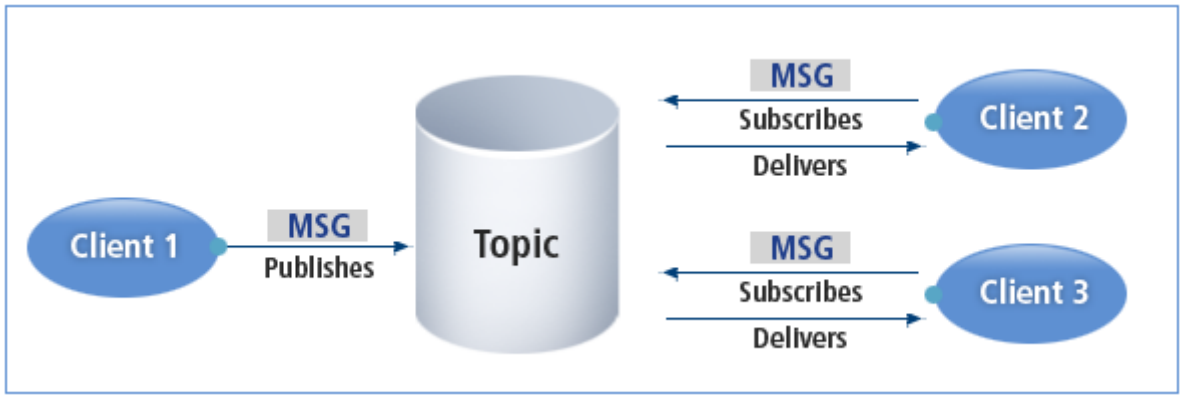
각 메시지는 하나의 소비자(consumer)에게만 전달되며, 송/수신 간의 시간 의존성(timing dependency)이 없다.



HMS-Queue 방식

- Topic 방식(Publish/Subscribe)

  각 메시지는 다수 소비자에게 전달될 수 있고, Durable subscription을 통한 메시지 전달로 신뢰성이 보장된다.



HMS-Topic 방식

HMS에 대한 자세한 내용은 "Tmax HMS User Guide"를 참고한다.

2.4. Tmax 특징

Tmax는 기존 Master/Slave 방식 대신 Peer-to-Peer 구조를 채택하였고 각 노드에 RACD(Remote Access Control Daemon)가 존재한다. 또한, Stream Pipe 통신 방식을 사용하여 Queue Full을 원천적으로 방지하고 네트워크 프로세스에서의 과다한 트랜잭션을 차단함으로써, 메시지 큐 방식보다 프로세스 간의 통신을 안정적으로 유지시킬 수 있다. Tmax의 이러한 특장점은 메모리 자원의 낭비를 최소화하고 빠른 장애 복구를 지원하며, 관리자가 능동적으로 장애에 대응할 수 있게 하여 탁월한 안정성을 제공한다.

다음은 Tmax의 특징이다.

- X/Open, ISO-TP 등 각종 DTP 국제표준 100% 준수
  - 분산 트랜잭션 프로세싱의 국제 표준인 X/Open DTP(Distributed Transaction Processing) 모델을 준수하므로 응용프로그램(AP), 트랜잭션 관리자(TM), 자원 관리자(RM), 통신 자원 관리자(CRM) 등을

기반으로 호환성 있는 API와 시스템 구조를 명시한다.

- 국제 표준 기구 OSI(Open Systems Interconnection group)의 DTP 서비스에 대한 기능적 분산과 기능 구성 요소 간 API, 시스템 인터페이스에 정의되어 분산 트랜잭션을 지원하기 위한 기능 그리고 분리된 개방형 시스템에 존재하는 다수 트랜잭션을 상호 조정이 가능하도록 기본 틀을 명시한다.
- 분산 환경에서 이기종 간의 투명한 업무 처리 및 OLTP(On-Line Transaction Processing)를 지원한다.

- Protection, Multiplexing의 효율성 향상

  Stream Pipe 방식의 IPC(Inter Process Communication) 구현으로 Protection, Multiplexing의 효율성을 향상시킨다.

- 다양한 메시지 타입 및 통신 유형 제공
  - Integer, Long, Character 등의 메시지 타입을 지원한다.
  - 동기형(Synchronous) 통신, 비동기형(Asynchronous Mode) 통신, 대화형(Conversational) 통신, 전달형 등의 통신 유형을 제공한다.
  - FDL(Field Definition Language) 및 Structure Array를 지원한다.
- 장애대책(Fault Tolerance, Fail-Over)
  - Peer-to-Peer 방식의 TP-Monitor 구조를 갖는다.
  - H/W 및 S/W에 대한 장애 대책을 갖는다.
  - 다양한 장애 방지 기능을 지원한다.
- 확장성(Scalability)
  - 클라이언트들의 증가에도 무리 없는 시스템 활용성을 제공한다.
  - CA(Client Agent)를 이용한 2-tier 모델에서 3-tier 모델로의 효율적인 전환이 가능하다.
  - 레거시 시스템에 대한 다양한 프로토콜을 제공한다.
  - WebT를 이용한 웹 환경으로 서비스를 확대하여 제공한다.
  - TCP/IP, SNA, X.25 등 레거시 시스템에 대한 다양한 프로토콜을 제공한다.
  - 다양한 프로세스 제어 방식을 지원한다.
- 유연성(Flexibility)
  - 다양한 프로세스 제어 방식을 지원한다.
  - 사용 환경별 특성을 고려한 기능이 요구될 경우 추가 기능을 지원한다.
- 뛰어난 성능
  - 시스템 자원의 효율적 활용이 가능하다.
  - 단순, 명료한 개발 생산성 향상을 위한 API를 제공한다.
  - 사용자 입장의 편리하고 사용하기 쉬운 시스템 관리(System Monitoring) 기능을 제공한다.
- 다양한 H/W 플랫폼 지원
  - IBM OS 390, 대부분의 UNIX 계열 운영체제, Linux, Windows NT 등의 플랫폼을 지원한다.
- PowerBuilder, Delphi, Visual C/C++, Visual Basic, .NET(C#, VB) 등의 모든 4GL 지원

2.5. Tmax 도입 시 고려 사항

2.5.1. 시스템 환경

다음은 Tmax 도입 시 고려해야 할 시스템 환경에 대한 설명이다.

- **지원환경**

구분	설명
프로토콜	Application API : XATMI, TX
	Integrating API : XA
	Network : TCP/IP, X.25, SNA(LU 0/6.2)
OS	서버 : All UNIX, NT, Linux
	클라이언트 : All UNIX, Windows, MS-DOS, etc.
지원 플랫폼	IBM OS 390, UNIX, Linux 및 NT를 지원하는 모든 H/W
서버용 개발언어	C, C++, COBOL
클라이언트용 개발언어	C, C++ 및 다양한 4GL(Power Builder, Delphi, Visual C/C++, Visual Basic, .NET(C#, VB) etc.)에 대한 인터페이스 지원
DBMS 지원	Oracle, Informix, Sybase and DB2(UDB)

- **서버 요구사항**

구분	설명
H/W	Memory: 0.50MB
	Disk(Tmax 클라이언트) : 0.277MB (Include - 83KB, DLL - 86KB, Type Compiler - 108KB)
S/W	IBM OS 390, UNIX, Linux, NT
	C 또는 C++, COBOL Compiler
네트워크 프로토콜	TCP/IP

- **클라이언트 요구사항**

구분	설명
H/W	Memory : 0.537MB + 0.2 ~ 0.5MB / application
	Disk(Tmax 클라이언트) : 0.277MB (Include - 83K,B DLL - 86KB, Type Compiler - 108KB) 단, Power Builder의 경우 203K(DLL, PBD) 추가

구분	설명
S/W	Linux, NT, Windows (2000, XP), MS-DOS, UNIX
	Power Builder, Delphi, Visual Basic, Visual C++, C, .NET(C#, VB)
네트워크 프로토콜	TCP/IP

2.5.2. 고려사항

Tmax를 도입할 때 기능, 성능 및 안정성 등 여러 측면에서 고려해야 할 사항이 있다.

- **기능 면에서 고려해야 할 사항**

고려사항	세부 사항
TP-Monitor의 기본 기능	• 프로세스 관리 • 분산 트랜잭션 지원 • 부하 분산 • 클라이언트/서버 간의 다양한 통신 • 장애 대책(모든 서버장애 대응 및 방지 기능) • 이기종 DBMS 지원
부가 기능	• 보안 기능 • 시스템 관리 • Naming 서비스 • BP 클라이언트의 Multiplexing 기능 • 시스템 특성에 맞는 보안기능 • Structure array 통신 지원 • 호스트와 연계될 때 대처 능력

- **기능 외에 고려해야 할 사항**

항목	고려사항
성능	• 평균 처리시간 또는 시간당 최대 처리 건수 • 자원 사용도
안정성(장애 대책)	• 고객사의 장애 발생 빈도 • 장애가 발생되는 경우 대응 능력 및 걸리는 시간
교육 및 기술지원	• 엔지니어 기술 수준 • 교육 및 컨설팅 지원(시스템 설계 단계의 컨설팅, 애플리케이션 개발 전문교육(OS, 네트워크, TP-Monitor))

항목	고려사항
리스크 관리	• 신시스템을 구축하는 경우 시행착오 최소화 • 신기술을 이용한 시스템 구축 • 구축 대상 업무 이해도
개발 및 운영 편리성	• 클라이언트/서버 개발 툴 지원 • 개발용 드라이버 제공 • 시스템 통계자료 모니터링 • TP-Monitor 환경 동적 변경 • 시스템 운영의 편리성 • 제공되는 보고서 기능
사용자 만족도	• 기능에 대한 만족도 • 기술지원 및 교육에 대한 만족도 • 제품의 우연성에 대한 만족도
호환성	• 국제 표준 준수 여부(X/Open DTP, OSI-TP) • 특정 H/W 및 DBMS와의 독립성
업체 규모	• 자본금 규모 • 종업원수 • 매출액 • 향후 성장성
기타	• 기술 이전 • 버전업(Version up) 계획

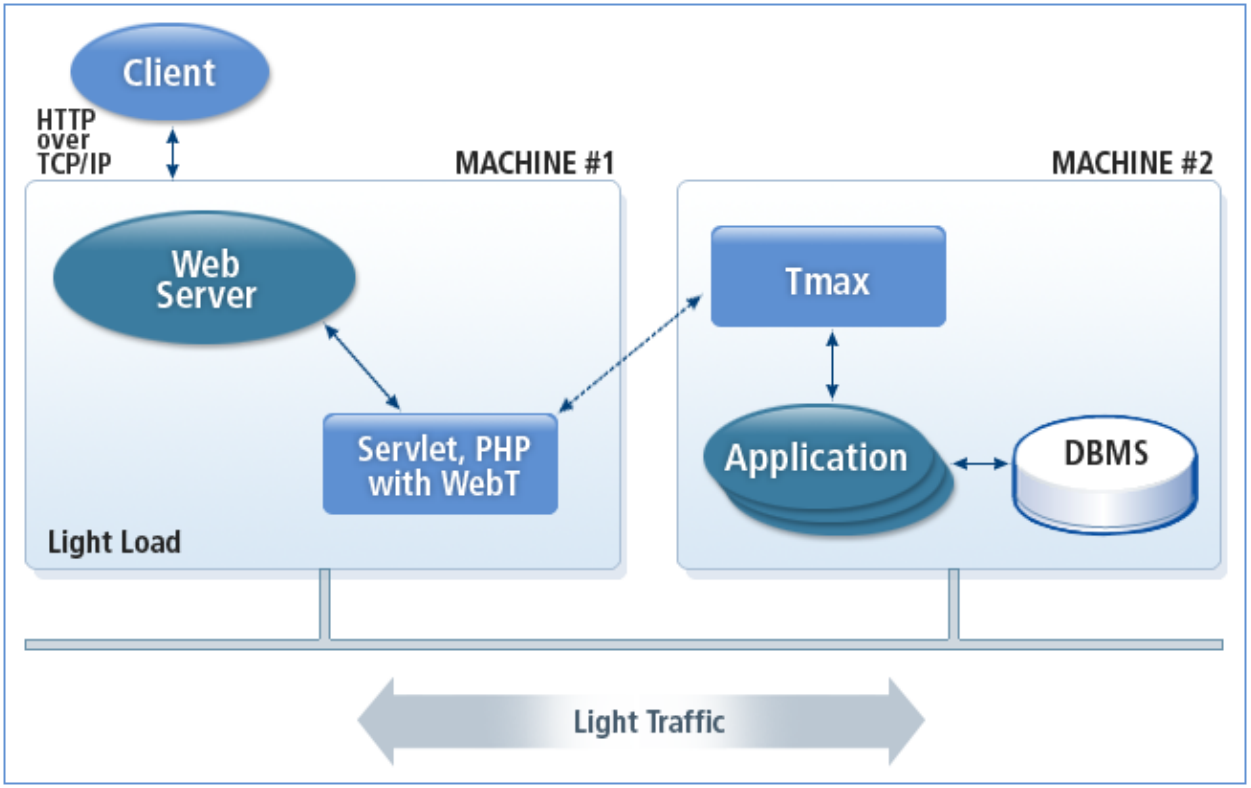
3. WebT 소개

본 장은 Tmax와 Java 애플리케이션 프로그램과의 트랜잭션 서비스를 지원하기위해 제공되는 WebT에 대한 기본 기능과 역할에 대해서 설명한다.

3.1. 개요

WebT(Web Transaction)는 클라이언트/서버 환경의 미들웨어 제품인 Tmax 서버와 Java 애플리케이션 프로그램 사이의 트랜잭션 서비스를 지원하는 프로그램으로 API 라이브러리 형태로 배포된다. WebT는 JEUS를 비롯한 웹 기반 환경의 WAS(Web Application Server) 제품에서 활용될 수 있다. Tmax의 트랜잭션 처리와 부하 조절 기능을 이용해 웹 환경에서 동적 데이터 서비스를 제공할 수 있도록 설계되었다.

WebT와 Tmax 서버 사이에서 서비스가 진행되는 과정은 다음과 같다. 클라이언트에서 요청을 받으면 WebT 모듈을 통해서 Tmax의 서버 프로그램이 실행되며 서비스가 진행된다.



WebT와 Tmax 간의 서비스의 흐름

3.2. WebTConnectionPool

WebT는 Tmax의 연결을 효율적으로 관리하기 위해 WebTConnectionPool 클래스를 제공한다. WebTConnectionPool은 Tmax 서비스를 요청할 때마다 매번 연결 객체를 새로 생성하지 않고 이전에 사용한 객체를 재사용한다. 연결 객체의 재사용을 통해 Tmax 서버에 네트워크 연결을 설정하고 종료하는 데 소비되는 자원 및 시간이 절약된다.

WebTConnectionPool은 1개 이상의 WebTConnectionGroup으로 구성되며, 각 WebTConnectionGroup은 1개의 Tmax 서버와 연결된다. 클라이언트 프로그램에서는 WebTConnectionGroup의 그룹명을 이용하여 원하는

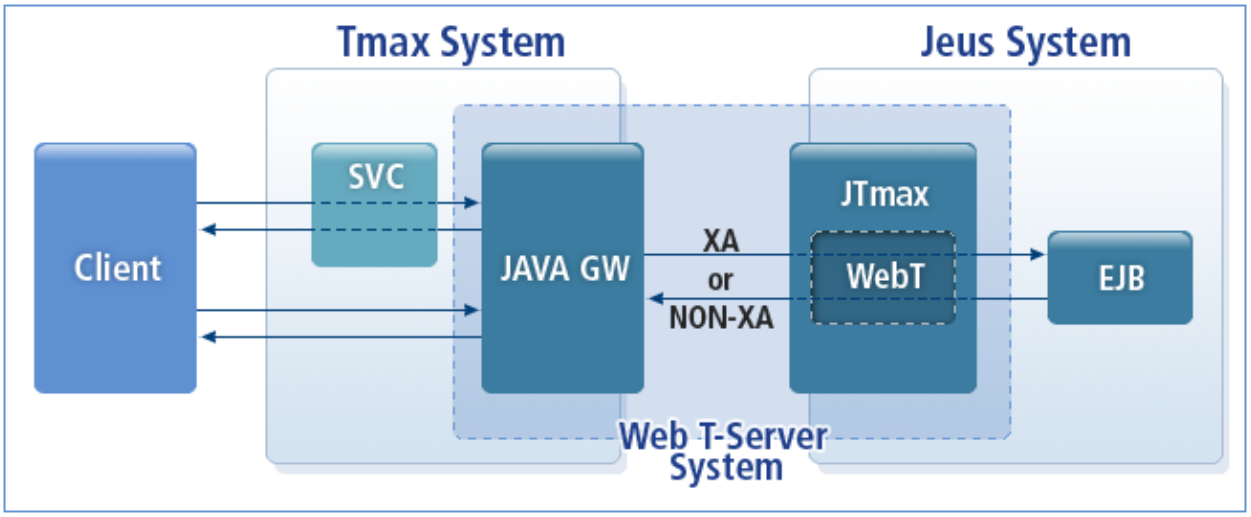
Tmax 서버에 연결할 수 있다.

WebT 모듈을 JEUS와 연동한 경우 JEUS 컨테이너에서 WebTConnectionPool을 관리한다. JEUS Container는 WebT.properties 환경설정 파일을 이용하거나 JEUS의 환경설정 파일을 이용하여 Connection Pool을 생성한다. 또한 사용자가 getConnection을 실행한 뒤 반환되지 않은 커넥션을 자동으로 Connection Pool에 반환한다.

3.3. WebT-Server 시스템

WebT-Server 시스템은 Tmax 시스템과 JEUS 사이에 존재하며, Tmax 클라이언트가 JEUS의 EJB 서비스를 호출할 수 있게 해준다.

다음 그림은 WebT-Server 시스템을 통해 EJB 서비스가 호출되는 과정이다.



WebT-Server 시스템

WebT-Server 시스템은 다음의 모듈로 구성된다.

- JAVA GW

  Tmax가 JEUS로 보내는 서비스 요청을 처리한다.

- JTmax

  JEUS에서 Tmax의 서비스 요청을 수신하는 데몬이다.

- WebT Library

  Tmax와 JEUS가 주고받는 데이터를 처리한다.

- 기타 Utility

  webtutil.jar는 jeus.jar와 jeusutil.jar에서 WebT를 사용하기 위한 클래스를 별도로 패키징한 파일로 JEUS가 없는 곳에서 WebT를 기동하기 위한 라이브러리이다.

4. Tmax 애플리케이션

본 장은 Tmax 애플리케이션을 개발하기 위해 숙지해야 할 기본 개념과 클라이언트/서버 프로그램의 흐름과 구성과 프로그램 개발을 위한 AP와 에러에 대해서 설명한다.

4.1. 애플리케이션 구성

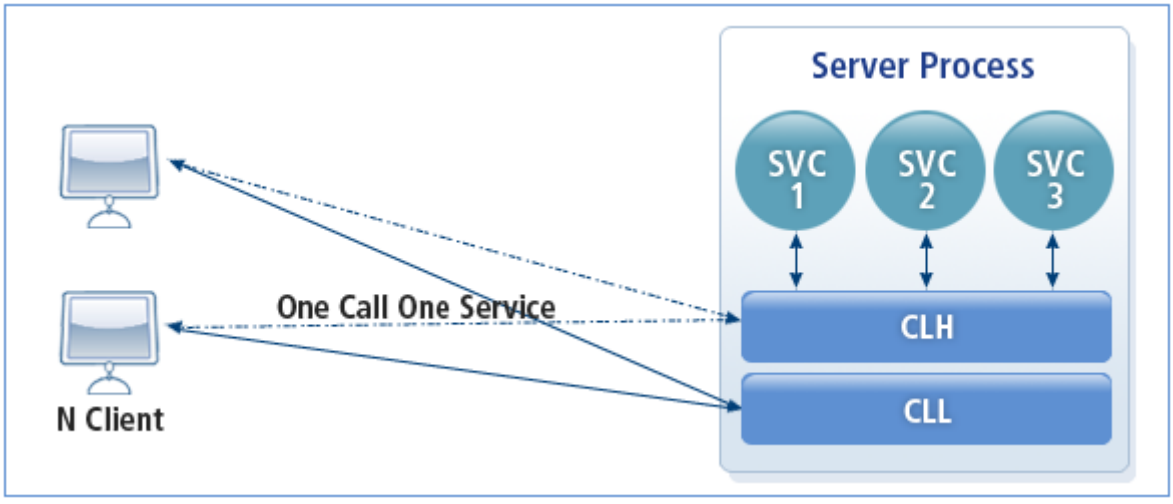
Tmax 애플리케이션은 클라이언트 프로그램과 서버 프로그램으로 구성된다. 클라이언트용 프로그램은 사용자 인터페이스를 담당(Presentation logic)하고 서버용 서비스 루틴은 업무 처리 및 데이터베이스 Access Logic을 구현한다.

클라이언트에서 요청이 발생하면 서버에서 해당 요청을 처리한다. 클라이언트가 N개 있는 경우도 하나의 호출에 대해서는 하나의 서비스가 기동한다. 클라이언트 요청은 CLL에 먼저 접속을 하고 요청에 대해 tpcall이 발생하고 CLL은 해당 서비스에 해당하는 요청을 실행하고 실행이 완료되면 요청을 반환하는 역할을 한다. 클라이언트 프로그램은 Tmax에 연결할 때 Tmax의 호스트 주소와 Port 등 연결을 위한 사전 정보 설정이 필요하다. 이 정보는 .profile 또는 tmax.env 파일에 저장되며, TMAX_HOST_ADDR, TMAX_HOST_PORT 항목에 설정된다.

서버 프로그램은 Tmax에서 제공하는 main()과 개발자가 개발한 서비스 루틴으로 구성된다.

Tmax는 클라이언트와 서버의 연결 데이터 요청 등의 처리를 위해서 다양한 API 함수를 제공한다. Tmax API는 분산 처리 국제 표준(X/Open DTP model)을 준수한다. 애플리케이션을 개발할 때 서비스 루틴만 개발한다.

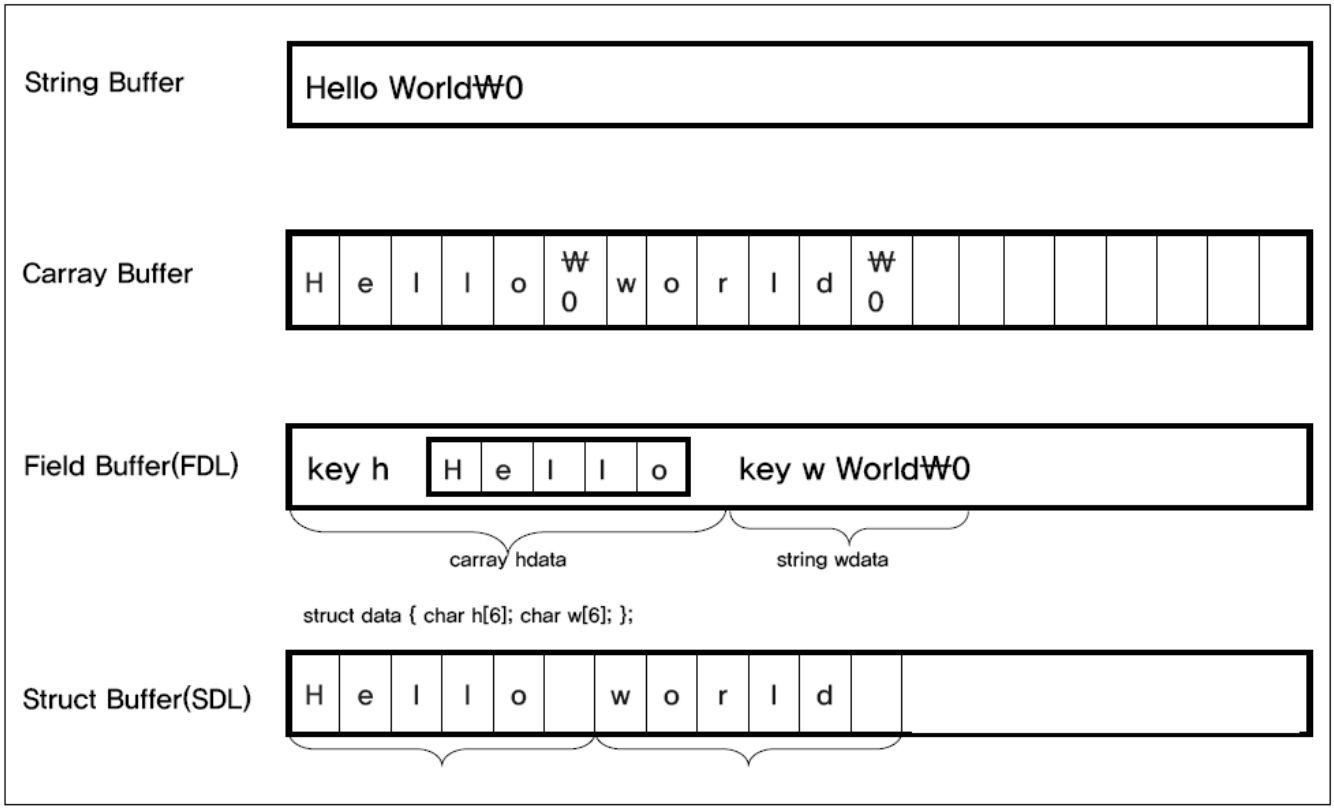
다음은 Tmax 애플리케이션의 구성을 보여주는 그림이다.



Tmax 애플리케이션 구성

4.2. 버퍼 유형

클라이언트에서 서버에 서비스 요청하는 경우 통신용으로 다음의 통신 버퍼를 사용한다.



Tmax 통신 버퍼 유형

- STRING 버퍼

  NULL 값으로 끝나는 문자열로 따로 버퍼의 길이를 명시할 필요가 없다. 플랫폼의 차이로 인한 문제는 발생하지 않는다.

- CARRAY, X_OCTET 버퍼

  길이가 지정된 Byte 열로 구성된 버퍼로 보통 바이너리 타입의 데이터를 보낼 때 사용되며 데이터를 교환하는 경우 반드시 길이를 명시해야 한다. 플랫폼의 차이로 인한 문제는 발생하지 않는다.

- STRUCT, X_C_TYPE 버퍼

  C 언어의 구조체를 데이터 통신에 사용하는 경우 사용한다. 구조체의 멤버로는 모든 원시 타입을 사용할 수 있으며 선언된 구조체 자체도 멤버로 사용할 수 있다. 또한, 구조체의 배열도 사용할 수 있다.

- X_COMMON 버퍼

  멤버 타입이 char, int, long으로 한정된 C 구조체이다.

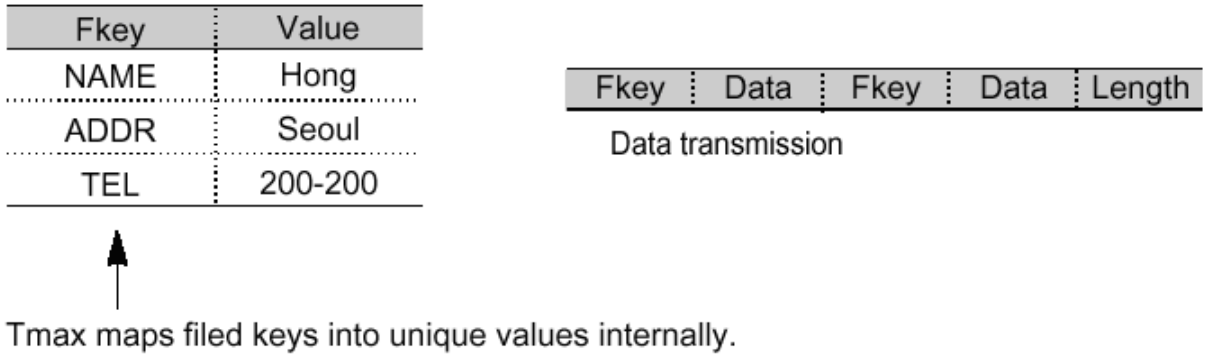
- FIELD 버퍼

  필드키와 데이터 값을 한 쌍으로 관리하는 데이터 버퍼로 모든 원시 타입의 데이터를 담을 수 있다. 교환되는 데이터형이 유동적일 경우 사용하며 다양한 방법의 데이터 접근 및 변환 API를 제공한다.

FDL(Field Definition Language)

일반적인 구조체와는 달리 필드키 버퍼(Field Key Buffer)를 사용하여 원하는 정보의 데이터만을 조작하여 처리할 수 있다.

각 필드키에 대한 이름(NAME, ADDR, TEL 등), 번호, 타입을 필드 버퍼 파일<XXX.f>에 기술한다. FDLC를 이용하여 <XXX.f>를 컴파일하면 <XXX_fdl.h>로 매핑된 파일을 생성시킨다. 컴파일될 때 include된 <XXX_fdl.h>를 참조하여 사용자가 선택한 필드키와 값만을 조작할 수 있다.



FDL 저장 방식

4.3. 클라이언트/서버 프로그램

클라이언트용 프로그램은 사용자 인터페이스를 담당(Presentation logic)하고, 서버용 서비스 루틴은 업무 처리 및 데이터베이스 access 로직을 구현한다.

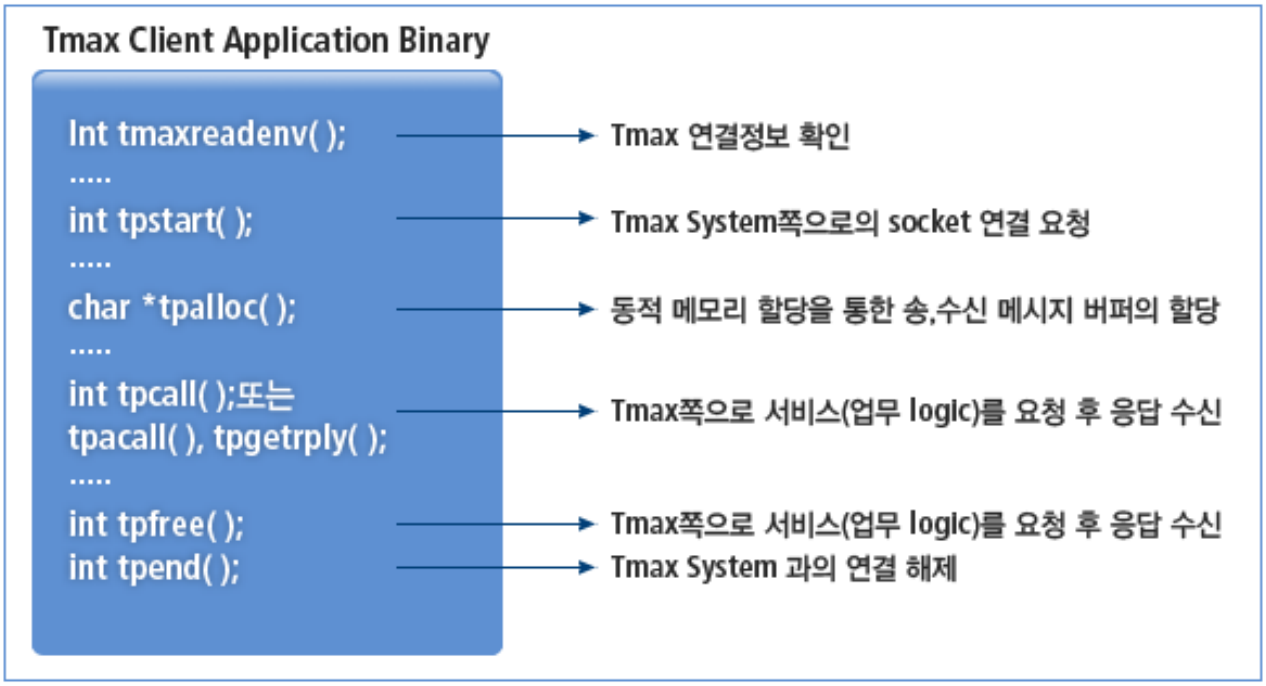
4.3.1. 클라이언트 프로그램

클라이언트 프로그램은 사용자의 입력(input)을 받아들여서 서버에게 서비스를 요청하고, 서버로부터 응답을 받아서 사용자에게 서비스 응답을 출력(output)하는 프로그램이다.

클라이언트 프로그램 흐름

다음은 클라이언트 프로그램의 흐름이다.

```
main()
{
     Tmax 접속
     송수신 메시지 버퍼 할당
     클라이언트 업무 로직 작성 (사용자 요구 사항을 입력받아서 송신 메시지 버퍼에 저장)
     서비스 요청 및 응답 (송신 메시지 버퍼를 Tmax CLH로 보냄,
               Tmax CLH를 통하여 수신 메시지 버퍼에 응답 데이터를 수신)
               클라이언트 업무 로직 작성 (응답 데이터를 사용자에게 보여줌)
     송수신 메시지 버퍼 해제
     Tmax 연결 해제
}
```



클라이언트 프로그램 흐름

다음은 클라이언트 프로그램의 주요 함수에 대한 설명이다.

- Tmax 통신 환경 함수

함수	설명
tmaxreadenv()	Tmax 클라이언트 프로그램을 실행할 때 기본적으로 참조하게 되는 일련의 Tmax 환경변수들을 특정 파일에 기술할 수 있다. tmaxreadenv() API를 호출하는 프로그램을 작성할 때 특정 파일의 경로명 및 파일명을 설정하게 되면, 해당 클라이언트 프로그램을 실행할 때 설정된 파일의 파일 포인터를 획득한 후 내용을 파싱한다. 파싱한 내용 중 TMAX_HOST_ADDR, TMAX_HOST_PORT 등의 Tmax 환경변수는 클라이언트 프로그램이 실행될 때 확보된 메모리 공간 내에 로드하고 관련 API 함수 호출에 이용한다.

- 클라이언트와 서버 연결 함수

함수	설명
tpstart()	CLL 프로세스로 소켓 연결을 요청하고 accept된 연결을 CLH로 전달한다. tpstart() 호출할 때 TMAX_HOST_ADDR, TMAX_HOST_PORT에 설정된 값을 사용한다.
tpend()	CLL 프로세스로 소켓 연결을 종료한다.

- 통신 버퍼와 해제 함수

함수	설명
tpalloc()	Tmax를 통한 클라이언트와 서버 사이의 데이터 송수신할 때 사용하는 메모리를 동적으로 할당한다. 데이터를 송수신할 때 필요한 길이 정보를 예측하여 그 크기만큼 미리 할당해야 한다. 동적 할당된 메모리는 반드시 명시적으로 해제해야 한다.
tpfree()	tpalloc()을 통해 동적 할당 메모리를 명시적으로 해제(반환)한다. 할당된 메모리 버퍼는 반환해야 하며, 그렇지 않으면 Gabage(memory leakage)가 발생한다.

- 클라이언트 동기 통신 함수

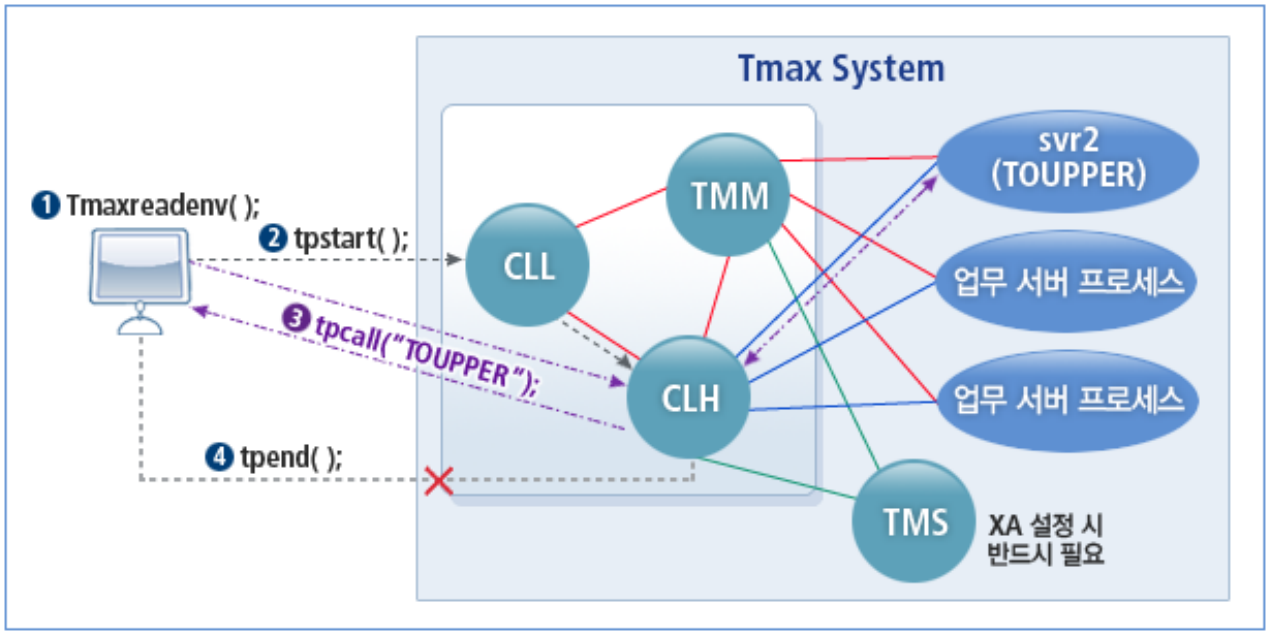
함수	설명
tpcall()	CLH로 서비스를 요청하고 송신 데이터를 전달하고 CLH는 클라이언트의 요청 서비스를 확인 후 서버 애플리케이션 프로세스로 전달한다. 클라이언트는 서비스 요청에 대한 응답이 도착할 때까지 대기한다.

- 클라이언트 비동기 통신 함수

함수	설명
tpacall()	CLH로 서비스를 요청하고 송신 데이터를 전달하고 CLH는 클라이언트의 요청 서비스를 확인 후 서버 애플리케이션 프로세스로 전달한다. 클라이언트는 서비스 요청에 대한 응답 여부와는 관계없이 tpacall() 이후 로직을 수행한다.
tpgetrply()	tpacall()에 대한 파라미터(cd) 값을 통해서 CLH로 응답 데이터를 요청한다. CLH에 이미 해당 클라이언트로 보낼 응답 데이터가 있으면 이를 클라이언트로 즉시 전달한다. 파라미터 중 flags 설정 값에 따라 응답 데이터가 올 때까지 기다리거나 혹은 기다리지 않고 즉시 응답 수신 에러를 클라이언트로 전달한다.

각 함수에 대한 자세한 내용과 사용법은 "Tmax Application Development Guide" 및 "Tmax Reference Guide"를 참고한다.

다음은 클라이언트 프로그램의 주요 함수 프로세스에 대한 그림이다.



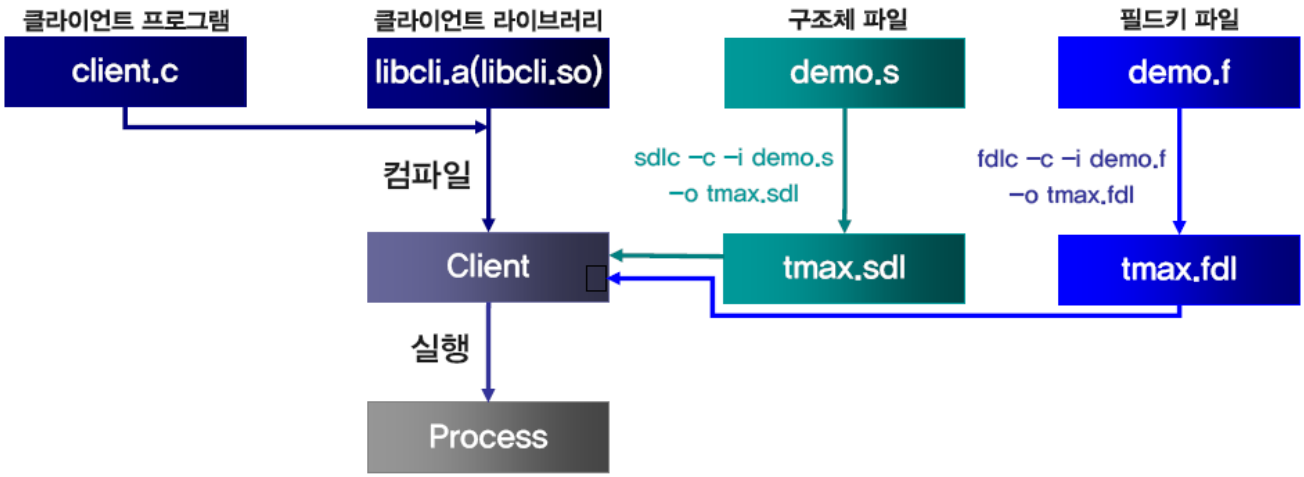
클라이언트 프로그램 함수 프로세스

클라이언트 프로그램 구성

클라이언트 프로그램의 코딩이 완료되면 컴파일하여 실행 파일을 생성한다.

클라이언트 프로그램을 컴파일하기 위해서는 개발자가 작성한 클라이언트 프로그램, Tmax 클라이언트 라이브러리, 구조체 버퍼를 사용하는 경우에는 구조체 파일, 그리고 필드 테이블이 정의된 필드키 버퍼 파일이 정의되어 있어야 한다.

다음은 클라이언트 프로그램의 구성에 대한 그림이다.



Tmax 클라이언트 프로그램 구성

- 클라이언트 프로그램

  개발자가 작성한 클라이언트 프로그램이다.

- Tmax 클라이언트 라이브러리(libcli.a / libcli.so)

Tmax가 제공하는 라이브러리로 클라이언트 프로그램 개발할 때 사용하는 함수들의 오브젝트 코드이다.

- 구조체 파일

  클라이언트 프로그램에서 구조체(STRUCT, X_C_TYPE, X_COMMON)를 사용하였다면 <파일명.s>로 끝나는 구조체 파일이 필요하다. 구조체 파일을 **sdlc** 명령어를 이용하여 미리 컴파일한다. 컴파일하면 구조체의 각 데이터가 표준 통신형으로 변환되는 데 필요한 정보를 담은 이진(binary) 형태의 파일이 생성된다. 이는 클라이언트 프로그램을 실행할 때 표준 통신형으로 데이터를 송수신하는 데 사용된다.

- 필드키 파일

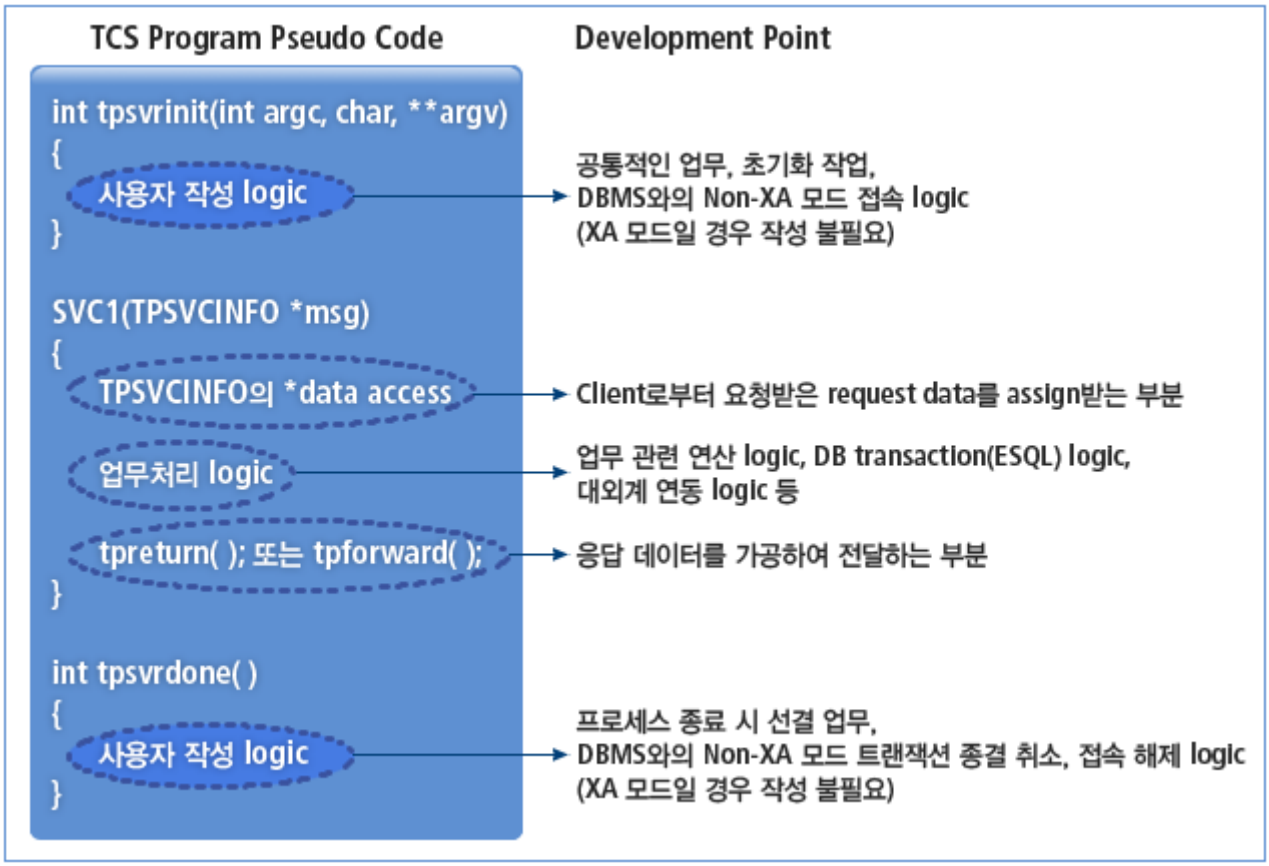
  필드키 구조를 사용하였다면 <.f>로 끝나는 필드 정의 파일이 필요하다. 파일을 **fdlc** 명령어를 이용하여 컴파일하면, 이때 필드키 버퍼 파일은 키 매핑을 이용하여 <필드키버퍼명_fdl.h>을 생성하여 프로그램 실행 시 사용하게 된다. 기존의 구조체 파일과는 달리 사용자가 원하는 필드의 값만을 조작하고 전달할 수 있어 자원의 낭비를 줄일 수 있다. 하지만 키 매핑하는 과정에서의 오버헤드도 발생할 수 있으므로 주의가 필요하다.

4.3.2. 서버 프로그램

서버 프로그램은 사용자의 요청을 받아서 처리하고 그 응답을 클라이언트에게 반환한다.

서버 프로그램 흐름

다음은 서버 프로그램의 흐름이다.



서버 프로그램 흐름

```
int tpsvrinit(int argc, char **argv)
{
     서버 초기화 작업(DBMS와의 Non-XA 모드 접속)
}

SVC_NAME(TPSVCINFO *msg)
{
     DB 트랜잭션 처리(ESQL), 업무 처리 로직 포함
     클라이언트로 응답 결과를 전송
}

int tpsvrdone()
{
     서버 종료 직전 작업(DBMS와의 Non-XA 모드 연결 해제)
}

void tpsvctimeout()
{
     SVCTIME 항목과 관련되며, 서비스 타임아웃 핸들러에 의해 호출됨
}
```

다음은 서버 프로그램의 주요 함수에 대한 설명이다.

- 서버 초기화 함수

함수	설명
tpsvrinit()	서버 애플리케이션 프로세스가 기동될 때 최초로 호출된다. tpsvrinit()의 표준 입력은 Tmax 환경 파일의 설정을 통하여 이루어진다. tpsvrinit()은 개발자가 재정의하여 사용한다. 재정의를 하지 않으면 Tmax 서버 라이브러리에 있는 기본 로직이 적용된 tpsvrinit()이 호출된다.

- 서버 종료 시 작업 함수

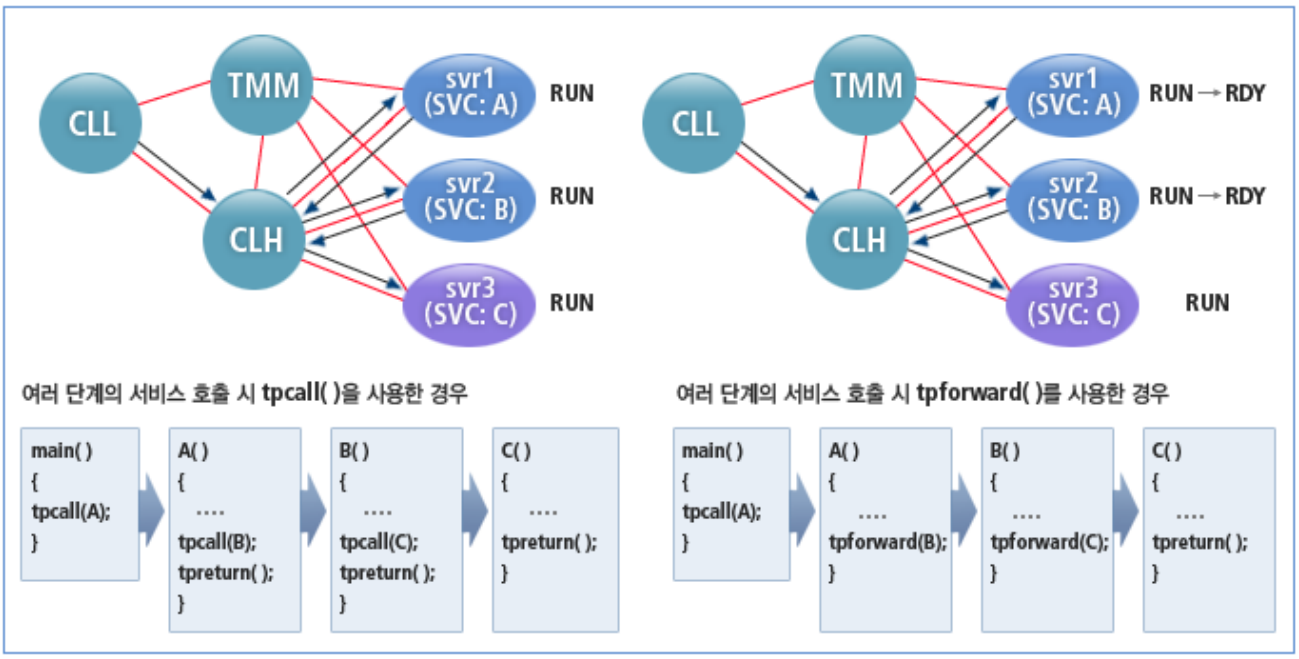
함수	설명
tpsvrdone()	서버 애플리케이션 프로세스가 정상 종료하기 직전 호출된다. tpsvrdone()은 개발자가 재정의하여 사용한다. 재정의를 하지 않으면 Tmax 서버 라이브러리 내에 있는 기본 로직이 적용된 tpsvrdone()이 호출된다.

- Tmax 응답 함수

함수	설명
tpreturn()	CLH로 서비스 로직의 종료를 알리고 응답 데이터를 전달하고 CLH는 서비스를 호출한 측의 정보를 확인하고 응답 데이터를 즉시 송신한다. 응답 데이터가 저장된 버퍼 포인터가 가리키는 동적 할당 메모리를 반환(free)하면 서비스를 포함하는 서버 애플리케이션은 프로세스의 동작을 멈추고 다음 응답을 받을 준비를 한다.

함수	설명
tpforward()	CLH로 서비스 로직의 종료를 알리고, SVC로 보낼 데이터를 전달하고 CLH는 전달할 서비스를 포함하는 서버 프로세스의 가용 여부를 확인 후 데이터를 즉시 전달한다. 응답 데이터 포인터가 가리키는 동적 할당 메모리를 반환(free)하면 서비스를 포함하는 서버 애플리케이션은 프로세스의 동작을 멈추고 다음 응답을 받을 준비를 한다.

다음은 서버 프로그램의 주요 함수 프로세스에 대한 그림이다.

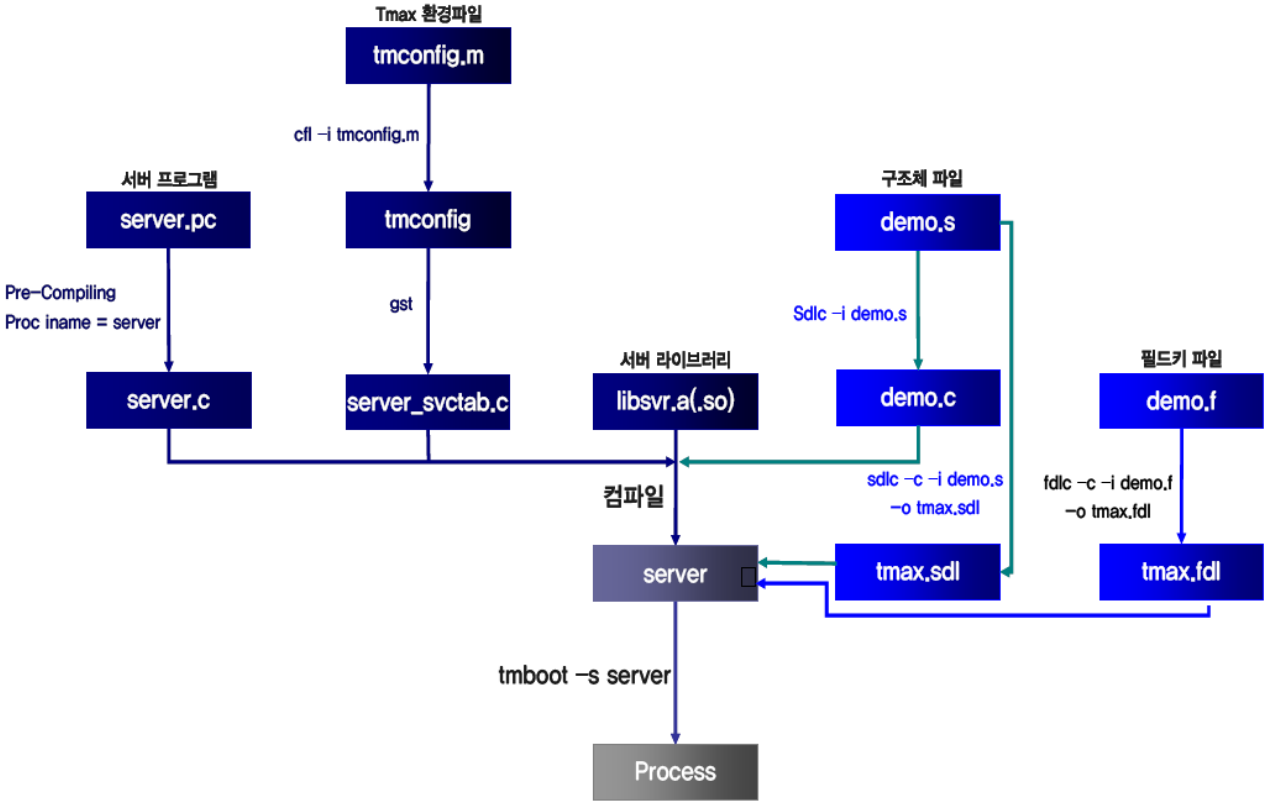


tpcall과 tpforward 비교

서버 프로그램 구성

서버 프로그램은 Tmax에서 제공하는 main()과 애플리케이션 개발자가 작성한 서비스 루틴(Service Routine)들로 구성된다.

다음은 서버 프로그램의 구성에 대한 그림이다.



Tmax 서버 프로그램 구성

- 서버 프로그램

  개발자가 작성한 서비스루틴이다. 클라이언트의 요청을 처리한다.

  SQL문을 사용하였다면 해당 데이터베이스 벤더가 제공하는 툴을 이용하여 프리컴파일을 마쳐야 한다.

- Tmax 서버 라이브러리(libsvr.a / libsvr.so)

  Tmax에서 제공하는 서버 라이브러리로, 서버 main()과 tpsvrinit(), tpsvrdone() 그리고 각종 Tmax 함수들이 있다.

- 서비스 테이블

  서버마다 제공되는 서비스명들이 나열된 파일이며, Tmax 환경 파일을 참조하여 생성한다.

  서비스 테이블은 서비스를 수행할 때 실제로 서버 내에서 해당 서비스 루틴의 위치를 찾기 위해 사용되며 시스템 관리자가 제공한다. 이에 대한 자세한 내용은 "Tmax Administration Guide"를 참조한다.

- 구조체 바이너리 테이블(SDLFILE)

  구조체형 버퍼(STRUCT, X_C_TYPE, X_COMMON)를 사용하였다면 이를 정의하는 <xxxx.s> 형식의 구조체 파일이 필요하다.

  구조체 파일은 표준 통신 타입(구조체 파일명_sdl.c)과 구조체 헤더 파일(구조체 파일명_sdl.h)이 있다. 구조체 파일은 **sdlc –c** 명령어를 이용하여 컴파일하며 그 결과 구조체의 멤버들을 표준 통신형으로 변환하는 데 필요한 바이너리 테이블 파일이 생성된다. 이는 서버 프로그램 실행 시에 구조체형의 데이터를 표준 통신형으로 변환 및 역변환하는 데 사용된다.

  구조체를 사용하지 않을 경우에는 $TMAXDIR/lib/sdl.o를 응용 서버와 컴파일해야 한다.

- 필드버퍼 바이너리 테이블(FDLFILE)

  필드 버퍼를 사용하였다면 <xxxx.f> 형식으로 정의된 필드 버퍼 파일이 필요하다. 이 파일은 **fdlc** 명령어를 이용하여 컴파일하며 그 결과 필드키와 데이터를 대응시켜 주는 바이너리 테이블 파일과 필드키와 필드키 이름을 대응시켜주는 헤더 파일(xxxx_fdl.h)이 생성된다. 기존의 구조체 파일과는 달리 사용자가 원하는 필드의 값만을 조작하고 전달할 수 있으므로 다양한 타입의 데이터를 사용하는 경우 자원의 낭비를 줄일 수 있다. 하지만, 데이터 값과 필드키 값을 같이 관리하므로 많은 필드를 사용하지 않는다면 역효과가 발생할 수도 있다.

- 구조체-표준버퍼 변환/역변환 프로그램

  서버 프로그램 내에서 구조체 버퍼를 사용하기 위해서는 **sdlc** 명령어로 만들어진 구조체-표준버퍼 변환/역변환 프로그램(xxxx_sdl.c, xxxx_sdl.h)을 컴파일하여 같이 링크해야 한다. 구조체를 사용하지 않을 때에는 TMAXDIR/lib/sdl.o를 링크하도록 한다.

4.4. 시스템 환경 파일

시스템 환경 파일은 Tmax 시스템에 필요한 정보를 담는 파일이다. Tmax 환경 파일은 Tmax 시스템 구성을 설정하며, Tmax 관리자가 작성한다. 이 파일은 서비스 테이블 생성과 Tmax 시스템 구동에 사용된다.

환경 파일은 8개의 절로 구성된다.

절	설명	필수 여부
DOMAIN 절	하나의 독립적인 Tmax 시스템 전체 환경을 정의한다.	필수
NODE 절	도메인을 이루는 각 노드에 관계된 환경을 정의한다.	필수
SVRGROUP 절	서버 그룹 및 데이터베이스에 관련된 사항을 정의한다.	필수
SERVER 절	서버에 관련된 사항을 정의한다.	필수
SERVICE 절	서비스에 관련된 사항을 정의한다.	필수
GATEWAY 절	도메인 간 게이트웨이에 관련된 사항을 정의한다.	선택
ROUTING 절	데이터 의존 라우팅에 관련된 사항을 정의한다.	선택
RQ 절	신뢰성 큐에 관련된 사항을 정의한다.	선택

- 각 절의 이름은 애스테리스크(*)로 시작(예: *DOMAIN, *NODE 등)한다.
- 각 절의 이름과 절의 하위 개체 이름은 반드시 줄의 첫 번째 칸에서 시작한다.
- 하나의 하위 개체에 대한 정의는 콤마(,)로 구분한다.

일반 텍스트 파일로 환경 파일이 생성되며 **cfl** 명령어로 컴파일한다.

```
cfl -i Tmax 환경 파일명
```

다음은 Tmax 환경 파일의 예이다(< >로 쌓여 있는 부분은 적당한 값으로 수정이 필요하다).

```
*DOMAIN
```

```
<resrc_name> SHMKEY = <UNIQUE IPCKEY>,
             MAXUSER = <256>,
             TPORTNO = <8999>

*NODE
<uname>       TMAXDIR = <TMAX installed directory>
             APPDIR = <APPLICATION directory>
             PATHDIR = <PATH directory>

*SVRGROUP
<svg_name>    NODENAME = <uname>,
             DBNAME = <ORACLE>,
             OPENINFO = "ORACLE_XA+Acc=P/tmaxsoft/tmaxsoft+SesTm=60"

*SERVER
<svr_name>    SVGNAME = <svg_name>,
             MIN = <5>,
             MAX = <10>

*SERVICE
<svc_name>    SVRNAME = <svr_name>
```

각 절에 대한 자세한 설명은 "Tmax Administration Guide"를 참고한다.

4.5. API

프로그램을 개발할 때에는 API의 원형이 정의되어 있는 헤더 파일을 include하여 사용해야 한다. API는 클라이언트/서버 라이브러리에 구현되어 있다. API에 대한 자세한 설명은 "Tmax Application Development Guide" 및 "Tmax Reference Guide"를 참고한다.

4.5.1. Tmax 표준 API

X/Open ATMI

X/Open ATMI(X/Open Application Transaction Monitor Interface) API는 X/Open DTP 모델에서 표준으로 규정한 ATMI 인터페이스로 애플리케이션 프로그램과 TP-Monitor 사이의 통신 방식을 하나로 사용한다.

함수는 atmi.h에 정의되어 있으며, 버퍼 관련 함수, 서비스 요청 및 응답 관련 함수, 대화형 모드 관련 함수, 서비스 종료 관련 함수로 나뉜다.

- 버퍼 할당, 해제 관련 함수

함수	설명
tpalloc()	데이터를 송수신할 버퍼를 할당하는 함수이다.

함수	설명
tprealloc()	버퍼의 크기를 변경하는 함수이다.
tpfree()	할당된 버퍼를 해제하는 함수이다.
tptypes()	버퍼의 크기와 형식에 대한 정보를 제공하는 함수이다.

- 서비스 요청 및 응답 관련 함수

함수	설명
tpcall()	서비스를 요청하고 응답이 올 때까지 대기하는 함수이다.
tpacall()	서비스를 요청하고 다른 처리를 하다가 tpgetrply() 함수가 호출될 때 처리 결과를 수신하는 함수이다.
tpcancel()	서비스 요청에 대한 응답을 취소하는 함수이다.
tpgetrply()	tpacall() 호출에 대한 응답을 수신하는 함수이다.

- 대화형 모드 관련 함수

함수	설명
tpconnect()	대화형 모드에서 메시지 송수신을 위한 연결 함수이다.
tpdiscon()	대화형 모드에서 서비스와 연결을 비정상적으로 종료하는 함수이다.
tprecv()	대화형 모드에서 메시지를 수신하는 함수이다.
tpsend()	대화형 모드에서 메시지를 송신하는 함수이다.

- 서비스 종료 관련 함수

함수	설명
tpreturn()	서비스 요청에 대한 응답을 클라이언트에게 보내고 서비스 루틴을 종료하는 함수이다.

X/Open TX API

X/Open TX API는 애플리케이션 프로그램과 TP-Monitor 간의 트랜잭션에 관해서 통신하는 방식을 제공한다. 함수는 tx.h에 정의되어 있고 트랜잭션 관리 함수로 구성된다.

다음은 TX API 목록이다.

- 트랜잭션 관련 함수

함수	설명
tx_begin()	트랜잭션을 시작하는 함수이다.
tx_commit()	트랜잭션을 commit 하는 함수로 결과를 저장한다.
tx_rollback()	트랜잭션을 원상태로 복구하는 함수이다.

함수	설명
tx_open()	내부적으로 작동하는 함수로 리소스 매니저를 시작하는 함수이다.
tx_close()	내부적으로 작동하는 함수로 리소스 매니저를 종료하는 함수이다.
tx_set_transaction_timeout()	트랜잭션이 종료되어야 할 시간 제한을 설정하는 함수이다.
tx_info()	전역 트랜잭션 정보를 반환하는 함수이다.
tx_set_commit_return()	전역 트랜잭션을 인가할 시점을 설정하는 함수이다.
tx_set_transaction_control()	트랜잭션 완료 후 자동으로 다음 트랜잭션을 시작하는 함수이다.

4.5.2. 비표준 API

Tmax ATMI

Tmax ATMI 함수는 비요청 메시지 처리, RQ 관련, 에러 설정, 타임아웃 설정 등에 사용되는 함수로 tmaxapi.h에 정의되어 있다. 정의된 API는 비표준 인터페이스로서 개발자의 개발 생산성 향상을 위해 개발되었으며 개발자에 의해 작성되는 애플리케이션 프로그램과 TP-Monitor 간의 통신 방식의 하나로 사용될 수 있다.

다음은 tmaxapi.h에 정의된 비표준 API 목록이다.

- 비요청 데이터 관련 함수

함수	설명
tpbroadcast()	비요청 데이터를 시스템에 등록된 클라이언트에게 일방적으로 전달하는 함수이다.
tpsetunsol()	비요청 데이터를 처리할 함수를 지정하는 함수이다.
tpgetunsol()	비요청 데이터를 수신하는 함수이다.
tpsetunsol_flag()	비요청 데이터 수신 플래그를 설정하는 함수이다.
tpchkunsol()	비요청 데이터의 도착을 확인하는 함수이다.

- 에러 관련 함수

함수	설명
tpstrerror()	에러의 내용을 스트링 형식으로 출력하는 함수이다.
Userlog()	먼저 에러를 버퍼에 기록(Log)하는 함수이다.
ulogsync()	디스크의 메모리 버퍼에 'ulog' 내용을 저장하는 함수이다.
UserLog()	userlog()와 ulogsync()의 기능이 복합된 함수이다.
gettperrno()	Tmax 시스템을 호출할 때 발생한 에러 번호를 반환하는 함수이다.
gettpurcode()	개발자가 설정한 urcode를 반환하는 함수이다.
tperrordetail()	Tmax 시스템을 호출할 때 발생한 에러에 관한 정보를 반환하는 함수이다.

- 소켓 정보 관련 함수

함수	설명
tpgetpeer_ipaddr()	연결된 클라이언트의 IP 주소를 반환하는 함수이다.
tpgetpeername()	연결된 클라이언트명을 반환하는 함수이다.
tpgetsockname()	연결된 클라이언트의 소켓명을 반환하는 함수이다.

- 블록 타임아웃 설정 함수

함수	설명
tpset_timeout()	블록 타임아웃 시간을 설정하는 함수이다.

- 장애대책 관련 함수

함수	설명
tptobackup()	백업 머신으로 연결을 맺는 함수이다.

- 연결 함수

함수	설명
tpstart()	Tmax 시스템과 연결을 시작하는 함수이다.
tpend()	Tmax 시스템과 연결을 종료하는 함수이다.

- RQ 관련 함수

함수	설명
tpenq()	RQ에 클라이언트에서 보낸 요청을 저장하는 함수이다.
tpdeq()	RQ에 있는 데이터를 불러들이는 함수이다.
tpqstat()	RQ에 저장된 데이터 통계를 요구하는 함수이다.
tpextsvcname()	RQ에 저장된 데이터에서 서비스명을 요구하는 함수이다.

- 환경변수 관련 함수

함수	설명
tmaxreadenv()	파일에서 환경변수를 불러들이는 함수이다.
tpputenv()	환경변수를 설정하는 함수이다.
tpgetenv()	환경변수 값을 반환하는 함수이다.

- Window 조작 관련 함수

함수	설명
WinTmaxStart()	Tmax 시스템과 연결하는 함수이다.
WinTmaxEnd()	Tmax 시스템과 연결을 해제하는 함수이다.
WinTmaxSetContext()	Window 핸들을 지정하는 함수이다.
WinTmaxSend()	데이터를 송신하는 함수이다.
WinTmaxAcall()	Window용 비동기 함수이다.
WinTmaxAcall2()	데이터의 수신을 Callback 함수로 처리하는 Window용 비동기 함수이다.

- 기타 함수

함수	설명
tpscmt()	환경 파일의 트랜잭션 제어 관련 설정을 무효화하는 함수이다.
tpgetlev()	트랜잭션 모드를 확인하는 함수이다.
tpchkauth()	인증 필요 여부를 확인하는 함수이다.
tpgprio()	서비스 요청 우선순위를 확인하는 함수이다.
tpsprio()	서비스 요청 우선순위를 설정하는 함수이다.
tpsleep()	지정된 시간 내에 메시지 수신에 대기하는 함수이다.
tp_sleep()	초 단위로 데이터 수신에 대기하는 함수이다.
tp_usleep()	마이크로 초 단위로 데이터 수신에 대기하는 함수이다.
tpschedule()	큐에 쌓여있는 업무를 꺼내 UCS에게 처리를 할당하는 함수이다.
tpuschedule()	UCS서버 프로세스에서 데이터의 도착을 입력한 시간 동안 기다리는 함수이다.
tpsvrinit()	Tmax 서버 프로세스를 초기화하는 함수이다.
tpsvrdone()	Tmax 서버 프로세스 종료 루틴을 호출하는 함수이다.
tpsvctimeout()	UCS 서버 프로세스를 다운하는 함수이다.
tmadmin()	서비스 호출의 형태로 시스템 관리하는 함수이다.

다음은 atmi.h에 정의된 비표준 API 목록이다.

- 서비스 종료 관련 함수

함수	설명
tpforward()	자신의 서비스 처리를 종료하고 클라이언트의 요청을 또다른 서비스 루틴으로 전달한다.

- 클라이언트 연결 관련 함수

함수	설명
tpstart()	클라이언트 애플리케이션과 Tmax를 연결한다.
tpend()	클라이언트 애플리케이션과 Tmax를 연결을 해제한다.

FDL API

FDL(Field Definition Language)은 비표준 API이며, 개발자의 개발 생산성 향상을 위해 개발되었다. FDL은 필드키라고 불리는 인덱스와 데이터가 함께 관리되는 associative-typed data에 해당한다. FIELD 버퍼에 관련한 API는 fbuf.h에 정의되어 있다.

이와 같은 데이터는 Tmax 시스템에서 제공하는 버퍼의 일종인 필드키 버퍼에 실리며 이 버퍼를 조작하기 위한 다음과 같은 함수를 제공한다.

- 필드키 사상함수

함수	설명
fbget_fldkey()	필드 이름에 대한 필드키 값을 반환한다.
fbget_fldname()	필드키의 이름을 반환한다.
fbget_fldno()	필드키로부터 필드 번호를 가져온다.
fbget_fldtype()	필드키로부터 필드형(type)을 가져온다. (정수 값 반환)
fbget_strfldtype()	필드키로부터 형에 대한 포인터 값을 가져온다.

- 버퍼 할당 관련 함수

함수	설명
fbisfbuf()	지정된 버퍼가 필드화되어 있는지 알아본다.
fbinit()	필트키 버퍼로 할당된 메모리 공간을 초기화시킨다.
fbcalcsize()	필드 버퍼의 크기를 계산한다.
fballoc()	필트키 버퍼를 동적으로 할당한다.
fbfree()	필드 버퍼를 해제한다.
fbget_fbsize()	바이트 단위로 필드키 버퍼 크기를 반환한다.
fbget_unused()	사용되지 않은 필드 버퍼 공간을 확인한다.
fbget_used()	사용 중인 필드키 버퍼 공간을 바이트 수 단위로 반환한다.
fbrealloc()	버퍼 크기를 조절한다.

- 필드 접근 및 수정 함수

함수	설명
fbput()	필드 버퍼에 필드키 값을 추가한다.
fbinsert()	필드키와 위치를 지정하고 필드 버퍼에 필드 값을 저장한다.
fbchg_tu()	데이터를 전송하기 전에 지정된 필드 버퍼를 이동시킨다.
fbdelete()	버퍼의 필드 데이터를 삭제한다.
fbdelall()	필드의 모든 값을 삭제한다.
fbdelall_tu()	필드키 배열(fieldkey[])에 나열된 모든 필드의 데이터를 삭제한다.

함수	설명
fbget()	버퍼에 있는 필드 내용을 찾는다.
fbgetf()	필드 버퍼에 있는 지정된 필드키의 필드 값을 얻는다.
fbget_tu()	지정된 위치에 있는 특정 필드키의 값을 얻는다.
fbnext_tu()	필드 버퍼의 특정 필드키의 필드 값을 순서대로 얻는다.
fbgetalloc_tu()	반환된 데이터를 저장하기 위해 다른 버퍼를 내부적으로 할당하고 그 포인터만을 버퍼에 반환한다.
fbgetval_last_tu()	필드 버퍼의 특정 필드키 occurrence와 최근 데이터 값을 얻는다.
fbgetlast_tu()	필드 버퍼에 지정된 필드의 최근 엔트리 데이터를 얻는다.
fbgetnth()	특정 필드 값을 검색한다.
fbfldcount()	특정 버퍼에 포함된 필드의 개수를 반환한다.
fbkeyoccur()	필드키에 지정된 필드 번호를 반환한다.
fbispres()	요청한 데이터가 필드 버퍼에 존재하는지 확인한다.
fbgetval()	요청한 데이터의 길이와 그 위치의 포인터를 반환한다.
fbgetvall_tu()	필드의 실제 값을 long 형식(type)으로 반환한다.
fbupdate()	지정된 위치의 필드 버퍼 안의 필드키의 필드 값을 갱신한다.
fbgetlen()	필드 버퍼 안의 지정한 필드키에 해당 되는 첫 번째 occurrence의 값을 반환한다.

- 변환 함수

함수	설명
fbtypecvt()	데이터 형식을 변환한다.
fbputt()	새로운 데이터 값과 데이터 형식을 필드 버퍼로 덧붙인다.
fbget_tut()	지정된 위치의 필드 데이터를 얻고 필드키의 형식을 지정한다.
fbgetalloc_tut()	반환된 데이터를 정의된 데이터 형식으로 변환하고 저장하기 위해 내부적으로 다른 버퍼를 할당한다.
fbgetvalt()	반환된 값의 포인터를 반환한다.
fbgetvali()	integer 형식의 필드 데이터를 반환한다.
fbgetvals()	string 형식의 필드 데이터를 반환한다.
fbgetvals_tu()	지정된 위치의 string 형식의 필드 데이터를 반환한다.
fbgetntht()	변환된 값을 반환한다.
fbchg_tut()	필드 버퍼의 특정 시작 지점에서 필드키의 값을 바꾼다.

- 버퍼 연산 관련 함수

함수	설명
fbbufop()	두 필드 버퍼의 내용을 비교, 복사, 이동, 변경한다.

- 함수

함수	설명
fbbufop_proj()	필드키에 해당되는 버퍼를 변경한다.

- I/O 관련 함수

함수	설명
fbbufop_proj()	필드키에 해당되는 버퍼를 변경한다.
fbread()	표준 입출력 라이브러리와 같이 사용하는 함수로 파일로부터 필드 버퍼를 읽어 들인다.
fbwrite()	표준 입출력 라이브러리와 같이 사용하는 함수로 파일에 쓰기를 한다.
fbprint()	표준 입출력으로 버퍼의 내용을 출력한다.
fbfprint()	필드 버퍼의 가능한 데이터를 파일 스트링으로 출력한다.

- 에러 관련 함수

함수	설명
fbstrerror()	필드 버퍼 조작 시 발생한 에러의 메시지를 스트링 형태로 얻는다.
getfberrno()	에러가 발생하는 경우 에러 번호를 반환한다.

- 기타 함수

함수	설명
fbmake_fldkey()	FDLFILE에 기록하지 않지만 새로운 필드키를 자동적으로 생성한다.
fbftos()	필드 버퍼에 저장된 데이터를 C 구조체(stname)로 옮긴다.
fbstof()	C 구조체로 저장된 데이터를 구조체 파일에 매핑되는 FIELD 버퍼로 옮긴다.
fbsnull()	C 구조체의 지정된 필드키 occurrence의 구조체 멤버 변수와 필드 버퍼와 매핑되는 것이 NULL 인지의 여부를 확인한다.
fbstelinit()	필드 버퍼와 매핑되는 C 구조체 멤버 변수를 NULL로 초기화시킨다.
fbstinit()	필드 버퍼와 매핑되는 C 구조체를 NULL로 초기화시킨다.

4.6. 에러 메시지

4.6.1. X/Open DTP 관련 에러

X/Open DTP에서 제공하는 인터페이스와 Tmax 시스템에서 제공하는 비표준 인터페이스를 사용하는 경우에 에러가 발생하면 해당 상황에 적절한 에러 값이 tperrno라고 불리는 전역 변수에 설정된다. 따라서 개발자는 에러가 발생하면 tperrno를 확인하여 적절한 후속 조처를 할 수 있다.

다음은 tperrno 에러 메시지 목록이다. tperrno는 에러상황이 발생할 경우에 설정이 되는 전역변수이다.

Error Message(tperrno)	설명
TPEBADDESC(2)	비동기식이나 대화형 타입에서 잘못된 Descriptor가 사용되는 경우에 발생한다.
TPEBLOCK(3)	네트워크 오류인 경우에 발생한다.
TPEINVAL(4)	적절하지 않은 인자가 입력된 경우 발생한다.
TPELIMIT(5)	시스템에서 제공하는 각종 한계값을 벗어난 경우에 발생한다.
TPENOENT(6)	서비스가 제공되지 않는 경우 발생한다.
TPEOS(7)	시스템적인 문제로 연결이 불가능한 경우 발생한다.
TPEPROTO(9)	프로토콜의 오류인 경우 발생한다.
TPESVCERR(10)	응용 프로그래밍의 실패로 인한 Tmax 시스템 버퍼가 손상된 경우에 발생한다.
TPESVCFAIL(11)	응용 프로그램의 레벨 서비스 오류인 경우 발생한다.
TPESYSTEM(12)	Tmax 내부 오류(로그 메시지 확인)인 경우 발생한다.
TPETIME(13)	처리시간이 초과(BLOCKTIME)된 경우 발생한다.
TPETRAN(14)	트랜잭션 실패로 트랜잭션이 취소되는 경우 발생한다.
TPGOTSIG(15)	시그널이 발생된 경우이다.
TPEITYPE(17)	등록되지 않은 구조체 형식이나 필드키가 사용된 경우이다.
TPEOTYPE(18)	버퍼 사용 오류 혹은 버퍼 형식 오류가 발생한 경우이다.
TPEEVENT(22)	대화형 모드에서 이벤트가 발생한 경우이다.
TPEMATCH(23)	RQ의 tpdeq() 함수를 호출할 때 해당 서비스에 대한 결과가 없는 경우에 발생한다.
TPENOREADY(24)	서버 프로세스가 동작이 준비되지 않은 경우에 발생한다.
TPESECURITY(25)	보안상 오류인 경우 발생한다.
TPEQFULL(26)	서버 프로세스의 큐 대기시간이 초과된 경우에 발생한다.
TPEQPURGE(27)	큐 퍼지로 인해 큐에서 제거되는 경우에 발생한다.
TPECLOSE(28)	Tmax 시스템과 연결이 해제되는 경우에 발생한다.
TPESVRDOWN(29)	응용 프로그램 오류로 서버 프로세스가 다운된 경우에 발생한다.
TPEPRESVC(30)	이전 서비스 처리 도중 에러가 발생한 경우이다.
TPEMAXNO(31)	동시 사용자 수가 한계값에 도달한 경우 발생한다.

에러에 대한 설명과 대응 방법에 대한 자세한 내용은 "Tmax Application Development Guide" 및 "Tmax Error Message Reference Guide"를 참고한다.

4.6.2. FDL 관련 에러

FDL 인터페이스에 대한 에러는 전역 변수 fberrno에 설정된다. 따라서 개발자는 에러가 발생하면 fberrno를 확인하여 적절한 후속 조치를 할 수 있다. 본 절에서는 FDL 관련한 에러 메시지에 대해 설명한다.

다음은 fberror 에러 메시지 목록이다. fberror은 에러상황이 발생할 경우에 설정이 되는 전역변수이다.

Error Message(fberror)	설명
FBEBADFB(3)	적절하지 않은 버퍼를 사용(필드키 버퍼가 아님)한 경우에 발생한다.
FBEINVAL(4)	적절하지 않은 인자값이 사용된 경우에 발생한다.
FBELIMIT(5)	시스템에서 제공하는 한계값을 벗어난 경우에 발생한다.
FBENOENT(6)	해당 필드키가 버퍼에 존재하지 않는 경우에 발생한다.
FBEOS(7)	운영체제에 오류가 발생한 경우이다.
FBEBADFLD(8)	적절하지 않은 필드키가 사용된 경우이다.
FBEPROTO(9)	프로토콜에 에러가 발생한 경우이다.
FBENOSPACE(10)	버퍼 공간이 부족한 경우 발생한다.
FBEMALLOC(11)	메모리 할당에 오류가 발생한 경우이다.
FBESYSTEM(12)	시스템에 오류가 발생한 경우이다.
FBETYPE(13)	타입에 오류가 발생한 경우이다.
FBEMATCH(14)	일치하는 값이 없는 경우에 발생한다.
FBEBADSTRUCT(15)	등록되지 않은 구조체가 사용된 경우에 발생한다.
FBEMAXNO(19)	존재하지 않은 에러 번호가 사용된 경우에 발생한다.

에러에 대한 설명과 대응 방법에 대한 자세한 내용은 "Tmax FDL Reference Guide" 및 "Tmax Error Message Reference Guide"를 참고한다.

5. 예제

본 장에서는 Tmax 시스템에서 제공하는 고유의 기능을 사용하기 위한 예제를 설명한다.

5.1. 통신 유형 예제

본 절에서는 Tmax에서 사용하는 3가지 통신형인 동기형, 비동기형, 대화형 애플리케이션의 간단한 예제를 통해 전체적인 흐름에 대해 설명한다.

5.1.1. 동기형 통신

클라이언트는 STRING 버퍼에 문자열을 복사해서 서비스를 호출하고, 서버의 서비스 루틴은 이 문자열을 받아서 대문자열로 바꾸어 반환하는 프로그램이다.

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
sample.m	Tmax 환경설정 파일이다.

- 클라이언트 프로그램

프로그램 파일	설명
sync_cli.c	클라이언트 프로그램이다.

- 서버 프로그램

프로그램 파일	설명
syncsvc.c	대문자로 바꾸는 서비스 프로그램이다.
Makefile	Tmax에서 제공되는 Makefile로 수정해야 한다.

프로그램 특징

- 클라이언트 프로그램

기능	설명
Tmax 연결	기본 연결한다. (클라이언트 정보 없음)
버퍼 유형	STRING이다.
통신 유형	tpcall()을 이용한 동기형 통신을 한다.

- 서버 프로그램

기능	설명
서비스	TOUPPERSTR이다.
데이터베이스 연결	없음

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
*DOMAIN
resrc         SHMKEY = 77990, MAXUSER = 256

*NODE
tmax          TMAXDIR = "/home/tmax",
              APPDIR =  "/home/tmax/appbin",
              PATHDIR = "/home/tmax/path",
              TLOGDIR = "/home/tmax/log/tlog",
              ULOGDIR = "/home/tmax/log/slog",
              SLOGDIR = "/home/tmax/log/ulog"

*SVRGROUP
svg1          NODENAME = tmax

*SERVER
syncsvc       SVGNAME = svg1,
              MIN = 1, MAX = 5,
              CLOPT = " -e $(SVR).err -o $(SVR).out "

*SERVICE
TOUPPERSTR    SVRNAME = syncsvc
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<sync_cli.c>

```
#include <stdio.h>
#include <string.h>
#include <usrinc/atmi.h>

main(int argc,char *argv[])
{
       char *sendbuf, *recvbuf;
       long rlen;

       if (argc != 2)
       {
              fprintf(stderr,“Usage: $ %s string \n”,argv[0]);
              exit(1);
       }
```

```
        if (tpstart((TPSTART_T*)NULL) == -1)
        {
                fprintf(stderr,“Tpstart failed\n”);
                exit(1);
        }

        if ((sendbuf = tpalloc(“STRING”,NULL,0)) == NULL) {
                fprintf(stderr,“Error allocation send buffer\n”);
                tpend();
                exit(1);
        }

        if ((recvbuf = tpalloc(“STRING”,NULL,0)) == NULL) {
                fprintf(stderr,“Error allocation recv buffer\n”);
                tpend();
                exit(1);
        }

        strcpy(sendbuf ,argv[ 1 ] ) ;

        if ( tpcall(“TOUPPERSTR”,sendbuf,0,&sendbuf,&rlen, TPNOFLAGS) == -1)
        {
                fprintf(stderr,“Can’t send request to service TOUPPER->%s!\n”,
                tpstrerror(tperrno)) ;
                tpfree(sendbuf) ;
                tpfree(recvbuf) ;
                tpend();
                exit(1);
        }
        printf(“Sent value:%s\n ”,sendbuf);
        printf(“Returned value:%s\n ”,recvbuf);
        tpfree(sendbuf);
        tpfree(recvbuf);
        tpend( );
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

<syncsvc.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>

TOUPPERSTR(TPSVCINFO *msg)
{
        int i;

        for (i = 0; i < msg->len ; i++)
        msg->data[i] = toupper(msg->data[i]);
        msg->data[i] = ‘\0’;

        tpreturn(TPSUCCESS, 0, msg->data, 0, TPNOFLAGS);
}
```

5.1.2. 비동기형 통신

클라이언트는 STRUCT 버퍼의 멤버에 문자열을 복사해서 서비스를 호출하고, 서버의 서비스 루틴은 이 문자열을 받아서 소문자열 혹은 대문자열로 바꾸어 반환하는 프로그램이다. 클라이언트는 비동기형 통신으로 TOUPPER 서비스를 요청하고 다시 동기형으로 TOLOWER 서비스를 호출하여 결과를 받은 후 앞서 요청한 TOUPPER 서비스의 수행 결과를 받는다.

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
demo.s	구조체 버퍼를 정의한다.
sample.m	Tmax 환경설정 파일이다.

- 클라이언트 프로그램

프로그램 파일	설명
async_cli.c	클라이언트 프로그램이다.

- 서버 프로그램

프로그램 파일	설명
asyncsvc.c	대문자/소문자로 바꾸는 서비스 프로그램이다.
Makefile	Tmax에서 제공되는 Makefile로 수정해야 한다.

프로그램 특징

- 클라이언트 프로그램

기능	설명
Tmax 연결	기본 연결을 한다.
버퍼 유형	STRUCT이다.
통신 유형	동기형 및 비동기형이다.

- 서버 프로그램

기능	설명
서비스	TOUPPER, TOLOWER이다.
데이터베이스 연결	없음
통신 유형	동기형 및 비동기형이다.

구조체 버퍼

다음은 비동기형 통신에서 사용하는 구조체 버퍼이다.

<demo.s>

```
struct strdata {
   int flag;
   char sdata[20];
};
```

Tmax 환경 파일

다음은 Tmax 환경 파일 예제이다.

<sample.m>

```
*DOMAIN
resrc         SHMKEY = 77990, MAXUSER = 256

*NODE
tmax          TMAXDIR = "/home/tmax",
              APPDIR = "/home/tmax/appbin",
              PATHDIR = "/home/tmax/path"

*SVRGROUP
svg1          NODENAME = tmax

*SERVER
asyncsvc      SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
TOUPPER        SVRNAME = asyncsvc
TOLOWER        SVRNAME = asyncsvc
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<async_cli.c>

```
#include <stdio.h>
#include <string.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

main(int argc,char *argv[ ])
{
       struct strdata *sendbuf, *sendbuf1;
       long dlen,clen;
       int cd;

       if (argc != 3) {
              fprintf(stderr, "Usage: $ %s string STRING\n", argv[0], argv[1]);
              exit(1) ;
```

```
        }

        if (tpstart((TPSTART_T *)NULL) == -1) {
                fprintf(stderr, “TPSTART_T failed\n”);
                exit(1) ;
        }

        sendbuf = (struct strdata *)tpalloc(“STRUCT”, “strdata”, 0);
        if ( sendbuf == NULL) {
                fprintf(stderr, “Error allocation send buffer\n”);
                tpend () ;
                exit(1) ;
        }

        sendbuf1 = (struct strdata *)tpalloc(“STRUCT”, “strdata”, 0);
        if (sendbuf1 == NULL) {
                fprintf(stderr, “Error allocation send1 buffer\n”);
                tpend();
                exit(1) ;
        }

        strcpy(sendbuf->sdata, argv[1]);
        strcpy(sendbuf1->sdata, argv[2]);

        if ((cd = tpacall(“TOUPPER”, (char *)sendbuf, 0, TPNOFLAGS)) == -1)
        {
                fprintf(stderr, “Toupper error -> %s”, tpstrerror(tperrno));
                tpfree((char *)sendbuf);
                tpend();
                exit(1) ;
        }
        if (tpcall(“TOLOWER”,(char *)sendbuf1,0,(char **)&sendbuf1, &dlen,
                                TPSIGRSTRT) == -1) {
                fprintf(stderr, “Tolower error -> %s”, tpstrerror(tperrno));
                tpfree((char *)sendbuf);
                tpend();
                exit(1) ;
        }
        if (tpgetrply(&cd, (char **)&sendbuf, &clen, TPSIGRSTRT) == -1) {
                fprintf(stderr, “Toupper getrply error -> %s”, tpstrerror(tperrno));
                tpfree((char *)sendbuf);
                tpend();
                exit(1) ;
        }
        printf(“Return value %s\n %s\n”, sendbuf -> sdata, sendbuf1 -> sdata);
        tpfree((char *)sendbuf);
        tpfree((char *)sendbuf1);
        tpend() ;
}
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

<asyncsvc.c>

```
#include <stdio.h>
```

```
#include <usrinc/atmi.h>
#include “../sdl/demo.s”

TOUPPER(TPSVCINFO *msg)
{
       int i = 0;
       struct strdata *stdata;

       stdata = (struct strdata *)msg -> data;

       while (stdata->sdata[ i ] != ‘0’) {
       stdata->sdata[ i ] = toupper(stdata->sdata[ i ]);
       i++ ;
       }

       tpreturn(TPSUCCESS, 0, (char *)stdata, 0, TPNOFLAGS);
}

TOLOWER(TPSVCINFO *msg)
{
       int i = 0;
       struct strdata *stdata;

       stdata = (struct strdata *)msg -> data;

       while ( stdata->sdata[ i ] != ‘0’) {
            stdata->sdata[ i ] = tolower(stdata->sdata[ i ]);
            i++;
       }

       tpreturn(TPSUCCESS, 0, (char *)stdata, 0, TPNOFLAGS);
}
```

5.1.3. 대화형 통신

클라이언트는 사용자의 입력을 받아 STRING 버퍼를 통해 고유번호를 보내며, 서버의 서비스 루틴은 데이터베이스에 저장된 테이블에서 그 고유번호보다 큰 번호를 가지는 고객정보를 구조체를 통해 반환한다.

클라이언트는 대화형 모드를 설정하면서 고유번호를 보내며 대화 주도권은 서버에 넘긴다. 서버는 조건을 만족하는 데이터베이스의 모든 데이터를 커서를 통해 읽어서 클라이언트로 보낸다. 클라이언트는 TPEVSVCSUCC를 통해 이상 없이 모든 데이터를 읽어왔다는 것을 확인할 수 있다.

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
demo.s	구조체를 정의한 파일이다.
sample.m	Tmax 환경설정 파일이다.
mktable.sql	테이블 생성 스크립트이다.
sel.sql	테이블 및 데이터 출력 스크립트이다.

- 클라이언트 프로그램

프로그램 파일	설명
conv_cli.c	클라이언트 프로그램이다.

- 서버 프로그램

프로그램 파일	설명
convsvc.pc	서버 프로그램이다.
Makefile	Tmax에서 제공되는 Makefile을 수정해야 한다.

프로그램 특징

- 클라이언트 프로그램

기능	설명
Tmax 연결	기본 연결을 한다.
버퍼 유형	보낼 때 STRING, 받을 때 STRUCT이다.
통신 유형	대화형 통신 STRUCT이다.

- 서버 프로그램

기능	설명
서비스	MULTI
데이터베이스 연결	Oracle 사용

구조체 버퍼

다음은 대화형 통신에서 사용하는 구조체 버퍼이다.

<demo.s>

```
struct sel_o {
    char seqno[10];
    char corpno[10];
    char compdate[8];
    int totmon;
    float guarat;
    float guamon;
} ;
```

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
* DOMAIN
resrc        SHMKEY = 77990, MAXUSER = 256

*NODE
tmax          TMAXDIR = “/home/tmax”,
              APPDIR  = “/home/tmax/appbin”,
              PATHDIR =“/home/tmax/path”

* SVRGROUP
svg1          NODENAME = tmax,
              DBNAME = ORACLE,
              OPENINFO = “ORACLE_XA+Acc=P/scott/tiger+SesTm=60”,
              TMSNAME = svg1_tms

*SERVER
convsvc       SVGNAME = svg1, CONV = Y

*SERVICE
MULTI         SVRNAME = convsvc
```

다음은 환경설정에 추가된 항목에 대한 설명이다.

항목	설명
DBNAME	사용하는 데이터베이스명을 정의한다.
OPENINFO	Oracle 데이터베이스와 연동하기 위한 연결 정보를 설정한다.
TMSNAME	전역 트랜잭션 처리를 주재하는 프로세스명을 설정한다.
CONV	대화형 모드 서버를 지정한다.

데이터베이스 스크립트

다음은 Oracle 테이블 생성 스크립트의 예제이다.

<mktable.sql>

```
sqlplus scott/tiger << EOF
create table multi_sel
(
       seqno      VARCHAR(10),
       corpno     VARCHAR(10),
       compdate   VARCHAR(8),
       totmon     NUMERIC(38),
       guarat      FLOAT,
       guamon     FLOAT
);
create unique index idx_tdb on multi_sel(seqno);
EOF
```

다음은 Oracle 테이블 및 데이터 출력 스크립트의 예제이다.

<sel.sql >

```
sqlplus scott/tiger << EOF
Desc multi_sel;
select * from multi_sel;
EOF
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<conv_cli.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

main(int argc, char *argv[])
{
        struct sel_o *rcvbuf;
        char *sndbuf;
        long sndlen, rcvlen, revent;
        int cd;

        if (argc != 2) {
                printf("Usage: client string\n");
                exit(1);
        }

        /* tpstart()와 함께 Tmax에 연결함. */
        if (tpstart((TPSTART_T *) NULL) == -1) {
                printf("tpstart failed\n");
                exit(1);
        }

        if ((sndbuf = tpalloc("STRING", NULL, 12)) == NULL) {
                printf("tpalloc failed:sndbuf\n");
                tpend();
                exit(1);
        }

        if ((rcvbuf = (struct sel_o *)tpalloc("STRUCT", "sel_o", 0)) == NULL) {
                printf("tpalloc failed:rcvbuf\n");
                tpfree(sndbuf);
                tpend();
                exit(1);
        }
        strcpy(sndbuf, argv[1]);

        if ((cd = tpconnect ("MULTI", sndbuf, 0, TPRECVONLY)) == -1){
                printf("tpconnect failed:CONVER service, tperrno=%d\n", tperrno);
                tpfree(sndbuf);
                tpfree((char *)rcvbuf);
                tpend();
                exit(1);
        }

      /* 대화형 통신 연결, 대화 주도권은 서버 측에 넘김 */
```

```
        printf("tpconnect SUCESS \"MULTI\" service\n");
        while ( 1 ) {    /* 다중 데이터 수신. */
                printf("tprecv strat\n");
                if( tprecv(cd, (char **)&rcvbuf, &rcvlen, TPNOTIME, &revent) < 0 ) {
                /* 서버에서 tpreturn()으로 끝냈다면 */
                if (revent == TPEV_SVCSUCC){
                        printf("all is completed\n");
                        break;
                }
                printf("tprecv failed, tperrno=%s, revent=%x\n",
                tpstrerror(tperrno), revent );
                tpfree(sndbuf);
                tpfree((char *)rcvbuf);
                tpend();
                exit(1);
        }
        printf("seqno = %s\t\t corpno =%s\n", rcvbuf->seqno, rcvbuf->corpno);
        printf("compdate = %s\t\t totmon =%d\n", rcvbuf->compdate, rcvbuf->totmon);
        printf("guarat = %f\t\t guamon =%f\n\n\n", rcvbuf->guarat, rcvbuf->guamon) ;
}
        tpfree(sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
        printf("FINISH\n");
}
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

<convsvc.pc>

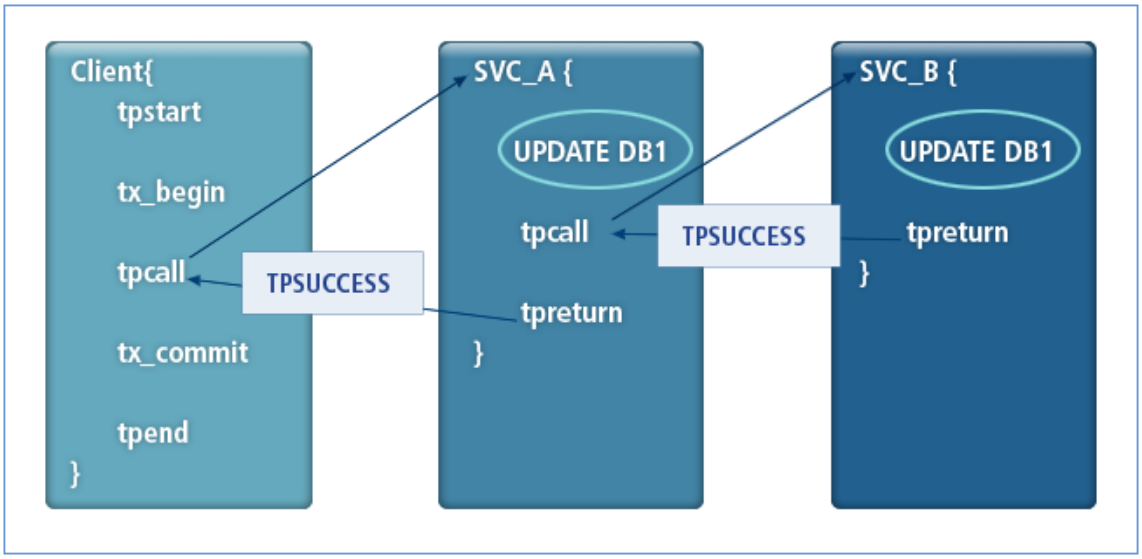
```
#include <stdio.h>
#include <string.h>
#include <usrinc/atmi.h>
#include “../sdl/demo.s”

EXEC SQL begin declare section; /* Oracle 전역 변수 선언 */
        char seq[10];
        struct sel_o *sndbuf;
EXEC SQL end declare section;
EXEC SQL include sqlca;

MULTI(TPSVCINFO *msg)
{
        int i, cd;
        long sndlen, revent;

        memset(seq, 0, 10);
        strcpy(seq, msg->data);

        if ((sndbuf = (struct sel_o *) tpalloc (“STRUCT”, “sel_o”, 0)) == NULL) {
                printf(“tpalloc failed:\n”);
                tpreturn (TPFAIL, -1, NULL, 0, TPNOFLAGS);
        }

        /* 다량 데이터 위해 커서 선언 */
```

```
        EXEC SQL declare democursor cursor for
        select *
        from corp
        where seqno > :seq;

        EXEC SQL open democursor;
        EXEC SQL whenever not found goto end_of_fetch;
        if (sqlca.sqlcode != 0){
               printf("oracle sqlerror=%s", sqlca.sqlerrm.sqlerrmc);
               tpreturn (TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
        }

         /* Oracle 에러가 없는 동안 데이터 전송 */
        while ( sqlca.sqlcode == 0 ){
               EXEC SQL fetch democursor into :sndbuf;

               if (tpsend (msg->cd, (char *)sndbuf, 0, TPNOTIME, &revent) == -1){
                      printf("tpsend failed, tperrno=%d, revent=%x\n", tperrno,
                               revent ) ;
                      tpfree ((char *)sndbuf);
                      tpreturn (TPFAIL, -1, NULL, 0, TPNOFLAGS);
               }
        }

        tpreturn (TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);

        end_of_fetch:
        exec sql close democursor;
        printf("tpreturn before");
        tpreturn (TPSUCCESS, 0, NULL, 0, TPNOFLAGS);
}
```

5.2. 전역 트랜잭션 프로그램 예제

전역 트랜잭션(Global Transaction Processing)이란 하나 이상의 자원 관리자(데이터베이스)와 하나 이상의 물리적인 사이트가 하나의 논리적인 단위로 참여하는 트랜잭션이다. Tmax 시스템에서는 모든 트랜잭션을 일단 전역 트랜잭션으로 간주하며, 데이터의 무결성을 위해 2PC(2 Phase Commit)를 사용한다.

클라이언트는 사용자의 입력을 받아 구조체 버퍼를 통해 고유번호와 데이터를 보내며 서버는 해당 고유번호의 데이터를 업데이트하고 이 데이터로 다른 데이터베이스를 사용하는 서비스를 호출하여 테이블에 추가한다.
클라이언트는 이 모든 과정을 하나의 트랜잭션으로 지정하여 에러가 발생했을 경우 2개의 데이터베이스를 동시에 rollback할 수 있도록 한다.



2개의 데이터베이스 접속

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
demo.s	구조체 버퍼 설정 파일이다.
sample.m	Tmax 환경설정 파일이다.
mktable.sql	데이터베이스에 테이블로 생성하는 SQL 스크립트이다.

- 클라이언트 프로그램

프로그램 파일	설명
client.c	클라이언트 프로그램이다.

- 서버 프로그램

프로그램 파일	설명
update.pc	데이터베이스에 업데이트하는 서버 프로그램이다.
insert.pc	데이터베이스에 INSERT하는 서버 프로그램이다.
Makefile	Tmax에서 제공되는 Makefile을 수정해야 한다.

프로그램 특징

- 클라이언트 부분

기능	설명
Tmax 접속	기본 접속을 한다.
버퍼 유형	STRUCT이다.
통신 유형	tpcall()에 의한 동기 통신을 한다.

기능	설명
트랜잭션 처리	클라이언트에서 트랜잭션 범위를 지정한다.

- 서버 부분

기능	설명
서버 프로그램	서로 다른 데이터베이스를 사용하는 2개의 서버 프로그램이다.
서비스	UPDATE, INSERT이다.
데이터베이스 연결	2종류의 Oracle 데이터베이스이다.

구조체 버퍼

다음은 전역 트랜잭션에서 사용하는 구조체 버퍼이다.

<demo.s>

```
struct input {
       int account_id;
       int branch_id;
       char phone[15];
       char address[61];
};
```

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
*DOMAIN
res       SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60

*NODE
tmax1     TMAXDIR = "/user/ tmax ",
          APPDIR  = "/user/ tmax /appbin",
          PATHDIR = "/user/ tmax /path",
          TLOGDIR = "/user/ tmax /log/tlog",
          ULOGDIR="/user/ tmax /log/ulog",
          SLOGDIR="/user/ tmax /log/slog"

tmax2     TMAXDIR = "/user/ tmax ",
          APPDIR  = "/user/ tmax /appbin",
          PATHDIR = "/user/ tmax /path",
          TLOGDIR = "/user/ tmax /log/tlog",
          ULOGDIR="/user/ tmax /log/ulog",
          SLOGDIR="/user/ tmax /log/slog"


*SVRGROUP
svg1       NODENAME = tmax1, DBNAME = ORACLE,
           OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
```

```
           TMSNAME = svg1_tms

svg2       NODENAME = tmax2, DBNAME = ORACLE,
           OPENINFO = "ORACLE_XA+Acc=P/scott/tiger+SesTm=60",
           TMSNAME = svg2_tms

*SERVER
update       SVGNAME=svg1
insert       SVGNAME=svg2

*SERVICE
UPDATE       SVRNAME=update
INSERT       SVRNAME=insert
```

데이터베이스 스크립트

다음은 Oracle 테이블 생성 스크립트의 예제이다.

<mktable.sql>

```
sqlplus scott/tiger << EOF
drop table ACCOUNT;

create table ACCOUNT (
    ACCOUNT_ID  integer,
    BRANCH_ID   integer not null,
    SSN     char(13) not null,
    BALANCE     number,
    ACCT_TYPE   char(1),
    LAST_NAME   char(21),
    FIRST_NAME  char(21),
    MID_INIT    char(1),
    PHONE       char(15),
    ADDRESS     char(61),
    CONSTRAINT ACCOUNT_PK PRIMARY KEY(ACCOUNT_ID)
);
quit
EOF
```

클라이언트 프로그램

다음은 클라이언트 프로그램 예제이다.

<client.c >

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

#define TEMP_PHONE "6283-2114"
#define TEMP_ADDRESS "Korea"

int main(int argc, char *argv[])
{
```

```
struct input *sndbuf;
char *rcvbuf;
int acnt_id, n, timeout;
long len;

if (argc != 2) {
       fprintf(stderr, "Usage:%s account_id \n", argv[0]);
       exit(1);
}

acnt_id = atoi(argv[1]);
timeout = 5;

n = tmaxreadenv("tmax.env", "tmax");
if (n < 0) {
       fprintf(stderr, "tmaxreadenv fail! tperrno = %d\n", tperrno);
       exit(1);
}

n = tpstart((TPSTART_T *)NULL);
if (n < 0) {
       fprintf(stderr, "tpstart fail! tperrno = %s\n", tperrno);
       exit(1);
}
sndbuf = (struct input *)tpalloc("STRUCT", "input", sizeof(struct input));
if (sndbuf == NULL) {
       fprintf(stderr, "tpalloc fail: sndbuf tperrno = %d\n", tperrno);
       tpend();
       exit(1);
}
rcvbuf = (char *)tpalloc("STRING", NULL, 0);
if (rcvbuf == NULL) {
       fprintf(stderr, "tpalloc fail: rcvbuf tperrno = %d\n", tperrno);
       tpend();
       exit(1);
}
sndbuf->account_id = acnt_id;
sndbuf->branch_id = acnt_id;
strcpy(sndbuf ->phone, TEMP_PHONE);
strcpy(sndbuf ->address, TEMP_ADDRESS);

tx_set_transaction_timeout(timeout);
n = tx_begin();
if (n < 0)
       fprintf(stderr, "tx begin fail! tperrno = %d\n", tperrno);

n = tpcall("UPDATE", (char *)sndbuf, sizeof(struct input),
           (char **)&rcvbuf, (long *)&len, TPNOFLAGS);
if (n < 0) {
       fprintf(stderr, "tpcall fail! tperrno = %d\n", tperrno);
       tpend();
       exit(1);
}

n = tx_commit();
if (n < 0) {
       fprintf(stderr, "tx commit fail! tx error = %d \n", n);
       tx_rollback();
       tpend();
```

```
            exit(1);
        }
        printf("rtn msg = %s\n", rcvbuf);
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
}
```

서버 프로그램

다음은 데이터베이스에 업데이트하는 서버 프로그램의 예제이다.

<update.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/sdl.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
int account_id;
    int     branch_id;
    char    ssn[15];
    char    phone[15];
    char    address[61];
EXEC SQL END DECLARE SECTION;

UPDATE(TPSVCINFO *msg)
{
    struct input *rcvbuf;
    int     ret;
    long    acnt_id, rcvlen;
    char    *send;

    rcvbuf = (struct input *)(msg->data);
    send = (char *)tpalloc("STRING", NULL, 0);
    if (send == NULL) {
       fprintf(stderr, "tpalloc fail errno = %s\n", strerror(tperrno));
       tpreturn(TPFAIL, 0, (char *)NULL, 0, 0);
    }
    account_id = rcvbuf->account_id;
    branch_id = rcvbuf->branch_id;
    strcpy(phone, rcvbuf->phone);
    strcpy(address, rcvbuf->address);
    strcpy(ssn, "1234567");

    EXEC SQL UPDATE ACCOUNT
                SET BRANCH_ID = :branch_id,
                    PHONE = :phone,
                    ADDRESS = :address,
                    SSN = :ssn
              WHERE ACCOUNT_ID = :account_id;
    if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 ) {
```

```
        fprintf(stderr, "update failed sqlcode = %d\n", sqlca.sqlcode);
        tpreturn(TPFAIL, -1, (char *)NULL, 0, 0);
    }
    rcvbuf->account_id++;
    ret = tpcall("INSERT", (char *)rcvbuf, 0, (char **)&send, (long *)&rcvlen,
                TPNOFLAGS);
    if (ret < 0) {
        fprintf(stderr, "tpcall fail tperrno = %d\n", tperrno);
        tpreturn(TPFAIL, -1, (char *)NULL, 0, 0);
    }
    strcpy(send, OKMSG);
    tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}
```

다음은 데이터베이스에 INSERT하는 서버 프로그램의 예제이다.

<insert.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/sdl.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
    int     account_id;
    int     branch_id;
    char    ssn[15];
    char    phone[15];
    char    address[61];
EXEC SQL END DECLARE SECTION;

INSERT(msg)
TPSVCINFO  *msg;
{
    struct input *rcvbuf;
    int     ret;
    long    acnt_id;
    char    *send;

    rcvbuf = (struct input *)(msg->data);
    send = (char *)tpalloc("STRING", NULL, 0);
    if (send == NULL) {
        fprintf(stderr, "tpalloc fail errno = %s\n", tpstrerror(tperrno));
        tpreturn(TPFAIL, 0, (char *)NULL, 0, TPNOFLAGS);
    }

    account_id = rcvbuf->account_id;
    branch_id = rcvbuf->branch_id;
    strcpy(phone, rcvbuf->phone);
    strcpy(address, rcvbuf->address);
    strcpy(ssn, "1234567");

    /*  Declare && Open Cursor for Fetch */
```

```
    EXEC SQL INSERT INTO ACCOUNT (
                ACCOUNT_ID, BRANCH_ID, SSN, PHONE, ADDRESS)
              VALUES (
                :account_id, :branch_id, :ssn, :phone, :address);

    if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 )
    {
        printf("insert failed sqlcode = %d\n", sqlca.sqlcode);
        tpreturn(TPFAIL, -1, (char *)NULL, 0, TPNOFLAGS);
    }
    strcpy(send, OKMSG);
    tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}
```

5.3. 데이터베이스 프로그램

대표적인 데이터베이스인 Oracle과 Informix를 사용하는 몇 가지 예제를 제시한다.

5.3.1. Oracle Insert 프로그램

클라이언트는 사용자의 입력을 받아 구조체 버퍼에 넣어 서비스를 호출하며 서버는 이를 받아 해당 테이블에 추가한다. 클라이언트는 트랜잭션을 지정하여 에러가 발생하였을 경우 rollback할 수 있도록 한다.

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
demo.s	구조체 버퍼 설정 파일이다.
sample.m	Tmax 환경설정 파일이다.
mktable.sql	데이터베이스 테이블을 생성하는 SQL 스크립트이다.
sel.sql	데이터베이스 테이블 내용을 출력하는 SQL 스크립트이다.

- 클라이언트 프로그램

프로그램 파일	설명
oins_cli.c	클라이언트 프로그램이다.

- 서버 프로그램

프로그램 파일	설명
oinssvc.pc	서비스 프로그램의 Oracle 소스이다.
Makefile	Tmax에서 제공되는 Makefile을 수정해야 한다.

프로그램 특징

- 클라이언트 프로그램

기능	설명
Tmax 접속	기본 접속을 한다.
버퍼 유형	STRUCT이다.
통신 유형	tpcall()에 의한 동기 통신을 한다.
트랜잭션 처리	클라이언트에서 트랜잭션 범위를 지정한다.

- 서버 프로그램

구분	설명
서비스	ORAINS이다.
데이터베이스 연결	Oracle 데이터베이스이다.

구조체 버퍼

다음은 구조체 버퍼의 예제이다.

<demo.s>

```
struct ktran {
    int no;
    char name[20];
};
```

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
*DOMAIN
resrc             SHMKEY = 77990, MAXUSER = 256

*NODE
tmax              TMAXDIR = /home/tmax,
                  APPDIR = /home/tmax/appbin,
                  PATHDIR = /home/tmax/path

*SVRGROUP
svg1              NODENAME = tmax,
                  DBNAME = ORACLE,
                  OPENINFO = “Oracle_XA+Acc=P/scott/tiger+SesTm=60”,
                  TMSNAME = svg1_tms

*SERVER
oinssvc           SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
```

```
ORAINS          SVRNAME = oinssvc
```

다음은 환경설정에 추가된 항목에 대한 설명이다.

항목	설명
DBNAME	사용하는 데이터베이스를 정의한다.
OPENINFO	Oracle 데이터베이스 연결 정보를 설정하는 항목으로 Oracle 데이터베이스인 경우 CLOSEINFO는 지정하지 않아도 된다. tpsvrinfo()에서 호출한다.
TMSNAME	트랜잭션 처리를 주재하는 프로세스명을 지정하는 항목으로 OPENINFO 지정으로 인한 자동 트랜잭션을 처리한다. svg1에 속한 해당 서비스를 자동 트랜잭션 상태에서 처리한다.

데이터베이스 스크립트

다음은 Oracle 테이블 생성 스크립트의 예제이다.

<mktable.sql>

```
sqlplus scott/tiger << EOF
       create table testdb1 (
       no number(7),
       name char(30)
) ;
EOF
```

다음은 Oracle 테이블 및 데이터 출력 스크립트의 예제이다.

<sel.sql>

```
sqlpus scott/tiger << EOF
desc testdb1;
select * from testdb1;
select count (*) from testdb1;
EOF
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<oins_cli.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include “../sdl/demo.s”

main(int argc, char *argv[])
{
       struct ktran *sndbuf, *rcvbuf;
       long sndlen, rcvlen;
```

```
        int cd;

        if (argc != 3) {
        printf(“Usage: client no name\n”);
               exit(1);
        }

        printf(“tpstart-start \n”);
        if(tpstart ((TPSTART_T *) NULL) == -1) {
               printf(“Tpstart failed\n”);
               exit(1);
        }
        printf(“tpstart-ok \n”);

        if((sndbuf=(struct ktran *) tpalloc("STRUCT","ktran",0))==NULL) {
             printf(“tpalloc failed:sndbuf, tperrno=%d\n”, tperrno);
             tpend();
             exit(1) ;
         }

         if((rcvbuf = (struct ktran *) tpalloc("STRUCT", "ktran", 0))== NULL) {
              printf(“tpalloc failed:rcvbuf, tperrno=%d\n”, tperrno);
              tpfree((char *)sndbuf);
              tpend();
              exit(1);
         }

        sndbuf->no = atoi(argv[1]);
        strcpy(sndbuf->name, argv[2]);
        printf(“tpcall-start \n”);
        tx_begin();

        if(tpcall("ORAINS",(char *)sndbuf,0,(char **)&rcvbuf,&rcvlen,TPNOFLAGS)==-1)
        {
             printf(“tpcall failed:ORA service, tperrno=%d”, tperrno);
             printf(“sql code=%d\n”, tpurcode);
             tx_rollback();
             tpfree ((char *)sndbuf);
             tpfree ((char *)rcvbuf);
             tpend();
             exit(1);
        }

        printf(“tpcall-success \n”);
        tx_commit();

        tpfree ((char *)sndbuf);
        tpfree ((char *)rcvbuf);
        tpend();
}
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

<oinssvc.pc>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include “../sdl/demo.s”

EXEC SQL begin declare section;
char name[20];
int no;

EXEC SQL end declare section;
EXEC SQL include sqlca;

ORAINS(TPSVCINFO *msg)
{
       struct ktran *stdata;
        stdata = (struct ktran *)msg->data;
        strcpy(name, stdata->name);
        no = stdata->no;
        printf(“Ora service started\n”);

       /* 데이터베이스에 삽입 */
       EXEC SQL insert into testdb1(no, name) values(:no, :name);

       if (sqlca.sqlcode != 0){
             printf(“oracle sqlerror=%s”,sqlca.sqlerrm.sqlerrmc);
             tpreturn (TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
        }

       tpreturn (TPSUCCESS, sqlca.sqlcode, stdata, 0, TPNOFLAGS);
}
```

5.3.2. Oracle Select 프로그램

클라이언트는 사용자의 입력을 받아 구조체 버퍼에 넣어 서비스를 호출하며 서버는 이에 해당하는 모든 데이터를 받아 구조체 배열을 사용하여 결과를 반환한다. 클라이언트는 트랜잭션을 지정하여 에러가 발생하였을 경우 rollback할 수 있도록 한다.

프로그램 구성

- 공통 프로그램

구성	설명
demo.s	구조체 버퍼 설정 파일이다.
sample.m	Tmax 시스템 환경 파일이다.
mktable.sql	데이터베이스에 테이블을 생성하는 SQL 스크립트이다.
sel.sql	데이터베이스 테이블 내용을 출력하는 SQL 스크립트이다.

- 클라이언트 프로그램

구성	설명
oins_cli.c	클라이언트 프로그램이다.
cdata.c	클라이언트 사용 함수 모듈이다.

- 서버 프로그램

구성	설명
oselsvc.pc	서비스 프로그램 Oracle 소스이다.
Makefile	Tmax에서 제공되는 Makefile로 수정해야 한다.

프로그램 특징

- 클라이언트 프로그램

기능	설명
Tmax 접속	기본 접속을 한다.
서비스	ORASEL이다.
버퍼 유형	STRUCT이다.
통신 유형	tpcall()에 의한 동기 통신을 한다.
트랜잭션 처리	클라이언트에서 트랜잭션 범위를 지정한다.

- 서버 프로그램

기능	설명
서비스	ORASEL이다.
데이터베이스 연결	Oracle 데이터베이스와 연결한다.
버퍼사용	필요에 의해 버퍼 크기를 재조정한다.

구조체 버퍼

다음은 구조체 버퍼의 예제이다.

<demo.s>

```
struct stru_his{
       long    ACCOUNT_ID ;
       long    TELLER_ID ;
       long    BRANCH_ID ;
       long    AMOUNT ;
} ;
```

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
*DOMAIN
resrc             SHMKEY = 77990, MAXUSER = 256

*NODE
tmax              TMAXDIR =“/home/tmax”,
                  APPDIR =“/home/tmax/appbin”,
                  PATHDIR =“/home/tmax/path”

*SVRGROUP
svg1              NODENAME = tmax,
                  DBNAME = ORACLE,
                  OPENINFO = “Oracle_XA+Acc=P/scott/tiger+SesTm=600”,
                  TMSNAME = svg1_tms

*SERVER
oselsvc           SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
ORASEL            SVRNAME = oselsvc
```

다음은 환경설정에 추가된 항목에 대한 설명이다.

항목	설명
DBNAME	사용할 데이터베이스명을 설정한다.
OPENINFO	Oracle 데이터베이스 연결 정보를 설정하는 항목으로 Oracle에서는 CLOSEINFO는 설정하지 않아도 된다. 다음은 OPENINFO 항목에서 사용할 수 있는 옵션에 대한 설명이다. ◦ LogDir : LogDir을 지정하면 지정된 곳에 XA에 관련된 로그를 남길 수 있다. LogDir을 지정하지 않으면 $ORACLE_HOME/rdbms/log 또는 현재 디렉터리에 <xa_NULL날짜.trc> 파일이 생성된다. ◦ DbgFl : 몇 번째 단계의 디버그 flags를 설정할지를 결정한다. 기본적인 단계인 0x01 또는 OCI 단계인 0x04 등을 사용하면 된다.
TMSNAME	트랜잭션 처리를 관리하는 프로세스명을 지정한다.
AUTOTRAN	해당 서비스를 처리할 때 자동으로 트랜잭션 상태로 처리한다.

다음은 OPENINFO 항목의 옵션으로 LogDir과 DbgFl을 사용하는 예제이다.

```
OPENINFO=“Oracle_XA+Acc=P/계정/암호 +SesTm=60+LogDir=/tmp+DbgFl=0x01”
```

개발할 때에는 디버그 모드를 해제하고 사용해야만 나중에 디스크가 Full되는 상황을 피할 수 있다.

데이터베이스 스크립트

다음은 Oracle 테이블 작성 스크립트의 예제이다.

<mktable.sql>

```
sqlplus scott/tiger << EOF
       create table sel_his(
       account_id number(6),
       teller_id number(6),
       branch_id number(6),
       amount number(6)
);
create unique index idx_tdb1 on sel_his(account_id);
EOF
```

다음은 Oracle 테이블 및 데이터 출력 스크립트의 예제이다.

<sel.sql>

```
sqlplus scott/tiger << EOF
desc sel_his;
select * from sel_his;
EOF
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<oins_cli.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/tx.h>
#include "../sdl/demo.s"
#define NARRAY 10
#define NOTFOUND 1403

main(int argc,char *argv[])
{
       struct stru_his *transf;
       int i, j;
       long urcode, nrecv, narray = NARRAY;
       long account_id, teller_id, branch_id, amount;

       if (argc != 2) {
              fprintf(stderr,"Usage:$%s ACOUNT_ID !\n", argv[0]);
              exit(0);
       }

       if (tpstart((TPSTART_T *) NULL) == -1) {    /* Tmax에 접속 */
           fprintf(stderr,"TPSTART_T(tpinfo) failed -> %s!\n",
           tpstrerror(tperrno)) ;
```

```
        exit(1) ;
    }

    /* c structure 구조로 버퍼 생성 */
    transf = (struct stru_his *) tpalloc("STRUCT", "stru_his",0);
    if (transf == (struct stru_his *)NULL) {
           fprintf(stderr,"Tpalloc failed->%s!\n",
           tpstrerror(tperrno)) ;
           tpend();
           exit(1);
    }
    memset(transf, 0x00, sizeof(struct stru_his));

    account_id = atoi(argv[1]);
    transf->ACCOUNT_ID = account_id;

    /* 트랜잭션 타임아웃 설정 */
    tx_set_transaction_timeout(30);

    /* 글로벌 트랜잭션 시작 */
    if (tx_begin() < 0) {
           fprintf(stderr, "tx_begin() failed ->%s!\n",
           tpstrerror(tperrno));
           tpfree((char*)transf);
           tpend();
           exit(0) ;
    }

    if (tpcall("ORASEL",(char *)transf, 0, (char **)&transf, &nrecv,
               TPNOFLAGS)== -1){
           /* 동기 통신으로 "ORASEL" 서비스 요청 */
           fprintf(stderr,"Tpcall(SELECT...)error->%s ! ",
           tpstrerror(tperrno)) ;
           tpfree((char *)transf);
           /* 실패시 트랜잭션 취소 */
           tx_rollback();
           tpend();
           exit(0) ;
    }
    /* 성공시 트랜잭션 Commit */
    if (tx_commit() == -1) {
           fprintf(stderr, "tx_commit() failed ->%s!\n",
           tpstrerror(tperrno)) ;
           tpfree((char *)transf);
                  tpend();
                  exit(0) ;
    }
     /* 받아온 데이터는 구조체의 배열이다. */
    for (j =0 ; j < tpurcode ; j++) {
        /* Oracle에서 셀렉트한 데이터 결과를 프린트 */
       if (j == 0)
           printf("%-12s%-10s%-10s%-10s\n",
                "ACCOUNT_ID","TELLER_ID", "BRANCH_ID", "AMOUNT");
           account_id=transf[j].ACCOUNT_ID;
           teller_id=transf[j].TELLER_ID;
           branch_id=(*(transf+j)).BRANCH_ID;
           amount=transf[j].AMOUNT;
           printf("%-12d %-10d %-10d %-10d\n", account_id,
                    teller_id,branch_id, amount);
```

```
        }
        /* 셀렉트한 데이터가 없거나 끝이라면 */
        if (urcode == NOTFOUND) {
        printf("No records selected!\n");
                tpfree((char *)transf);
                tpend();
                return 0;
}
        tpfree((char *)transf);
        tpend();
}
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

<oselsvc.pc>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"
#define NARRAY 10
#define TOOMANY 2112
#define NOTFOUND 1403
EXEC SQL include sqlca.h;

EXEC SQL begin declare section;
        long key, rowno= NARRAY;
        long account_id[NARRAY],teller_id[NARRAY], branch_id[NARRAY],
        amount[NARRAY] ;
EXEC SQL end declare section;

ORASEL(TPSVCINFO *msg)
{
        struct stru_his *transf;
        int i , lastno;
        transf=(struct stru_his *) msg->data;

        /* msg 버퍼의  내용을  프로그램 변수에 전달. */
        key = transf->ACCOUNT_ID;

         /* transf 버퍼의 크기를 재조정 */
        if((transf=(struct stru_his *) tprealloc((char*)transf,
            sizeof(struct stru_his) * NARRAY ))==(struct stru_his*)NULL){
               fprintf(stderr, "tprealloc error ->%s\n",
               tpstrerror(tperrno));
               tpreturn(TPFAIL, tperrno, NULL, 0, TPNOFLAGS);
        }
        EXEC SQL select account_id, teller_id, branch_id, amount
                    into :account_id, :teller_id, :branch_id, :amount  from sel_his
                   where account_id > :key
               /* account_id가 client에서 보낸 key값 보다 큰 것을 */
               order by account_id;       /* global 변수에 넣어줌 */

        /* sql error 체크 (셀렉트된 것이 없거나 너무 많은 것 제외.) */
        if (sqlca.sqlcode!=0 && sqlca.sqlcode!=NOTFOUND && sqlca.sqlcode!=TOOMANY) {
               fprintf(stderr,"SQL ERROR ->NO(%d):%s\n", sqlca.sqlcode,
```

```
                sqlca.sqlerrm.sqlerrmc) ;
                tpreturn(TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
        }

        /* access한 개수를 lastno에 넣어줌 */
        lastno = sqlca.sqlerrd[2];

        /* 셀렉트된 데이터가 너무 많음 */
        if (sqlca.sqlcode == TOOMANY)
                lastno =rowno;

        /* No records */
        if (lastno == 0)
                transf->ACCOUNT_ID = 0;

         /* Oracle에서 셀렉트한 데이터를 전송할 버퍼에 넣어줌 */
         for ( i = 0 ; i < lastno; i++) {
                transf[i].ACCOUNT_ID = account_id[i];
                transf[i].TELLER_ID = teller_id[i];
                transf[i].BRANCH_ID = branch_id[i];
                transf[i].AMOUNT = amount[i];
        }
        tpreturn(TPSUCCESS, lastno, transf, i * sizeof(struct stru_his),
        TPNOFLAGS );
}
```

5.3.3. Informix Insert 프로그램

클라이언트는 사용자의 입력을 받아 구조체 버퍼에 넣어 서비스를 호출하며 서버는 이를 받아 해당 테이블에 추가한다. 클라이언트는 트랜잭션을 지정하여 에러가 발생한 경우 Rollback한다.

Informix 애플리케이션을 컴파일하기 전 다음 사항을 먼저 확인한다.

1. UNIX 환경을 확인한다(.profile,.login, .cshrc).

   다음과 같이 설정한다.

   ```
   INFORMIXDIR=/home/informix
   INFORMIXSERVER=tmax
   ONCONFIG=onconfig
   PATH=$INFORMIXDIR/bin: …
   LD_LIBRARY_PATH=/home/informix/lib:/home/informix/lib/esql:
   …
   ```

2. Makefile을 확인한다.

   다음 오퍼레이션과 설정을 확인한다.

   ```
   # Server esql makefile

   TARGET  = <target filename>
   APOBJS  = $(TARGET).o
   ```

```
SDLFILE = info.s

LIBS    = -lsvr -linfs
# solaris의 경우 -lnsl -lsocket 추가

OBJS    = $(APOBJS) $(SDLOBJ) $(SVCTOBJ)
SDLOBJ  = ${SDLFILE:.s=_sdl.o}
SDLC    = ${SDLFILE:.s=_sdl.c}
SVCTOBJ = $(TARGET)_svctab.o

CFLAGS =-O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# Solaris 32bit, Compaq, Linux: CFLAGS = -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# Solaris 64bit: CFLAGS = -xarch=v9 -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# HP 32bit: CFLAGS = -Ae -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# HP 64bit: CFLAGS = -Ae +DA2.0W +DD64 +DS2.0 -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# IBM 32bit: CFLAGS = -q32 -brtl -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# IBM 64bit: CFLAGS = -q64 -brtl -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)

INFLIBD = $(INFORMIXDIR)/lib/esql
INFLIBDD = $(INFORMIXDIR)/lib
INFLIBS = -lifsql -lifasf -lifgen -lifos -lifgls -lm -ldl -lcrypt
          $(INFORMIXDIR)/lib/esql/
checkapi.o -lifglx -lifxa

APPDIR  = $(TMAXDIR)/appbin
SVCTDIR = $(TMAXDIR)/svct
TMAXLIBDIR  = $(TMAXDIR)/lib

#
.SUFFIXES : .ec .s .c .o

.ec.c :
        esql -e $*.ec

#
# server compile
#
all: $(TARGET)

$(TARGET):$(OBJS)
        $(CC) $(CFLAGS) -L$(TMAXLIBDIR) -L$(INFLIBD) -L$(INFLIBDD) -o $(TARGET)
        $(OBJS) $(LIBS) $(INFLIBS)
        mv $(TARGET) $(APPDIR)/.
        rm -f $(OBJS)

$(APOBJS): $(TARGET).ec
        esql -e -I$(TMAXDIR)/usrinc $(TARGET).ec
        $(CC) $(CFLAGS) -c $(TARGET).c

$(SVCTOBJ):
        touch $(SVCTDIR)/$(TARGET)_svctab.c
        $(CC) $(CFLAGS) -c $(SVCTDIR)/$(TARGET)_svctab.c

$(SDLOBJ):
        $(TMAXDIR)/bin/sdlc -i ../sdl/$(SDLFILE)
        $(CC) $(CFLAGS) -c ../sdl/$(SDLC)

#
clean:
```

```
-rm -f *.o core $(TARGET) $(TARGET).lis
```

<TMS Makefile>

```
#
TARGET  = info_tms

INFOLIBDIR = ${INFORMIXDIR}/lib
INFOELIBDIR = ${INFORMIXDIR}/esql
INFOLIBD = ${INFORMIXDIR}/lib/esql
INFOLIBS = -lifsql -lifasf -lifgen -lifos -lifgls -lm -ldl -lcrypt
           /opt/informix/lib/esql/checkapi.o
 -lifglx -lifxa
# solaris는 라이브러리에 -lnsl -lsocket -laio -lelf 추가

CFLAGS =-O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# Solaris 32bit, Compaq, Linux: CFLAGS = -O -I$(INFORMIXDIR)/incl/esql
                                        -I$(TMAXDIR)
# Solaris 64bit: CFLAGS = -xarch=v9 -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# HP 32bit: CFLAGS = -Ae -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# HP 64bit: CFLAGS = -Ae +DA2.0W +DD64 +DS2.0 -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR)
# IBM 32bit: CFLAGS = -q32 -brtl -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR
# IBM 64bit: CFLAGS = -q64 -brtl -O -I$(INFORMIXDIR)/incl/esql -I$(TMAXDIR

TMAXLIBDIR = $(TMAXDIR)/lib
TMAXLIBS  = -ltms -linfs
# CC = /opt/SUNWspro/bin/cc : solaris only


$(TARGET): $(APOBJ)
        $(CC) $(CFLAGS) -o $(TARGET) -L$(TMAXLIBDIR) -L$(INFOLIBD)
        -L$(INFOLIBDIR) -L
$(INFOELIBDIR) $(INFOLIBS) $(TMAXLIBS)
        mv $(TARGET) $(TMAXDIR)/appbin

#
clean:
        -rm -f  *.o core $(TARGET)
```

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
demo.s	구조체 버퍼 설정 파일이다.
sample.m	Tmax 시스템 환경 파일이다.
mkdb.sql	데이터베이스 생성 SQL 스크립트로 XA 모드는 데이터베이스가 로깅 모드에서 생성될 때 지원한다.
mktable.sql	데이터베이스에 테이블을 생성하는 SQL 스크립트이다.
sel.sql	데이터베이스 테이블에 내용을 출력하는 SQL 스크립트이다.
info.s	SDLFILE이다.

- 클라이언트 프로그램

프로그램 파일	설명
client.c	클라이언트 프로그램이다.

- 서버 프로그램

구성	설명
tdbsvr.ec	서버 프로그램이다.
Makefile	Tmax에서 제공되는 Makefile을 수정한다.

프로그램 특징

- 클라이언트 프로그램

기능	설명
Tmax 접속	기본 접속을 한다.
버퍼 유형	STRUCT이다.
통신 유형	tpcall()에 의한 동기 통신을 한다.
트랜잭션 처리	클라이언트에서 트랜잭션 범위를 지정한다.

- 서버 프로그램

기능	설명
서비스	INSERT를 한다.
데이터베이스 연결	Informix 데이터베이스이다.

구조체 버퍼

다음은 구조체 버퍼의 예제이다.

<demo.s>

```
struct info {
     char seq[8];
     char data01[128];
     char data02[128];
     char data03[128];
};
```

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
*DOMAIN
resrc            SHMKEY = 77990, MAXUSER = 256

*NODE
tmax             TMAXDIR =”/home/tmax”, A
                 APPDIR =”/home/tmax/appbin”,
                 PATHDIR =”/home/tmax/path”

*SVRGROUP
svg1             NODENAME = tmax,
                 DBNAME = INFORMIX,
                 OPENINFO = “stores7”,
                 CLOSEINFO = “”,
                 TMSNAME = info_tms

*SERVER
tdbsvr          SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
INSERT         SVRNAME = tdbsvr
```

다음은 환경 설정에 추가된 항목에 대한 설명이다.

항목	설명
DBNAME	사용할 데이터베이스명을 설정한다.
OPENINFO, CLOSEINFO	Informix 데이터베이스 연결 및 해제를 위한 정보를 tpsvrinfo(), tpsvrdone()에서 호출한다.
TMSNAME	트랜잭션을 처리하는 프로세스명을 지정하는 항목으로 OPENINFO 지정으로 인한 자동 트랜잭션 처리 svg1에 속한 해당 서비스를 자동 트랜잭션 상태에서 처리한다.

데이터베이스 스크립트

다음은 Informix 테이블 생성 스크립트의 예제이다.

<mktable.sql>

```
dbaccess << EOF
create database stores7 with buffered log;
grant connect to public;

database stores7;
drop table testdb1;
create table testdb1 (
        seq        VARCHAR(8) ,
        data01     VARCHAR(120) ,
        data02     VARCHAR(120) ,
        data03     VARCHAR(120)
) lock mode row;

create unique index idx_tdb1 on testdb1(seq);
```

```
EOF
```

다음은 Informix 테이블 및 데이터 출력 스크립트의 예제이다.

<sel.sql>

```
dbaccess << EOF
database stores7;
select * from testdb1;
EOF
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<client.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/tx.h>
#include “info.s”
main(int argc, char **argv)
{
       struct info *transf;
       char data[256];
       long nrecv;

       /* Tmax에 연결. */
       if ((tpstart((TPSTART_T *)NULL) == -1) {
            fprintf(stderr, “tpstart(TPINFO...) failed ->%s!\n”,
            tpstrerror(tperrno)) ;
            exit(1) ;
       }

      /* 응용 프로그램에서 사용할 버퍼 메모리 할당. */
     if ((transf=(struct info *)tpalloc ("STRUCT","info",0))==(struct info *)NULL){
          fprintf(stderr, “tpalloc(struct info, ...) failed ->%s!\n”,
          tpstrerror(tperrno)) ;
          tpend();
          exit(1) ;
       }

       /* 전송할 데이터 필드를 채움. */
       strcpy(transf->seq, “000001”);
       strcpy(transf->data01, “Hello”);
       strcpy(transf->data02, “World”);
       strcpy(transf->data03, “1234”);

        /* 트랜잭션 timeout 설정 * /
       tx_set_transaction_timeout (30);

       /* 트랜잭션 시작을 알림. */
       if (tx_begin() < 0) {
              fprintf(stderr, “tx_begin() failed ->%s!\n”,
```

```
              tpstrerror(tperrno)) ;
              tpfree ((char *)transf);
              tpend( ); exit(0);
       }

       /* 서비스 호출 */
     if (tpcall("INSERT",(char*) transf,0,(char **)&transf,&nrecv,TPNOFLAGS)==-1){
         fprintf(stderr,"tpcall(struct info, ...)
         failed ->%s!\n",tpstrerror(tperrno)) ;
         tx_rollback ();
         tpfree ((char *)transf),
         tpend();
          exit(0);
       }

      /* 트랜잭션 성공. */
     if (tx_commit () < 0) {
          fprintf(stderr, “tx_commit() failed ->%s!\n”, tpstrerror(tperrno)) ;
          tpfree ((char *)transf);
          tpend( );
          exit(0);
       }

       tpfree ((char *)transf );
       /* Tmax 연결을 끊음. */
       tpend( ) ;
}
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

< tdbsvr.ec>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include “info.s”

EXEC SQL include sqlca.h;

/* 서비스 명*/
INSERT(TPSVCINFO *msg)
{
      /* 프로그램상의 버퍼 형식 선언. */
      struct info *INFO;

       /* SQL문 내의 버퍼 형식 선언. */
       EXEC SQL begin declare section;
       varchar seq[8],buf01[128],buf02[128],buf03[128];
       EXEC SQL end declare section;

       /* 메시지 버퍼로부터 구조체형으로 데이터를 전달받는다. */
       INFO = (struct info *)msg -> data;

       /* 구조체 형식으로 전달받은 데이터를 데이터베이스 버퍼로 복사. */
       strcpy(seq, INFO->seq);
```

```
        strcpy(buf01, INFO->data01);
        strcpy(buf02, INFO->data02);
        strcpy(buf03, INFO->data03);

       /* SQL문의 insert 수행 */
        EXEC SQL insert into testdb1 (seq,data01,data02,data03)
        values(:seq, :buf01, :buf02, :buf03);

       /* 에러가 발생한다면 */
        if ( sqlca.sqlcode ! = 0)  {
              /* Insert 실패를 알린다. */
               printf("SQL error => %d !" ,sqlca.sqlcode);
               tpreturn (TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
        }

        /* Insert가 성공적으로 끝났을 때 */
        tpreturn (TPSUCCESS, 0, NULL, 0, TPNOFLAGS);
}
```

5.3.4. Informix Select 프로그램

클라이언트는 사용자의 입력을 받아 구조체 버퍼에 넣어 서비스를 호출하며 서버는 이에 해당하는 모든 데이터를 받아 구조체 배열을 사용하여 결과를 반환한다. 클라이언트는 트랜잭션을 지정하여 에러가 발생하였을 경우 rollback할 수 있도록 한다.

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
acct.s	SDLFILE이다.
sample.m	Tmax 환경 파일이다.
mkdb.sql	데이터베이스를 생성하는 SQL 스크립트이다.
mktables.sql	데이터베이스에 테이블을 생성하는 SQL 스크립트이다.
sel.sql	데이터베이스 테이블에 내용을 출력하는 SQL 스크립트이다.

- 클라이언트 프로그램

구성	설명
client.c	클라이언트 프로그램이다.
cdate.c	client.c에서 사용된 함수 모듈이다.

- 서버 프로그램

구성	설명
sel_acct.ec	서비스 프로그램 Informix 소스이다.

구성	설명
Makefile	Tmax에서 제공되는 Makefile을 수정한다.

프로그램 특징

- 클라이언트 프로그램

구분	설명
Tmax 접속	기본 접속을 한다.
버퍼 유형	STRUCT이다.
통신 유형	tpcall()에 의한 동기 통신을 한다.
트랜잭션 처리	클라이언트에서 트랜잭션 범위를 지정한다.

- 서버 프로그램

구분	설명
서비스	SEL_ACCT이다.
데이터베이스 연결	Informix 데이터베이스이다.
버퍼 사용	필요에 의해 버퍼 크기 재조정한다.

구조체 버퍼

다음은 구조체 버퍼의 예제이다.

<acct.s>

```
struct stru_acct {
    int  ACCOUNT_ID;
    char PHONE[20];
    char ADDRESS[80];
};
```

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
*DOMAIN
resrc         SHMKEY = 77990, MAXUSER = 256

*NODE
tmax          TMAXDIR =”/home/tmax”,
              APPDIR =”/home/tmax/appbin”,
              PATHDIR =”/home/tmax/path”

*SVRGROUP
```

```
svg1          NODENAME = tmax,
              DBNAME = INFORMIX,
              OPENINFO = “stores7”,
              CLOSEINFO = “”,
              TMSNAME = info_tms
*SERVER
SEL_ACCT      SVGNAME = svg1, MIN = 1, MAX = 5

*SERVICE
SEL_ACCT      SVRNAME = sel_acct
```

다음은 환경 설정에 추가된 항목에 대한 설명이다.

항목	설명
DBNAME	사용할 데이터베이스를 설정한다.
OPENINFO, CLOSEINFO	Informix 데이터베이스에 연결 및 해제를 위한 정보를 tpsvrinit(), tpsvrdone()에서 호출한다.
TMSNAME	트랜잭션을 처리하는 프로세스명을 지정하는 항목으로 OPENINFO 지정으로 인한 자동 트랜잭션을 처리한다. svg1에 속한 해당 서비스를 자동 트랜잭션 상태에서 처리한다.

데이터베이스 스크립트

다음은 Informix 테이블 생성 스크립트의 예제이다.

<mkdb.sql>

```
dbaccess << EOF
create database stores7 with buffered log;
grant connect to public;

database stores7;
drop table ACCOUNT;

create table ACCOUNT (
       account_id INTEGER,
       phone VARCHAR(20),
       address VARCHAR(80)
) lock mode row;

create unique index idx_tdb1 on ACCOUNT(account_id);
EOF
```

다음은 Informix 테이블 및 데이터 출력 스크립트의 예제이다.

<sel.sql>

```
dbaccess << EOF
database stores7;
select * from ACCOUNT;
```

```
EOF
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<client.c>

```
#include <stdio.h>
#include <time.h>
#include <usrinc/atmi.h>
#include <usrinc/tx.h>
#include “acct.s”
#define NOTFOUND 1403

void htime(char *, int *);

main(int argc, char *argv[ ])
{
       struct stru_acct *transf;
       float tps;
       int i, j, loop, cnt_data = 0, sec1, sec2;
       long urcode, nrecv, narray;
       char ts[30], te[30], phone[20], address[80];
       int account_id, key;

       if(argc != 2) {
          fprintf(stderr,“Usage:$%sLOOP (NARRAY = 30) !\n”, argv[0]);
          exit(0) ;
       }

       /*user가 원하는 수행 횟수만큼 루프를 돈다.*/
       loop = atoi(argv[1]);

        /* Tmax에 접속 */
       if (tpstart((TPSTART_T *)NULL) == -1) {
           fprintf(stderr, “tpstart(tpinfo) failed ->%s!\n”,
           tpstrerror(tperrno));
           exit(1) ;
       }

      /* sec1 = 시작시간 */
       htime(ts,&sec1); key=0;

      /* message 버퍼 할당 */
     for( i = 0; i < loop; i++) {
         if ((transf=(struct stru_acct *)tpalloc("STRUCT","stru_acct",0))
                ==(struct stru_acct *)NULL) {
               fprintf(stderr,“Tpalloc(STRUCT.. )failed->%s!\n” ,
               tpstrerror(tperrno)) ;
                tpend(); exit(1);
              }
         transf -> ACCOUNT_ID = key;

         /* time-out value= 30 */
         tx_set_transaction_timeout(30) ;
```

```
  if (tx_begin() < 0) { /* 트랜잭션 시작 */
       fprintf(stderr, “tx_begin() failed ->%s!\n”, tpstrerror(tperrno)) ;
       tpfree((char*)transf);
       tpend();
       exit(0);
  }

 /* select service 호출 */
 if (tpcall(“SEL_ACCT”, (char *)transf, 0, (char **)&transf, &nrecv,
     TPNOFLAGS)== -1) {
     /* service error : message buffer free & 트랜잭션 취소 & 접속 끊음 */
      fprintf(stderr,“Tpcall(SELECT...)error->%s! ” ,
                tpstrerror(tperrno)) ;
      tpfree ((char *)transf);
      tx_rollback ( ) ;
      tpend( ) ;
      exit ( 1 ) ;
 }
 urcode = tpurcode;

/*성공적인 서비스 완결 : 트랜잭션 결과로 실제 리소스 변화시킴*/
 if (tx_commit() < 0 ) {
        fprintf(stderr, “tx_commit() failed ->%s!\n”, tpstrerror(tperrno)) ;
        tpfree((char *)transf);
        tpend();
        exit(0);
 }

/*데이터가 선택된 경우*/
if ( urcode != NOTFOUND) {
     narray =urcode;
     /* select한 데이터의 마지막 Record */
     key=transf[narray-1].ACCOUNT_ID;
    /* selsct한 개수 만큼 결과를 user에게 출력 */
    for ( j = 0 ; j < narray ; j++ ) {
         if ( j == 0)
             printf(“%-10s%-14s%s\n”, “ACCOUNT_ID”, “PHONE”,“ADDRESS”) ;
             account_id = transf[j].ACCOUNT_ID;
             strcpy(phone, transf[j].PHONE);
             strcpy(address, transf[j].ADDRESS);
             printf(“%-10d %-14s %s\n”, account_id, phone, address);
      }/* for2 end */

     /* 전체 결과 개수 증가 */
     cnt_data += j;

     /* message buffer free */
     tpfree ((char *)transf);
      if(urcode == NOTFOUND) {
         printf(“No records selected!\n”);
         break ;
       }
 }/* for1 end */

 /* message buffer free */
 tpfree ((char *)transf);
 /* Tmax 연결을 끊음. */
 tpend ();
```

```
        /* sec2 = 끝 시간 */
        htime(te,&sec2);

        /* 데이터 하나당 처리시간 계산. */
        printf("TOT.COUNT = %d\n", cnt_data);
        printf("Start time = %s\n", ts);
        printf("End time = %s\n", te);
       if ((sec2-sec1) ! = 0)
            tps = (float) cnt_data / (sec2 - sec1);
        else
            tps = cnt_data;
        printf("Interval = %d secs ==> %10.2f T/S\n", sec2-sec1,tps);
}

htime(char *cdate, int *sec)
{
        long time(), timef, pt;
         char ct[20], *ap;
         struct tm *localtime(), *tmp;

         pt = time(&timef);
         *sec = pt;
         tmp = localtime(&timef);
         ap = asctime(tmp);

         sscanf(ap, "%*s%*s%*s%s",ct);
         sprintf( cdate, "%02d. %02d. %02d %s", tmp->tm_year, ++tmp->tm_mon,
                  tmp->tm_mday,ct);
}
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

<sel_acct.pc>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/tx.h>
#include "acct.s"
#define NFETCH 5
#define NOTFOUND 100

EXEC SQL include sqlca.h;
EXEC SQL begin declare section;
long account_id, key;
varchar phone[20], address[80];
EXEC SQL end declare section;

SEL_ACCT(TPSVCINFO *msg)
{
        int i , j , nfetch;
        int return_code;
        struct stru_acct *ACCT_V;

        /*클라이언트 데이터를 받는다. */
        ACCT_V = (struct stru_acct *) msg->data;
```

```
/* select원하는 account id 값을 key에 옮김. */
key = ACCT_V->ACCOUNT_ID;

/* 클라이언트 버퍼 사이즈 재할당*/
if ((ACCT_V = (struct stru_acct *)tprealloc((char *)ACCT_V,
    sizeof(struct stru_acct)*NFETCH )) == (struct stru_acct *)NULL) {
    fprintf(stderr, “tprealloc error =%s\n”, tpstrerror(tperrno));
    tpreturn (TPFAIL, tperrno, NULL, 0, TPNOFLAGS);
}

/*버퍼 초기화*/
ACCT_V->ACCOUNT_ID = 0;
strcpy(ACCT_V->PHONE,“ ” ) ;
strcpy(ACCT_V->ADDRESS,“ ” ) ;

/* ACCOUNT 테이블로부터 phone, address 필드를 뽑음 */
EXEC SQL declare CUR_1 cursor for
          select account_id,phone,address
           into :account_id, :pfone, :address
           from ACCOUNT
        where account_id > :key;/* 필드 키보다 account id가 클 때 */

/* cursor open */
EXEC SQL open CUR_1;
/* cursor open error */
if (sqlca.sqlcode ! = 0) {
       printf(“open cursor error !\n”);
       tpreturn(TPFAIL, sqlca.sqlcode, NULL, 0, TPNOFLAGS);
}

nfetch=0 ;
return_code = NOTFOUND;

/* cursor open success */
while (sqlca.sqlcode == 0) {
    /* cursor서부터 data 하나씩 fetch */
    EXEC SQL fetch CUR_1;

    /* fetch error */
    if (sqlca.sqlcode ! = 0) {
        if (sqlca.sqlcode == NOTFOUND)
            break ;
         break ;
     }

    ACCT_V[nfetch].ACCOUNT_ID = account_id;
    strcpy(ACCT_V[nfetch].PHONE, phone );
    strcpy(ACCT_V[nfetch].ADDRESS, address );

     /* select한 데이터 개수 증가 */
     nfetch++;
     return_code = nfetch

     /* NFETCH 개수만큼 select 했다면 while문 벗어남 */
    if (nfetch > NFETCH) {
        nfetch = NFETCH;
        return_code = nfetch;
        break ;
```

```
            }
        }
        /* cursor close */
        EXEC SQL close CUR_1;

        /* select의 결과와 그 데이터를 클라이언트에게 리턴 */
        tpreturn ( TPSUCCESS, return_code, (char *)ACCT_V,
                   sizeof(struct stru_acct)*nfetch, TPNOFLAGS);
}
```

5.3.5. DB2 프로그램

클라이언트는 사용자의 입력을 받아 STRING 버퍼에 EMPNO 넣어 서비스를 호출하며 서버는 이를 받아 해당 테이블에 추가한다. 클라이언트는 트랜잭션을 지정하여 에러가 발생하였을 경우 rollback할 수 있도록 한다.

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
sample.m	Tmax 환경설정 파일이다.
create.ers	데이터베이스 테이블을 생성하는 SQL 스크립트이다.

- 클라이언트 프로그램

프로그램 파일	설명
clidb2tx.c	클라이언트 프로그램이다.

- 서버 프로그램

프로그램 파일	설명
svr_db2.sqc	서비스 프로그램의 Oracle 소스이다.
Makefile	TMS 및 Server 프로그램을 컴파일하는 Makefile 이다.

프로그램 특징

- 클라이언트 프로그램

기능	설명
Tmax 접속	기본 접속을 한다.
버퍼 유형	STRING이다.
통신 유형	tpcall()에 의한 동기 통신을 한다.
트랜잭션 처리	클라이언트에서 트랜잭션 범위를 지정한다.

- 서버 프로그램

구분	설명
서비스	XASERVICE2이다.
데이터베이스 연결	DB2 데이터베이스이다.

DB2 연동 테스트 전에 확인해야 될 사항

DB2 연동할 때 다음과 같은 사항을 고려해야 한다.

1. DB2 클라이언트 엔진이 32bit 인지 64bit인지 확인한다.
2. DB2 클라이언트 버전이 8.0 이하인지 9.0 이상인지 확인한다.
3. Tmax 환경설정 파일의 SVRGROUP 절 XAOPTION 항목을 1, 2번의 결과를 바탕으로 설정한다.
    - DB2 클라이언트 버전이 8.0 이하일 경우
        - DB2 클라이언트 엔진이 32bit일 경우

        ```
        XAOPTION = "DYNAMIC"
        ```

        - DB2 클라이언트 엔진이 64bit이고 Linux/UNIX 계열일 경우

        ```
        XAOPTION = "DYNAMIC XASWITCH32"
        ```

        - DB2 클라이언트 엔진이 64bit이고 Windows 계열일 경우

        ```
        XAOPTION = "DYNAMIC"
        ```

    - DB2 클라이언트 버전이 9.0 이상일 경우(DB2 클라이언트 엔진 bit나 Linux/UNIX /Windows에 상관없음)

    ```
    XAOPTION = 미설정
    ```

4. TMS 또는 SVR 컴파일할 때 1, 2, 3번의 결과를 바탕으로 적절한 라이브러리를 link한다.
    - DB2 클라이언트 버전이 8.0 이하일 경우
        - DB2 클라이언트 엔진이 32bit일 경우

        ```
        -ldb2s or $(TMAXDIR)/lib/libdb2s.a
        ```

        - DB2 클라이언트 엔진이 64bit이고 Linux/UNIX 계열일 경우

        ```
        -ldb2_64s or $(TMAXDIR)/lib64/libdb2_64s.a
        ```

        - DB2 클라이언트 엔진이 64bit이고 Windows 계열일 경우

```
-ldb2s or $(TMAXDIR)/lib/libdb2s.a
```

◦ DB2 클라이언트 버전이 9.0 이상일 경우(DB2 클라이언트 엔진 bit나 Linux/UNIX /Windows에 상관없음)

```
-ldb2s_static or $(TMAXDIR)/lib/libdb2s_static.a
```

DB2 클라이언트 버전이 9.0 이상 동적 등록을 꼭 써야 할 필요가 있을 경우 위의 8.0의 경우와 동일하게 설정할 수는 있으나 사용을 권장하지 않는다.

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
*DOMAIN
tmax1           SHMKEY = 71990, MINCLH = 1, MAXCLH = 3,
                TPORTNO = 7789, BLOCKTIME = 30,
                MAXCPC = 150

*NODE
phk         TMAXDIR = "/home/tmaxha/tmax",
                APPDIR  = "/home/tmaxha/tmax/appbin",
                PATHDIR = "/home/tmaxha/tmax/path",
                TLOGDIR = "/home/tmaxha/tmax/log/tlog",
                ULOGDIR = "/home/tmaxha/tmax/log/ulog",
                SLOGDIR = "/home/tmaxha/tmax/log/slog"

*SVRGROUP
xa_svg_db2      NODENAME = "phk",
                  DBNAME = IBMDB2,
#                 XAOPTION = "DYNAMIC XASWITCH32",
                  XAOPTION = "DYNAMIC",
                  OPENINFO = "db=test,uid=tmaxha,pwd=ha0115",
                  TMSNAME = tms_db2,
                  RESTART=N

*SERVER
svr_db2                 SVGNAME = xa_svg_db2

*SERVICE
XASERVICE2          SVRNAME = svr_db2
```

다음은 환경설정에 추가된 항목에 대한 설명이다.

항목	설명
DBNAME	사용하는 데이터베이스를 정의한다.
OPENINFO	DB2 데이터베이스 연결 정보를 설정하는 항목이다.

항목	설명
TMSNAME	트랜잭션 처리를 주재하는 프로세스명을 지정하는 항목으로 OPENINFO 지정으로 인한 자동 트랜잭션을 처리한다.

데이터베이스 스크립트

다음은 DB2 테이블 생성 예제이다.

```
#'db2start' 실행
$ db2start

# 'TPTEST'라는 database 생성
$ db2 "CREATE DATABASE TPTEST"

# TPTEST databse에 접속
$ db2 "CONNECT TO TPTEST"

# EMP table 생성
$ db2 -vf create.ers -t

<create.ers>
CREATE TABLE EMP (
         EMPNO                  DECIMAL(8) NOT NULL,
         ENAME                  VARCHAR(16),
         JOB                    VARCHAR(16),
         SAL                    DECIMAL(8),
         HIREDATE               DECIMAL(8),
         XID                    CHAR(32)
);

# EMP table이 생성되었는지 확인
$ db2 "LIST TABLES"
```

다음은 DB2 테이블 및 데이터 출력 예제이다.

```
$ db2 "DESCRIBE TABLE EMP"

Column                         Type      Type
name                           schema    name               Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
EMPNO                          SYSIBM    DECIMAL                 8     0 No
ENAME                          SYSIBM    VARCHAR                16     0 Yes
JOB                            SYSIBM    VARCHAR                16     0 Yes
SAL                            SYSIBM    DECIMAL                 8     0 Yes
HIREDATE                       SYSIBM    DECIMAL                 8     0 Yes
XID                            SYSIBM    CHARACTER              32     0 Yes

  6 record(s) selected.
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<clidb2tx.c>

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <usrinc/atmi.h>

int main(int argc, char *argv[])
{
    char       *sndbuf, *rcvbuf;
    long       rcvlen;
    int        ret;

    if ( (ret = tmaxreadenv( "tmax.env","TMAX" )) == -1 ){
        printf( "tmax read env failed.[%s]\n", tpstrerror(tperrno));
        exit(1);
    }

    if (tpstart((TPSTART_T *)NULL) == -1){
        printf("tpstart failed.[%s]\n", tpstrerror(tperrno));
        exit(1);
    }

    if ((sndbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL) {
        printf("sendbuf alloc failed! [%s]\n", tpstrerror(tperrno));
        tpend();
        exit(1);
    }

    if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL) {
        printf("recvbuf alloc failed! [%s]\n", tpstrerror(tperrno));
        tpfree(sndbuf);
        tpend();
        exit(1);
    }

    strcpy(sndbuf, argv[1]);

    ret = tx_begin();
    if (ret < 0) {
        printf("tx_begin is failed.[%s]\n", tpstrerror(tperrno));
        resource_free(sndbuf, rcvbuf);
        exit(1);
    } else
        printf("tx_begin success.\n");

    if (tpcall("XASERVICE2", sndbuf, strlen(sndbuf), &rcvbuf, &rcvlen, 0) == -1){
        printf("Can't send request to service XASERVICE2.[%s]\n", tpstrerror(tperrno));
        ret = tx_rollback();
        if (ret < 0)
            printf("tx_rollback is failed. [%s]\n", tpstrerror(tperrno));

        resource_free(&sndbuf, &rcvbuf);
        exit(1);
    } else
        printf("XASERVCE2 success.\n");
```

```
    ret = tx_commit();
    if (ret < 0) {
        printf("tx_commit is failed. [%s]\n", tpstrerror(tperrno));
        ret = tx_rollback();
        if (ret < 0)
            printf("tx_rollback is failed. [%s]\n", tpstrerror(tperrno));
    } else
        printf("tx_commit success.\n");
    resource_free(sndbuf, rcvbuf);

    return 0;
}

resource_free(char* sndbuf, char *rcvbuf)
{
    if (rcvbuf != NULL)
        tpfree((char*)rcvbuf);

    if (sndbuf != NULL)
       tpfree((char*)sndbuf);

   tpend();
}
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

<svr_db2.sqc>

```
#include <stdio.h>
#include <usrinc/atmi.h>

EXEC SQL INCLUDE SQLCA;

EXEC SQL begin declare section;
         sqlint32 h_empno;
         sqlint32 h_count;
EXEC SQL end declare section;

XASERVICE2(TPSVCINFO *msg)
{
         char         *res_msg;
         int  h_count=0;

          h_empno = atoi(msg->data);
         printf("h_empno = %d \n", h_empno);

         EXEC SQL
                  INSERT into EMP (EMPNO) VALUES (:h_empno);
         if (sqlca.sqlcode != 0) {
                  printf("insertion is failed : sqlcode[%d]%d\n", sqlca.sqlcode, sqlca.sqlstate);
                  tpreturn(TPFAIL, -1, 0, 0, 0);
         } else

        EXEC SQL SELECT COUNT(*)
        INTO    :h_count
```

```
        FROM    emp
        WHERE   empno = :h_empno;

         if (sqlca.sqlcode != 0) {
                 printf("insertion is failed : sqlcode[%d]%d\n", sqlca.sqlcode, sqlca.sqlstate);
                 tpreturn(TPFAIL, -1, 0, 0, 0);
         } else
                 printf("insertion is success. selcnt(%d) \n", h_count);

         tpreturn(TPSUCCESS, 0, 0, 0, 0);
}
```

서버 Makefile

다음은 서버 Makefile의 예제이다.

```
# Server makefile for DB2
# Linux 64bit

DB2LIBDIR = $(DB2_HOME)/lib
DB2LIBS           = -ldb2

DB         = TEST
DB2USER = tmaxha
DB2PASS = ha0115

TARGET  = $(COMP_TARGET)
APOBJS  = $(TARGET).o
APOBJS2  = utilemb.o
NSDLOBJ = $(TMAXDIR)/lib64/sdl.o

#OBJS    = $(APOBJS) $(APOBJS2) $(SVCTOBJ)
OBJS    = $(APOBJS)  $(SVCTOBJ)
SVCTOBJ = $(TARGET)_svctab.o

#CFLAGS  = -m64 -O -I$(TMAXDIR) -I$(DB2_HOME)/include
CFLAGS  =  -O -I$(TMAXDIR) -I$(DB2_HOME)/include
LDFLAGS =

TMAXAPPDIR  = $(TMAXDIR)/appbin
TMAXSVCTDIR = $(TMAXDIR)/svct
TMAXLIBDIR  = $(TMAXDIR)/lib64

TMAXLIBS    = -lsvr -ldb2s          # dynmic XAOPTION=DYNAMIC (DB2 Client v8.0(below) 32bit  or  DB2
Client 64bit & Windows)
#TMAXLIBS    = -lsvr -ldb2_64s      # dynmic XAOPTION=DYNAMIC (DB2 Client v8.0(below) 64bit &
Linux/Unix)
#TMAXLIBS    = -lsvr -ldb2s_static       #static XAOPTION=none (DB2 Client v9.0(above))
#
.SUFFIXES : .c

.c.o:
         $(CC) $(CFLAGS) $(LDFLAGS) -c $<

#
# server compile
#
```

```
all: $(TARGET)

$(TARGET): $(OBJS)
	$(CC) $(CFLAGS) $(LDFLAGS) -L$(TMAXLIBDIR) -o $(TARGET) -L$(DB2LIBDIR) $(DB2LIBS) $(OBJS)
$(TMAXLIBS) $(NSDLOBJ)
	mv $(TARGET) $(TMAXAPPDIR)
	rm -f $(OBJS)

$(APOBJS): $(TARGET).sqc
	db2 connect to $(DB) user $(DB2USER) using $(DB2PASS)
	db2 prep $(TARGET).sqc bindfile
	db2 bind $(TARGET).bnd
	db2 connect reset
	db2 terminate
	$(CC) $(CFLAGS) $(LDFLAGS) -c $(TARGET).c

$(SVCTOBJ):
	cp -f $(TMAXSVCTDIR)/$(TARGET)_svctab.c .
	touch ./$(TARGET)_svctab.c
	$(CC) $(CFLAGS) -c ./$(TARGET)_svctab.c

#
clean:
	:-rm -f *.o core $(TMAXAPPDIR)/$(TARGET) $(TARGET).bnd
```

TMS Makefile

다음은 TMS Makefile의 예제이다.

```
# TMS Makefile for DB2
# Linux 64bit

TARGET  = tms_db2
APOBJ   = dumy.o

APPDIR  = $(TMAXDIR)/appbin
TMAXLIBD= $(TMAXDIR)/lib64
TMAXLIBS    = -lsvr -ldb2s          # dynmic XAOPTION=DYNAMIC (DB2 Client v8.0(below) 32bit  or  DB2
Client 64bit & Windows)
#TMAXLIBS    = -lsvr -ldb2_64s      # dynmic XAOPTION=DYNAMIC (DB2 Client v8.0(below) 64bit &
Linux/Unix)
#TMAXLIBS    = -lsvr -ldb2s_static       #static XAOPTION=none (DB2 Client v9.0(above))

DB2PATH = $(DB2_HOME)
DB2LIBDIR= $(DB2PATH)/lib
DB2LIB  = -ldb2

CFLAGS  =
LDFLAGS =
SYSLIBS =

all: $(TARGET)

$(TARGET): $(APOBJ)
	$(CC) $(CFLAGS) $(LDFLAGS) -o $(TARGET) -L$(TMAXLIBD) $(TMAXLIBS) $(APOBJ) -L$(DB2LIBDIR)
$(DB2LIB) $(SYSLIBS)
	mv $(TARGET) $(APPDIR)/.
```

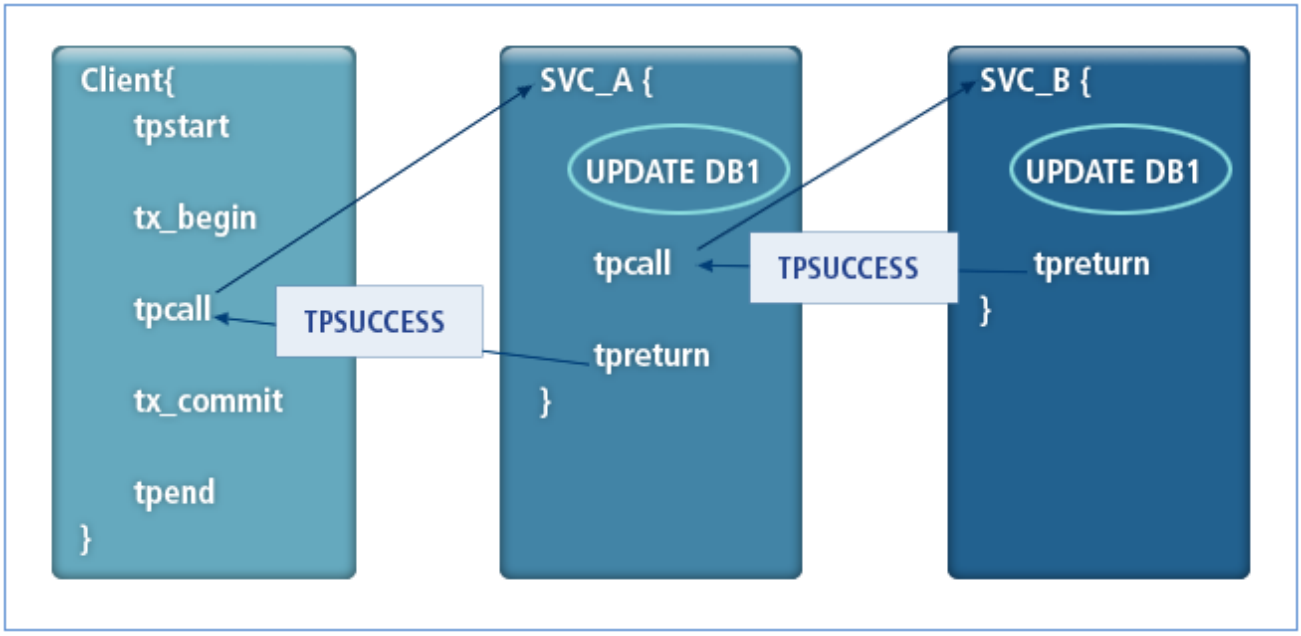
```
$(APOBJ):
        $(CC) $(CFLAGS) -c dumy.c
#
clean:
        -rm -f *.o core $(APPDIR)/$(TARGET)
```

5.4. 데이터베이스 연동 프로그램

애플리케이션을 개발할 때 응용하여 사용할 수 있는 예제를 통해 실제 프로그래밍의 기법을 설명한다. 데이터베이스 연동은 동일 기종 데이터베이스와 이기종 데이터베이스로 나뉜다.

5.4.1. 동기형 모드(동일 기종)

다음은 동기형 모드에서 동일 기종 데이터베이스에 접속하는 경우 프로그램 흐름에 대한 그림이다.



동기형 모드 흐름도(동일 기종 데이터베이스)

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
demo.s	SDLFILE이다.
sample.m	Tmax 환경 파일이다.
tmax.env	환경 파일이다.
mktable.sql	데이터베이스에 테이블을 생성하는 SQL 스크립트이다.

- 클라이언트 프로그램

프로그램 파일	설명
client.c	클라이언트 프로그램이다.

- 서버 프로그램

프로그램 파일	설명
update.pc, insert.pc	서버 프로그램이다.
Makefile	Tmax에서 제공되는 Makefile을 수정해야 한다.

프로그램 특징

- 클라언트 프로그램

구분	설명
Tmax 연결	NULL 파라미터로 연결한다.
버퍼 유형	STRUCT이다.
하위 유형	input 구조체 파일을 sdlc로 컴파일하여 SDL 파일 생성이 필요하다.
트랜잭션 여부	클라이언트에서 트랜잭션을 지정한다.

- 서버 프로그램

구분	설명
서비스 수	UPDATE 서비스에서 INSERT 서비스를 요청한다.
데이터베이스 연결	Oracle 데이터베이스 사용 Tmax 환경 파일의 SVRGROUP 절에 데이터베이스 정보를 지정한다.

프로그램 환경

구분	설명
시스템	SunOS 5.7 32bit
데이터베이스	Oracle 8.0.5

구조체 버퍼

다음은 구조체 버퍼의 예제이다.

<demo.s>

```
struct input {
        int     account_id;
        int     branch_id;
        char    phone[15];
        char    address[61];
};
```

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
*DOMAIN
res   SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60

*NODE
tmax          TMAXDIR = "/user/ tmax ",
              APPDIR  = "/user/ tmax /appbin",
              PATHDIR = "/user/ tmax /path",
              TLOGDIR = "/user/ tmax /log/tlog",
              ULOGDIR="/user/ tmax /log/ulog",
              SLOGDIR="/user/ tmax /log/slog"

*SVRGROUP
svg1          NODENAME = tmax, DBNAME = ORACLE,
              OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
              TMSNAME = svg1_tms

*SERVER
update        SVGNAME=svg1
insert        SVGNAME=svg1

*SERVICE
UPDATE        SVRNAME=update
INSERT        SVRNAME=insert
```

다음은 환경 파일의 예제이다.

<tmax.env>

```
[tmax]
TMAX_HOST_ADDR=192.168.1.39
TMAX_HOST_PORT=8880
SDLFILE=/user/tmax/sample/sdl/tmax.sdl
TMAX_CONNECT_TIMEOUT=5
```

데이터베이스 스크립트

다음은 데이터베이스 테이블을 생성하는 SQL 스크립트의 예제이다.

<mktable.sql>

```
$ORACLE_HOME/bin/sqlplus bmt/bmt <<!
drop table ACCOUNT;
create table ACCOUNT (
        ACCOUNT_ID  integer,
        BRANCH_ID   integer not null,
        SSN     char(13) not null,
        BALANCE     number,
```

```
        ACCT_TYPE   char(1),
        LAST_NAME   char(21),
        FIRST_NAME  char(21),
        MID_INIT    char(1),
        PHONE       char(15),
        ADDRESS     char(61),
        CONSTRAINT ACCOUNT_PK PRIMARY KEY(ACCOUNT_ID)
);
quit
!
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<client.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

#define TEMP_PHONE "6283-2114"
#define TEMP_ADDRESS "Korea"

int main(int argc, char *argv[])
{
        struct input *sndbuf;
        char    *rcvbuf;
        int acnt_id, n, timeout;
        long    len;

        if (argc != 2) {
            fprintf(stderr, "Usage:%s account_id \n", argv[0]);
            exit(1);
        }

        acnt_id = atoi(argv[1]);
        timeout = 5;

        n = tmaxreadenv("tmax.env", "tmax");
        if (n < 0) {
            fprintf(stderr, "tmaxreadenv fail! tperrno = %d\n", tperrno);
            exit(1);
        }

        n = tpstart((TPSTART_T *)NULL);
        if (n < 0) {
            fprintf(stderr, "tpstart fail! tperrno = %s\n", tperrno);
            exit(1);
        }

        sndbuf = (struct input *)tpalloc("STRUCT", "input", `
                  sizeof(struct input));
        if (sndbuf == NULL) {
            fprintf(stderr, "tpalloc fail: sndbuf tperrno = %d\n", tperrno);
            tpend();
            exit(1);
```

```
        }

        rcvbuf = (char *)tpalloc("STRING", NULL, 0);
        if (rcvbuf == NULL) {
            fprintf(stderr, "tpalloc fail: rcvbuf tperrno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        sndbuf->account_id = acnt_id;
        sndbuf->branch_id = acnt_id;
        strcpy(sndbuf ->phone, TEMP_PHONE);
        strcpy(sndbuf ->address, TEMP_ADDRESS);

        tx_set_transaction_timeout(timeout);
        n = tx_begin();
        if (n < 0)
            fprintf(stderr, "tx begin fail! tperrno = %d\n", tperrno);

        n = tpcall("UPDATE", (char *)sndbuf, sizeof(struct input),
                   (char **)&rcvbuf, (long *)&len, TPNOFLAGS);
        if (n < 0) {
            fprintf(stderr, "tpcall fail! tperrno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        n = tx_commit();
        if (n < 0) {
            fprintf(stderr, "tx commit fail! tx error = %d \n", n);
            tx_rollback();
            tpend();
            exit(1);
        }

        printf("rtn msg = %s\n", rcvbuf);
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
}
```

서버 프로그램

다음은 데이터베이스에 UPDATE하는 서버 프로그램의 예제이다.

<update.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/sdl.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
```

```
        int     account_id;
        int     branch_id;
        char    ssn[15];
        char    phone[15];
        char    address[61];
EXEC SQL END DECLARE SECTION;

UPDATE(TPSVCINFO *msg)
{
        struct input *rcvbuf;
        int     ret;
        long    acnt_id, rcvlen;
        char    *send;

        rcvbuf = (struct input *)(msg->data);
            send = (char *)tpalloc("STRING", NULL, 0);
        if (send == NULL) {
            fprintf(stderr, "tpalloc fail errno = %s\n", strerror(tperrno));
            tpreturn(TPFAIL, 0, (char *)NULL, 0, 0);
        }
        account_id = rcvbuf->account_id;
        branch_id = rcvbuf->branch_id;
}
```

다음은 데이터베이스에 INSERT하는 서버 프로그램의 예제이다.

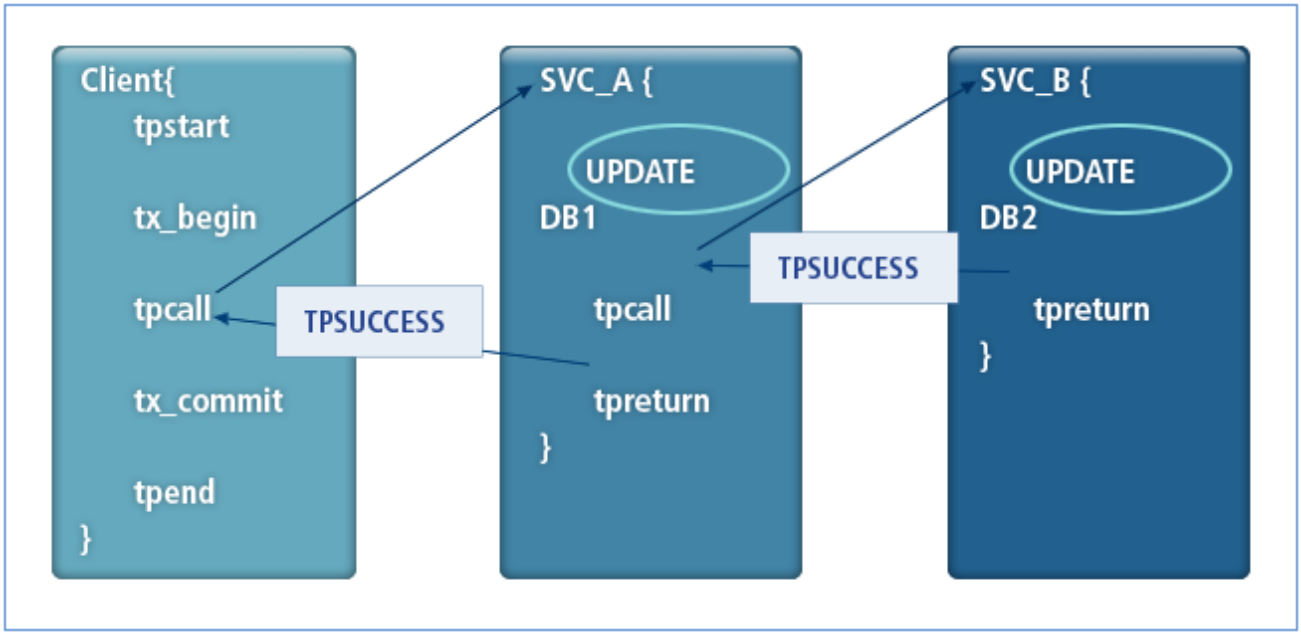
< insert.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/sdl.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
        int     account_id;
        int     branch_id;
        char    ssn[15];
        char    phone[15];
        char    address[61];
EXEC SQL END DECLARE SECTION;

INSERT(msg)
TPSVCINFO  *msg;
{
        struct input *rcvbuf;
        int     ret;
        long    acnt_id;
        char    *send;

        rcvbuf = (struct input *)(msg->data);
        send = (char *)tpalloc("STRING", NULL, 0);
        if (send == NULL) {
            fprintf(stderr, "tpalloc fail errno = %s\n", tpstrerror(tperrno));
```

```
            tpreturn(TPFAIL, 0, (char *)NULL, 0, TPNOFLAGS);
        }

        account_id = rcvbuf->account_id;
        branch_id = rcvbuf->branch_id;
        strcpy(phone, rcvbuf->phone);
        strcpy(address, rcvbuf->address);
        strcpy(ssn, "1234567");

        /*  Declare && Open Cursor for Fetch */
        EXEC SQL INSERT INTO ACCOUNT (
            ACCOUNT_ID,
            BRANCH_ID,
            SSN,
            PHONE,
            ADDRESS )
        VALUES (:account_id, :branch_id, :ssn, :phone, :address);

        if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 )
        {
            printf("insert failed sqlcode = %d\n", sqlca.sqlcode);
            tpreturn(TPFAIL, -1, (char *)NULL, 0, TPNOFLAGS);
        }
        strcpy(send, OKMSG);
        tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}
```

5.4.2. 동기형 모드(이기종)

다음은 동기형 모드에서 이기종 데이터베이스에 접속하는 경우 프로그램 흐름에 대한 그림이다.



동기형 모드 흐름도(이기종 데이터베이스 접속)

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
demo.s	SDLFILE이다.
sample.m	Tmax 환경 파일이다.
tmax.env	환경 파일이다.
mktable.sql	데이터베이스에 테이블을 생성하는 SQL 스크립트이다.

- 클라이언트 프로그램

구성	설명
client.c	클라이언트 프로그램이다.

- 서버 프로그램

구성	설명
update.pc, insert.pc	서버 프로그램이다.
Makefile	Tmax에서 제공되는 Makefile을 수정해야 한다.

클라이언트와 서버프로그램 등 모두 동기형 모드(동일 기종)의 경우와 같다. 멀티 노드 환경 설정에 대한 자세한 내용은 "Tmax Administration Guide"를 참고한다.

프로그램 특징

- 클라이언트 프로그램

구분	설명
Tmax 연결	NULL 파라미터로 연결한다.
버퍼 유형	STRUCT이다.
하위 유형	input 구조체 파일을 sdlc로 컴파일하여 SDL 파일 생성이 필요하다.
트랜잭션 여부	클라이언트에서 명시적으로 트랜잭션 범위를 지정한다.

- 서버 프로그램

구분	설명
서비스 수	UPDATE 서비스에서 INSERT 서비스를 요청한다.
데이터베이스 연결	Oracle 데이터베이스를 사용하고 시스템 구성 파일의 SVRGROUP 절에 데이터베이스 정보를 지정한다.

프로그램 환경

구분	설명
시스템	SunOS 5.7 32bit, SunOS 5.8 32bit

구분	설명
데이터베이스	Oracle 8.0.5

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

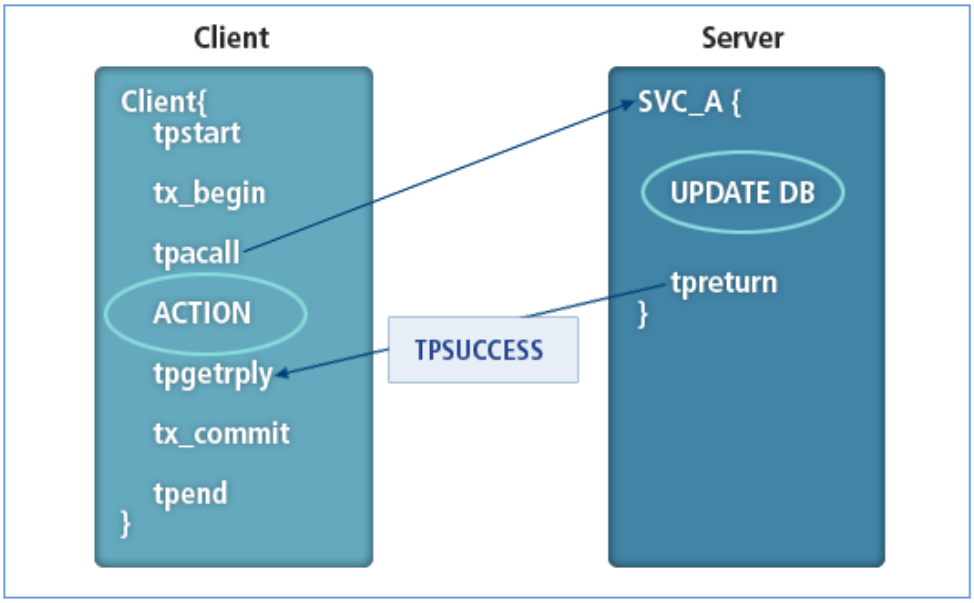
```
*DOMAIN
res         SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60

*NODE
tmax1       TMAXDIR="/user/ tmax ",
            APPDIR="/user/ tmax /appbin",
            PATHDIR = "/user/ tmax /path",
            TLOGDIR = "/user/ tmax /log/tlog",
            ULOGDIR="/user/ tmax /log/ulog",
            SLOGDIR="/user/ tmax /log/slog"

tmax2       TMAXDIR="/user/ tmax ",
            APPDIR="/user/ tmax /appbin",
            PATHDIR = "/user/ tmax /path",
            TLOGDIR = "/user/ tmax /log/tlog",
            ULOGDIR="/user/ tmax /log/ulog",
            SLOGDIR="/user/ tmax /log/slog"

*SVRGROUP
svg1        NODENAME = tmax1, DBNAME = ORACLE,
            OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
            TMSNAME = svg1_tms

svg2        NODENAME = tmax2, DBNAME = ORACLE,
            OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
            TMSNAME = svg2_tms

*SERVER
update      SVGNAME=svg1
insert      SVGNAME=svg2

*SERVICE
UPDATE      SVRNAME=update
INSERT      SVRNAME=insert
```

다음은 환경 파일의 예제이다.

```
[tmax1]
TMAX_HOST_ADDR=192.168.1.39
TMAX_HOST_PORT=8880
SDLFILE=/user/tmax/sample/sdl/tmax.sdl
TMAX_CONNECT_TIMEOUT=5
```

5.4.3. 비동기형 모드(동일 기종)

다음은 비동기형 모드에서 동일 기종데이터베이스에 접속하는 경우 프로그램 흐름에 대한 그림이다.



비동기형 모드 흐름도(동일 기종 데이터베이스 접속)

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
demo.s	SDLFILE이다.
sample.m	Tmax 환경 파일이다.
tmax.env	환경 파일이다.
mktable.sql	데이터베이스에 테이블을 생성하는 SQL 스크립트이다.

- 클라이언트 프로그램

프로그램 파일	설명
client.c	클라이언트 프로그램이다.

- 서버 프로그램

프로그램 파일	설명
update.pc	서버 프로그램이다.
Makefile	Tmax에서 제공되는 Makefile로 수정해야 한다.

프로그램 특징

- 클라이언트 프로그램

기능	설명
Tmax 연결	NULL 파라미터로 연결한다.
버퍼 유형	STRUCT이다.
하위 유형	input 구조체 파일을 sdlc로 컴파일하여 SDL 파일 생성이 필요하다.
트랜잭션 여부	클라이언트에서 트랜잭션을 지정한다.

- 서버 프로그램

기능	설명
서비스 수	UPDATE 서비스를 요청한다.
데이터베이스 연결	Orale 데이터베이스 사용하고 시스템 구성 파일의 SVRGROUP 절에 데이터베이스 정보를 지정한다.

프로그램 환경

구분	설명
시스템	SunOS 5.7 32bit
데이터베이스	Oracle 8.0.5

구조체 버퍼

다음은 구조체 버퍼의 예제이다.

<demo.s>

```
struct input {
        int     account_id;
        int     branch_id;
        char    phone[15];
        char    address[61];
};
```

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

<sample.m>

```
*DOMAIN
res         SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60

*NODE
tmax        TMAXDIR="/user/ tmax ",
            APPDIR="/user/ tmax /appbin",
            PATHDIR = "/user/ tmax /path",
            TLOGDIR = "/user/ tmax /log/tlog",
```

```
            ULOGDIR="/user/ tmax /log/ulog",
            SLOGDIR="/user/ tmax /log/slog"

*SVRGROUP
svg1        NODENAME = tmax, DBNAME = ORACLE,
            OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
            TMSNAME = svg1_tms

*SERVER
update      SVGNAME=svg1

*SERVICE
UPDATE      SVRNAME=update
```

다음은 환경 파일의 예제이다.

<tmax.env>

```
[tmax]
TMAX_HOST_ADDR=192.168.1.39
TMAX_HOST_PORT=8880
SDLFILE=/user/tmax/sample/sdl/tmax.sdl
TMAX_CONNECT_TIMEOUT=5
```

데이터베이스 스크립트

다음은 데이터베이스에 테이블을 생성하는 SQL 스크립트의 예제이다.

<mktable.sql>

```
$ORACLE_HOME/bin/sqlplus bmt/bmt <<!
drop table ACCOUNT;
create table ACCOUNT (
        ACCOUNT_ID  integer,
        BRANCH_ID   integer not null,
        SSN     char(13) not null,
        BALANCE     number,
        ACCT_TYPE   char(1),
        LAST_NAME   char(21),
        FIRST_NAME  char(21),
        MID_INIT    char(1),
        PHONE       char(15),
        ADDRESS     char(61),
        CONSTRAINT ACCOUNT_PK PRIMARY KEY(ACCOUNT_ID)
);
quit
!
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<client.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

#define TEMP_PHONE "6283-2114"
#define TEMP_ADDRESS "Korea"

int main(int argc, char *argv[])
{
        struct input *sndbuf;
        char    *rcvbuf;
        int acnt_id, n, cd, timeout;
        long    len;

        if (argc != 2) {
            fprintf(stderr, "Usage:%s account_id \n", argv[0]);
            exit(1);
        }

        acnt_id = atoi(argv[1]);
        timeout = 5;

        n = tmaxreadenv("tmax.env", "tmax");
        if (n < 0) {
            fprintf(stderr, "tmaxreadenv fail! tperrno = %d\n", tperrno);
            exit(1);
        }

        n = tpstart((TPSTART_T *)NULL);
        if (n < 0) {
            fprintf(stderr, "tpstart fail! tperrno = %s\n", tperrno);
            exit(1);
        }

        sndbuf = (struct input *)tpalloc("STRUCT", "input", sizeof(struct input));
        if (sndbuf == NULL) {
            fprintf(stderr, "tpalloc fail: sndbuf tperrno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        rcvbuf = (char *)tpalloc("STRING", NULL, 0);
        if (rcvbuf == NULL) {
            fprintf(stderr, "tpalloc fail: rcvbuf tperrno = %d\n", tperrno);
            tpend();
            exit(1);
        }
        sndbuf->account_id = acnt_id;
        sndbuf->branch_id = acnt_id;
        strcpy(sndbuf->phone, TEMP_PHONE);
        strcpy(sndbuf->address, TEMP_ADDRESS);
        tx_set_transaction_timeout(timeout);
        n = tx_begin();
        if (n < 0)
            fprintf(stderr, "tx begin fail! tperrno = %d\n", tperrno);
```

```
        cd = tpacall("UPDATE", (char *)sndbuf, sizeof(struct input), TPNOFLAGS);
        if (cd < 0) {
            fprintf(stderr, "tpacall fail! tperrno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        n = tpgetrply(&cd, (char **)&rcvbuf, (long *)&len, TPNOFLAGS);
        if (n < 0) {
            fprintf(stderr, "tpgetrply fail! tperrno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        n = tx_commit();
        if (n < 0) {
            fprintf(stderr, "tx commit fail! tx error = %d \n", n);
            tx_rollback();
            tpend();
            exit(1);
        }
        printf("rtn msg = %s\n", rcvbuf);
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
}
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

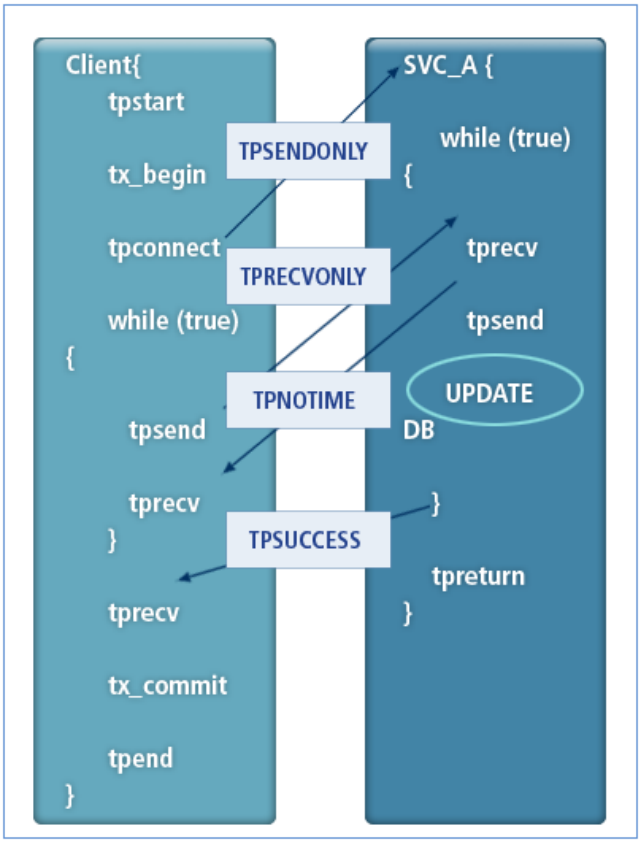
<update.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
        int     account_id;
        int     branch_id;
        char    ssn[15];
        char    phone[15];
        char    address[61];
EXEC SQL END DECLARE SECTION;

UPDATE(TPSVCINFO *msg)
{
        struct input *rcvbuf;
        int     ret, cd;
        long    acnt_id, rcvlen;
        char    *send;

        rcvbuf = (struct input *)(msg->data);
```

```
        send = (char *)tpalloc("STRING", NULL, 0);
        if (send == NULL) {
            fprintf(stderr, "tpalloc fail errno = %s\n", strerror(tperrno));
            tpreturn(TPFAIL, 0, (char *)NULL, 0, TPNOFLAGS);
        }
        account_id = rcvbuf->account_id;
        branch_id = rcvbuf->branch_id;
        strcpy(phone, rcvbuf->phone);
        strcpy(address, rcvbuf->address);
        strcpy(ssn, "1234567");

        EXEC SQL UPDATE ACCOUNT
        SET BRANCH_ID = :branch_id,
        PHONE = :phone,
        ADDRESS = :address,
        SSN = :ssn
        WHERE ACCOUNT_ID = :account_id;
        if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 ) {
            fprintf(stderr, "update failed sqlcode = %d\n", sqlca.sqlcode);
            tpreturn(TPFAIL, -1, (char *)NULL, 0, TPNOFLAGS);
        }
        strcpy(send, OKMSG);
        tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}
```

5.4.4. 대화형 모드(동일 기종)

다음은 대화형 모드에서 동일 기종 데이터베이스에 접속하는 경우 프로그램 흐름에 대한 그림이다.



대화형 모드 흐름도(동일 기종 데이터베이스 접속)

프로그램 구성

- 공통 프로그램

프로그램 파일	설명
demo.s	SDLFILE이다.
sample.m	Tmax 환경 파일이다.
tmax.env	환경 파일이다.
mktable.sql	데이터베이스에 테이블을 생성하는 SQL 스크립트이다.

- 클라이언트 프로그램

프로그램 파일	설명
client.c	클라이언트 프로그램이다.

- 서버 프로그램

프로그램 파일	설명
update.pc	서버 프로그램이다.
Makefile	Tmax에서 제공되는 Makefile을 수정해야 한다.

프로그램 특징

- 클라이언트 프로그램

기능	설명
Tmax 연결	NULL 파라미터로 연결한다.
버퍼 유형	STRUCT이다.
하위 유형	input 구조체 파일을 sdlc로 컴파일하여 SDL 파일 생성이 필요하다. (애플리케이션을 실행시키기 위해 필요)
트랜잭션 여부	클라이언트에서 트랜잭션을 지정한다.

- 서버 프로그램

기능	설명
서비스 수	UPDATE 서비스를 요청한다.
데이터베이스 연결	Oracle 데이터베이스 지정하고 Tmax 환경 파일의 SVRGROUP 절에 데이터베이스 정보를 지정한다.

프로그램 환경

구분	설명
시스템	SunOS 5.7 32bit

구분	설명
데이터베이스	Oracle 8.0.5

구조체 버퍼

다음은 구조체 버퍼의 예제이다.

<demo.s>

```
struct input {
        int     account_id;
        int     branch_id;
        char    phone[15];
        char    address[61];
};
```

Tmax 환경 파일

다음은 Tmax 환경 파일의 예제이다.

```
*DOMAIN
res   SHMKEY=88000, MINCLH=1, MAXCLH=5, TPORTNO=8880, BLOCKTIME=60


*NODE
tmax        TMAXDIR="/user/ tmax ",
            APPDIR="/user/ tmax /appbin",
            PATHDIR = "/user/ tmax /path",
            TLOGDIR = "/user/ tmax /log/tlog",
            ULOGDIR="/user/ tmax /log/ulog",
            SLOGDIR="/user/ tmax /log/slog"

*SVRGROUP
svg1        NODENAME = tmax, DBNAME = ORACLE,
            OPENINFO = "ORACLE_XA+Acc=P/bmt/bmt+SesTm=60",
            TMSNAME = svg1_tms

*SERVER
update      SVGNAME=svg1, CONV=YES

*SERVICE
UPDATE      SVRNAME= update
```

다음은 환경 파일의 예제이다.

<tmax.env>

```
[tmax]
TMAX_HOST_ADDR=192.168.1.39
TMAX_HOST_PORT=8880
SDLFILE=/user/tmax/sample/sdl/tmax.sdl
```

```
TMAX_CONNECT_TIMEOUT=5
```

데이터베이스 스크립트

다음은 데이터베이스에 테이블을 생성하는 SQL 스크립트의 예제이다.

<mktable.sql>

```
$ORACLE_HOME/bin/sqlplus bmt/bmt <<!
drop table ACCOUNT;
create table ACCOUNT (
        ACCOUNT_ID  integer,
        BRANCH_ID   integer not null,
        SSN     char(13) not null,
        BALANCE     number,
        ACCT_TYPE   char(1),
        LAST_NAME   char(21),
        FIRST_NAME  char(21),
        MID_INIT    char(1),
        PHONE       char(15),
        ADDRESS     char(61),
        CONSTRAINT ACCOUNT_PK PRIMARY KEY(ACCOUNT_ID)
);
quit
!
```

클라이언트 프로그램

다음은 클라이언트 프로그램의 예제이다.

<client.c>

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

#define TEMP_PHONE "6283-2115"
#define TEMP_ADDRESS "Korea"

void main(int argc, char *argv[])
{
        struct input *sndbuf;
        char    *rcvbuf;
        int acntid, timeout;
        long revent, rcvlen;
        int cd, n;

        if (argc != 2) {
            fprintf(stderr, "Usage:%s acntid\n", argv[0]);
            exit(1);
        }

        acntid = atoi(argv[1]);
        timeout = 5;
```

```
        n = tmaxreadenv("tmax.env", "tmax");
        if (n < 0) {
            fprintf(stderr, "tmaxreadenv fail tperrno = %d\n", tperrno);
            exit(1);
        }

        n = tpstart((TPSTART_T *)NULL);
        if (n < 0) {
            fprintf(stderr, "tpstart fail tperrno = %s\n", tperrno);
            exit(1);
        }
        printf("tpstart ok!\n");
        sndbuf = (struct input *)tpalloc("STRUCT", "input", sizeof(struct input));
        if (sndbuf == NULL) {
            fprintf(stderr, "tpalloc fail: sndbuf tperrno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        rcvbuf = (char *)tpalloc("CARRAY", NULL, 0);
        if (rcvbuf == NULL) {
            fprintf(stderr, "tpalloc fail: rcvbuf tperrno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        sndbuf->account_id = acntid;
        sndbuf->branch_id = acntid;
        strcpy(sndbuf->phone, TEMP_PHONE);
        strcpy(sndbuf->address, TEMP_ADDRESS);

        tx_set_transaction_timeout(timeout);
        n = tx_begin();
        if (n < 0)
            fprintf(stderr, "tx begin fail tx error = %d\n", n);
        printf("tx begin ok!\n");

        cd = tpconnect("UPDATE", (char *)sndbuf, 0, TPSENDONLY);
        if (cd < 0) {
            fprintf(stderr, "tpconnect fail tperrno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        while (1) {
            n = tpsend(cd, (char *)sndbuf, sizeof(struct input), TPRECVONLY,
                       &revent);
            if (n < 0) {
                fprintf(stderr, "tpsend fail revent = 0x%08x\n", revent);
             tx_rollback();
                tpend();
                exit(1);
            }
            printf("tpsend ok\n");

            n = tprecv(cd, (char **)&rcvbuf, (long *)&rcvlen, TPNOTIME, &revent);
            if (n < 0 && revent != TPEV_SENDONLY) {
                fprintf(stderr, "tprecv fail revent = 0x%08x\n", revent);
                tx_rollback();
```

```
                tpend();
                exit(1);
            }
            printf("tprecv ok\n");
            sndbuf->account_id++;

            if (revent != TPEV_SENDONLY)
            break;
        }
        n = tprecv(cd, (char **)&rcvbuf, (long *)&rcvlen, TPNOTIME, &revent);
        if (n < 0 && revent != TPEV_SVCSUCC) {
            fprintf(stderr, "tprecv fail revent = 0x%08x\n", revent);
            tx_rollback();
            tpend();
            exit(1);
        }
        printf("rcvbuf = [%s]\n", rcvbuf);

        n = tx_commit();
        if (n < 0) {
            fprintf(stderr, "tx commit fail tx error = %d\n", n);
            tx_rollback();
            tpend();
            exit(1);
        }
        printf("tx commit ok!\n");
        printf("rtn msg = %s\n", rcvbuf);
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
}
```

서버 프로그램

다음은 서버 프로그램의 예제이다.

<update.pc>

```
#include <stdio.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include "../sdl/demo.s"

void _db_work();

#define OKMSG "YOU COMPLETE THE TRANSACTION"

EXEC SQL include sqlca.h;
EXEC SQL BEGIN DECLARE SECTION;
        int     account_id;
        int     branch_id;
        char    ssn[15];
        char    phone[15];
        char    address[61];
EXEC SQL END DECLARE SECTION;

struct input *rcvbuf;
```

```
UPDATE(TPSVCINFO *msg)
{
        int     ret, count;
        long    acnt_id;
        long    revent, rcvlen, flag;
        char    *send;

        rcvbuf = (struct input *)tpalloc("STRUCT", "input", 0);
        send = (char *)tpalloc("CARRAY", NULL, 0);

        count = 1;
        flag = 0;

        while (1) {
            ret = tprecv(msg->cd, (char **)&rcvbuf, &rcvlen, TPNOTIME, &revent);
            if (ret < 0 && revent != TPEV_SENDONLY) {
                fprintf(stderr, "tprecv fail revent = 0x%08x\n", revent);
                tpreturn(TPFAIL, -1, (char *)rcvbuf, 0, TPNOFLAGS);
            }
            printf("tprecv ok!\n");

            if (count == 10) {
                flag &= ~TPRECVONLY;
                flag |= TPNOTIME;
            }
            else
                flag |= TPRECVONLY;

            ret = tpsend(msg->cd, (char *)send, strlen(send), flag, &revent);
            if (ret < 0) {
                fprintf(stderr, "tpsend fail revent = 0x%08x\n", revent);
                tpreturn(TPFAIL, -1, (char *)NULL, 0, TPNOFLAGS);
            }
            printf("tpsend ok!\n");
            _db_work();

            /* break after 10 iterations */
            if (count == 10)
                break;

            count++;
        }

        strcpy(send, OKMSG);
        printf("tpreturn ok!\n");
        tpreturn(TPSUCCESS, 1, (char *)send, strlen(send), TPNOFLAGS);
}

void _db_work() {
        account_id = rcvbuf->account_id;
        branch_id = rcvbuf->branch_id;
        strcpy(phone, rcvbuf->phone);
        strcpy(address, rcvbuf->address);
        strcpy(ssn, "1234567");

        EXEC SQL UPDATE ACCOUNT
        SET BRANCH_ID = :branch_id,
            PHONE = :phone,
```

```
            ADDRESS = :address,
            SSN = :ssn
            WHERE ACCOUNT_ID = :account_id;

        if (sqlca.sqlcode != 0 && sqlca.sqlcode != 1403 )
        {
            fprintf(stderr, "update failed sqlcode = %d\n", sqlca.sqlcode);
            tpreturn(TPFAIL, -1, (char *)NULL, 0, 0);
        }
}
```

5.5. TIP를 이용한 프로그램

Tmax에서는 TIP(Tmax Information Provider)를 사용하여 시스템 환경 정보, 통계 정보 확인, 시스템 운용 관리 등 다양한 기능을 수행할 수 있다. TIP은 TIPSVC를 처리하기 위하여 Tmax에서 제공하는 기능 프로세스로 TIP을 사용하여 다음의 기능을 수행할 수 있다.

- 시스템 환경 정보 확인: 시스템의 정적인 환경 정보를 확인한다.
- 시스템 통계 정보 확인: 시스템 운영 중에 각각 프로세스의 상태 등을 확인한다.
- 시스템 운용 관리: 프로세스를 추가로 기동하거나 종료한다.

5.5.1. TIP 구조

TIPSVC를 처리하기 위한 별도의 기능 프로세스인 TIP 서버는 SYS_SVR 서버 타입을 가지며, TIP 서버 그룹에 속한다. 클라이언트 혹은 서버의 요청을 받은 TIP 서버는 그 요청을 CLH/TMM에게 전달하여 처리 결과를 다시 요청자에게 돌려준다. TIP 서버는 필드 키를 바탕으로 서비스를 처리한다. 요청하는 클라이언트나 서버는 필드 버퍼에 요구하려는 데이터를 실어 요청을 하며, 그 결과를 필드 버퍼로 받는다.

- CHLOG section (로그 레벨 변경)

  CHLOG는 TMM, CLH, TMS, SVR의 log level을 변경할 수 있는 섹션이다. tmadmin에서 **chlog**를 실행하는 것과 동일한 동작을 수행한다. TIP 서비스를 호출할 경우, 필드 버퍼에 아래를 설정하여 TIPSVC를 호출한다.

항목	설명
TIP_OPERATION (string)	GET으로 설정한다.
TIP_SEGMENT (string)	ADMINISTRATION으로 설정한다.
TIP_SECTION (string)	CHLOG로 설정한다.
TIP_MODULE (int)	동적으로 로그를 변경하고 싶은 모듈을 설정하는 항목으로 다음 중 하나를 선택한다. • TIP_TMM • TIP_CLH • TIP_TMS • TIP_SVR

항목	설명
TIP_FLAGS (int)	flags를 설정하는 항목으로 다음 중 하나를 선택한다. • TIP_VFLAG • TIP_GFLGA • TIP_NFLAG
TIP_SVRNAME (string)	해당 서버명을 설정하는 항목으로 TIP_FLAGS를 TIP_VFLAG로 설정한 경우에만 설정한다.
TIP_SVGNAME (string)	해당 서버 그룹명을 설정하는 항목으로 TIP_FLAGS를 TIP_GFLAG로 설정한 경우에만 설정한다.
TIP_NODENAME (string)	해당 노드명으로 TIP_FLAGS를 TIP_NFLAG로 설정한 경우에만 설정한다.
TIP_LOGLVL (string)	변경하고자 하는 로그 레벨을 설정하는 항목으로 다음 중에 하나를 선택하고 반드시 소문자로 설정한다. 결과로 받아오는 값은 TIP_ERROR에 설정된다. • compact • basic • detail • debug1 • debug2 • debug3 • debug4
TIP_ERROR (int)	에러값이 설정되는 항목으로 설정되는 에러값은 다음과 같다. • TIPESVCFAIL : 해당 서비스를 제대로 처리하지 못한 경우 • TIPEOS : 메모리 할당에 실패한 경우 • TIPEBADFLD : TIP_MODULE 값을 설정하지 않은 경우

- CHTRC section

  TMS와 SPR의 TMAX_TRACE를 수정하여 trace log 옵션을 변경할 수 있는 섹션이다. tmadmin의 **chtrc**와 동일한 동작을 수행한다. TIP 서비스를 호출할 경우, 필드 버퍼에 아래를 설정하여 TIPSVC를 호출한다.

항목	설명
TIP_OPERATION (string)	GET으로 설정해야 한다.
TIP_SEGMENT (string)	ADMINISTRATION으로 설정해야 한다.
TIP_SECTION (string)	CHTRC로 설정해야 한다.

항목	설명
TIP_FLAGS (int)	flags를 설정하는 항목으로 다음 중에 하나를 설정한다. • TIP_PFLAG • TIP_VFLAG • TIP_GFLAG • TIP_NFLAG
TIP_SPRI (int)	spri를 설정하는 항목으로 TIP_FLAGS를 TIP_PFLAG로 설정한 경우에만 설정한다.
TIP_SVGNAME (string)	해당 서버 그룹명을 설정하는 항목으로 TIP_FLAGS를 TIP_GFLAG로 설정한 경우에만 설정한다.
TIP_NODENAME (string)	해당 노드명을 설정하는 항목으로 TIP_FLAGS를 TIP_NFLAG로 설정한 경우에만 설정한다.
TIP_SPEC (string)	변경하고자 하는 filter spec, receiver spec, trigger spec을 설정하는 항목이다. 결과로 받아오는 값은 TIP_ERROR에 설정된다.
TIP_ERROR (int)	에러값이 설정된다. 설정되는 에러값은 다음과 같다. • TIPESVCFAIL : 해당 서비스를 제대로 처리하지 못한 경우 • TIPEOS : 메모리 할당에 실패한 경우 • TIPEBADFLD : TIP_MODULE 값을 설정하지 않은 경우

다음은 TIPSVC에 요청하기 위하여 포함해야 할 데이터이다.

- 조직(TIP_OPERATION)

설정값	설명
GET	시스템의 정적인 환경 정보, 통계 정보의 확인, BOOT/DOWN 시스템 운용 관리할 경우에는 GET으로 설정한다.
SET	시스템 설정 상태를 변경하려고 하는 경우에 SET으로 설정한다. 현재는 GET만 사용된다.

- 세그먼트(TIP_SEGMENT)

  어떤 기능을 수행할지 결정하기 위하여 사용하는 필드로 다음은 TIP_SEGMENT 필드에 설정할 수 있는 값이다.

설정값	설명
CONFIGURATION	시스템의 정적인 설정 정보를 확인한다.
STATISTICS	시스템의 운영 중에 통계 정보를 확인한다.
ADMINISTRATION	시스템 운용 관리(BOOT/DOWN)를 확인한다.

- 절(TIP_SECTION)

TIP_SECTION을 설정했을 경우 어떤 항목에 대하여 처리를 할 것인지 세부 항목을 결정한다.

설정값	설명
CONFIGURATION	DOMAIN, NODE, SVRGROUP, SERVER, SERVICE, ROUTING, RQ, GATEWAY
STATISTICS	NODE, TPROC, SPR, SERVICE, RQ, TMGW, NTMGW, TMS, TMMS, CLHS, SERVER(SVR)
ADMINISTRATION	BOOT, DOWN, CHLOG, CHTRC

- 명령어(TIP_CMD)

  필드는 TIP_SECTION이 ADMINISTRATION일 경우에만 사용된다.

설정값	설명
TIP_BOOT	Tmax 시스템을 기동한다.
TIP_DOWN	Tmax 시스템을 종료한다.

5.5.2. TIP 사용

TIP의 사용 방법과 에러 확인하는 방법에 대해 설명한다.

TIP 사용 방법

- 환경 설정

  TIP 서버는 Tmax에서 제공하는 기능 프로세스이기 때문에 사용자가 서비스를 작성할 필요가 없다. 단, 환경 파일에 TIP 서버를 등록해야 한다. 다음은 환경 설정의 예제이다.

```
*DOMAIN
res             ..., TIPSVC = TIPSVC

*NODE
tmaxs1       ...

*SVRGROUP
tsvg            ..., SVGTYPE = TIP

*SERVER
TIP             SVGNAME = tsvg, SVRTYPE = SYS_SVR
```

  - DOMAIN 절에는 TIPSVC를 등록한다. 등록하지 않을 경우에는 기본값으로 TIPSVC가 등록된다.
  - SVRGROUP 절에는 TIP 서버 그룹(SVGTYPE=TIP)을 등록한다.
  - SERVER 절에는 TIP 서버(SVRTYPE=SYS_SVR)를 등록한다.

- 시스템 접속

사용자는 Tmax 시스템 접속할 때에 TPSTART_T 구조체의 usrname 항목에 .tpadmin을 설정해야 한다. TIP_SEGMENT가 ADMINISTRATION일 경우에만 usrname을 설정하며, 다른 경우에는 설정하지 않아도 된다. 잘못된 usrname이 설정되었을 경우에는 TIP_ERROR 필드에 TIPEAUTH 에러가 설정된다.

```
strcpy(tpinfo->usrname, “.tpadmin”);
```

- 버퍼 할당

  TIP 서버는 필드 키를 바탕으로 서비스를 처리하므로 요청하는 클라이언트나 서버 프로그램은 필드 키 버퍼를 할당해야 한다.

- TIP 요청 항목 설정

항목	설명
TIP_OPERATION	시스템 환경정보 변경 및 확인할 경우 설정한다.
TIP_SEGMENT	시스템 정보확인 및 운영관리를 위해서 설정한다.
TIP_SECTION	SEGMENT에서 설정항목에 대한 세부처리를 설정한다.
TIP_CMD	TIP_SEGMENT가 ADMINISTRATION일 경우만 설정한다.
TIP_NODENAME	멀티 노드일 경우만 설정하며, 단일 노드일 경우 지정하지 않으면 기본값으로 로컬 노드가 설정된다. 잘못 설정하였을 경우 TPEINVAL 에러가 발생한다.

- TIPSVC 요청

  TIP 요청 항목을 설정하였다면 설정한 필드 버퍼를 sndbuf로 하여, tpcall이나 tpacall을 이용하여 TIPSVC로 요청을 한다. 클라이언트나 서버에서 모두 요청할 수 있으며, 트랜잭션은 지원하지 않는다.

- 결과 수신

  수신 필드 키 버퍼에 서비스 결과가 저장된다.

에러 확인

- 정상 처리된 경우

  수신 필드 키 버퍼의 TIP_ERROR 항목에 0이 설정된다.

- 에러가 발생한 경우

에러	설명
TIP_STATUS	TIP_ERROR에 설정된 에러 내용에 대하여 상세한 에러 내용을 확인할 수 있다.
TIP_BADFIELD	에러를 유발시킨 필드를 확인할 수 있다.

에러	설명
TIP_ERROR	수신 필드 키 버퍼의 TIP_ERROR 항목에 0보다 큰 값이 설정된다. 이는 /usrinc/tip.h에서 확인할 수 있다. 다음은 TIP_ERROR에 설정되는 에러값에 대한 설명이다. • TIPNOERROR : 에러가 발생하지 않는다. • TIPEBADFLD : 유효하지 않은 필드 키가 사용되었다. 일반적으로 fdlc 유틸리티를 사용하여 컴파일되지 않은 필드 키가 사용되었을 경우 TIP_ERROR는 TIPEBADFLD로 설정된다. • TIPEIMPL : 구현되지 않은 기능에 대하여 요청한 경우이다. • TIPEAUTH : 현재의 권한으로는 허용되지 않는 서비스이다. • TIPEOS : 운영 시스템 또는 시스템 에러로서 메모리 할당 실패, Tmax 시스템에 접속 실패, 또는 네트워크 상태가 불안정할 경우 등에 나타나는 에러이다. • TIPENOENT : 존재하지 않는 항목을 접근하는 경우에 발생한다. • TIPESVCFAIL : TIP 서비스 루틴이 에러가 발생하여 TPFAIL로 tpreturn()을 호출하였다.

5.5.3. TIP 사용 예제

본 절은 TIP를 사용한 예제를 설명한다.

환경 파일

다음은 단일 노드를 사용한 예제이다.

<cfg.m>

```
*DOMAIN
res          SHMKEY=78850,     MAXUSER=200, MINCLH=1, MAXCLH=5,
             TPORTNO=8850,     BLOCKTIME=60, TXTIME=50, RACPORT=3355

*NODE
tmaxh4         TMAXDIR="/data1/starbj81/tmax",
               APPDIR="/data1/starbj81/tmax/appbin",
               PATHDIR ="/data1/starbj81/tmax/path",
               TLOGDIR ="/data1/starbj81/tmax/log/tlog",
               ULOGDIR="/data1/starbj81/tmax/log/ulog",
               SLOGDIR="/data1/starbj81/tmax/log/slog"

*SVRGROUP
tsvg          NODENAME = "tmaxh4", SVGTYPE=TIP
svg1          NODENAME = "tmaxh4"

*SERVER
TIP            SVGNAME=tsvg, SVRTYPE=SYS_SVR, MIN=1, MAX=1
svr            SVGNAME=svg1, MIN=1

*SERVICE
```

```
TOUPPER         SVRNAME=svr
```

다음은 멀티 노드를 사용한 예제이다.

<cfg.m>

```
*DOMAIN
tmax           SHMKEY=98850,
               TPORTNO=8850,
               BLOCKTIME=60,
               RACPORT=3355,
               MAXUSER=10

*NODE
Tmaxh4          TMAXDIR="/data1/starbj81/tmax",
                APPDIR="/data1/starbj81/tmax/appbin",
                PATHDIR = "/data1/starbj81/tmax/path",
                TLOGDIR = "/data1/starbj81/tmax/log/tlog",
                ULOGDIR="/data1/starbj81/tmax/log/ulog",
                SLOGDIR="/data1/starbj81/tmax/log/slog"

tmaxh2          TMAXDIR="/data1/starbj81/tmax",
                APPDIR="/data1/starbj81/tmax/appbin",
                PATHDIR = "/data1/starbj81/tmax/path",
                TLOGDIR = "/data1/starbj81/tmax/log/tlog",
                ULOGDIR="/data1/starbj81/tmax/log/ulog",
                SLOGDIR="/data1/starbj81/tmax/log/slog"

*SVRGROUP
tsvg           NODENAME = "tmaxh4", SVGTYPE=TIP

svg1           NODENAME = "tmaxh4", COUSIN = "svg2"
svg2           NODENAME = "tmaxh2"


*SERVER
TIP           SVGNAME=tsvg, SVRTYPE=SYS_SVR, MIN=1, MAX=1
svr           SVGNAME=svg1, MIN=1, MAX=5

*SERVICE
TOUPPER        SVRNAME=svr, ROUTING = "rout1"

*ROUTING
rout1           FIELD="STRING", RANGES = "'bbbbbbb'-'ccccccc' : svg1, * :
                svg2
```

필드 키 테이블

다음은 필드 키 테이블의 예제이다.

< tip.f >

```
#
#       common field
```

```
#
# name          number  type    flags   comments
*base 16000001
TIP_OPERATION   0       string
TIP_SEGMENT     1       string
TIP_SECTION     2       string
TIP_NODE        3       string
TIP_OCCURS      4       int
TIP_FLAGS       5       int
TIP_CURSOR      6       string
TIP_SESTM       7       int
TIP_ERROR       8       int
TIP_STATE       9       int
TIP_MORE        10      int
TIP_BADFIELD    11      string
TIP_CMD         12      string
TIP_CLID        13      int
TIP_MSG         14      string

#
#       DOMAIN section fields
#
# name                  number  type    flags   comments
*base 16000100
TIP_NAME                0       string
TIP_SHMKEY              1       int
TIP_MINCLH              2       int
TIP_MAXCLH              3       int
TIP_MAXUSER             4       int
TIP_TPORTNO             5       int
TIP_RACPORT             6       int
TIP_MAXSACALL           7       int
TIP_MAXCACALL           8       int
TIP_MAXCONV_NODE        9       int
TIP_MAXCONV_SERVER      10      int
TIP_CMTRET              11      int
TIP_BLOCKTIME           12      int
TIP_TXTIME              13      int
TIP_IDLETIME            14      int
TIP_CLICHKINT           15      int
TIP_NLIVEINQ            16      int
TIP_SECURITY            17      string
TIP_OWNER               18      string
TIP_CPC                 19      int
#TIP_LOGINSVC           20      string
#TIP_LOGOUTSVC          21      string
TIP_NCLHCHKTIME         22      int
TIP_DOMAINID            23      int
TIP_IPCPERM             24      int
TIP_MAXNODE             25      int
TIP_MAXSVG              26      int
TIP_MAXSVR              27      int
TIP_MAXSVC              28      int
TIP_MAXSPR              29      int
TIP_MAXTMS              30      int
TIP_MAXCPC              31      int
TIP_MAXROUT             32      int
TIP_MAXROUTSVG          33      int
TIP_MAXRQ               34      int
```

```
TIP_MAXGW               35      int
TIP_MAXCOUSIN           36      int
TIP_MAXCOUSINSVG        37      int
TIP_MAXBACKUP           38      int
TIP_MAXBACKUPSVG        39      int
TIP_MAXTOTALSVG         40      int
TIP_MAXPROD             41      int
TIP_MAXFUNC             42      int
TIP_TXPENDINGTIME       43      int
TIP_NO                  44      int
TIP_TIPSVC              45      string
TIP_NODECOUNT           46      int
TIP_SVGCOUNT            47      int
TIP_SVRCOUNT            48      int
TIP_SVCCOUNT            49      int
TIP_COUSIN_COUNT                50      int
TIP_BACKUP_COUNT                51      int
TIP_ROUT_COUNT          52      int
TIP_STYPE               53      string
TIP_VERSION             54      string
TIP_EXPDATE             55      string
TIP_DOMAINCOUNT         56      int
TIP_RSVG_GCOUNT         57      int
TIP_RSVG_COUNT          58      int
TIP_CSVG_GCOUNT         59      int
TIP_CSVG_COUNT          60      int
TIP_BSVG_GCOUNT         61      int
TIP_BSVG_COUNT          62      int
TIP_PROD_COUNT          63      int
TIP_FUNC_COUNT          64      int
TIP_SHMSIZE             65      int
TIP_CRYPT               66      string
TIP_DOMAIN_TMMLOGLVL    67      string
TIP_DOMAIN_CLHLOGLVL    68      string
TIP_DOMAIN_TMSLOGLVL    69      string
TIP_DOMAIN_LOGLVL       70      string
TIP_DOMAIN_MAXTHREAD    71      int

#
#       NODE section fields
#
# name          number  type    flags   comments
*base 16000200
#TIP_NAME       0       string
TIP_DOMAINNAME  1       string
#TIP_SHMKEY     2       int
#TIP_MINCLH     3       int
#TIP_MAXCLH     4       int
TIP_CLHQTIMEOUT 5       int
#TIP_IDLETIME   6       int
#TIP_CLICHKINT  7       int
#TIP_TPORTNO    8       int
#TIP_TPORTNO2   9       int
#TIP_TPORTNO3   10      int
#TIP_TPORTNO4   11      int
#TIP_TPORTNO5   12      int
#TIP_RACPORT    13      int
#TIP_TMAXPORT   14      string
TIP_CMPRPORT    15      string
```

```
TIP_CMPRSIZE    16      int
#TIP_MAXUSER    17      int
TIP_TMAXDIR     18      string
TIP_TMAXHOME    19      string
TIP_APPDIR      20      string
TIP_PATHDIR     21      string
TIP_TLOGDIR     22      string
TIP_SLOGDIR     23      string
TIP_ULOGDIR     24      string
TIP_ENVFILE     25      string
#TIP_LOGINSVC   26      string
#TIP_LOGOUTSVC  27      string
TIP_IP          28      string
#TIP_PEER       29      string
TIP_TMMOPT      30      string
TIP_CLHOPT      31      string
#TIP_IPCPERM    32      int
#TIP_MAXSVG     33      int
#TIP_MAXSVR     34      int
#TIP_MAXSPR     35      int
#TIP_MAXTMS     36      int
#TIP_MAXCPC     37      int
TIP_MAXGWSVR    38      int
TIP_MAXRQSVR    39      int
TIP_MAXGWCPC    40      int
TIP_MAXRQCPC    41      int
TIP_CPORTNO     42      int
TIP_REALSVR     43      string
TIP_RSCPC       44      int
TIP_AUTOBACKUP  45      int
TIP_HOSTNAME    46      string
TIP_NODETYPE    47      int
TIP_CPU         48      int
#TIP_MAXRSTART  49      int
#TIP_GPERIOD    50      int
#TIP_RESTART    51      int
TIP_CURCLH      49      int
TIP_LIVECTIME   50      string
TIP_NODE_TMMLOGLVL      51      string
TIP_NODE_CLHLOGLVL      52      string
TIP_NODE_TMSLOGLVL      53      string
TIP_NODE_LOGLVL         54      string
TIP_NODE_MAXTHREAD      55      int
TIP_EXTPORT             56      int
TIP_EXTCLHPORT          57      int
TIP_MSGSIZEWARN 58      int
TIP_MSGSIZEMAX  59      int


#
#       SVRGROUP section fields
#
# name          number  type    flags   comments
*base 16000300
#TIP_NAME       0       string
#TIP_NODENAME   1       string
TIP_SVGTYPE     2       string
#TIP_PRODNAME   3       string
TIP_COUSIN      4       string
```

```
TIP_BACKUP      5       string
TIP_LOAD        6       int
#TIP_APPDIR     7       string
#TIP_ULOGDIR    8       string
TIP_DBNAME      9       string
TIP_OPENINFO    10      string
TIP_CLOSEINFO   11      string
TIP_MINTMS      12      int
#TIP_MAXTMS     13      int
TIP_TMSNAME     14      string
#TIP_SECURITY   15      string
#TIP_OWNER      16      string
#TIP_ENVFILE    17      string
#TIP_CPC                18      int
TIP_XAOPTION            19      string
TIP_SVG_TMSTYPE         20      string
TIP_SVG_TMSOPT          21      string
TIP_SVG_TMSTHREADS      22      int
TIP_SVG_TMSLOGLVL       23      string
TIP_SVG_LOGLVL          24      string
TIP_NODENAME            25      string

#
#       SERVER section fields
#
# name          number  type    flags   comments
*base 16000350
#TIP_NAME       0       string
TIP_SVGNAME     1       string
#TIP_NODENAME   2       string
TIP_CLOPT       3       string
TIP_SEQ         4       int
TIP_MIN         5       int
TIP_MAX         6       int
#TIP_ULOGDIR    7       string
TIP_CONV        8       int
TIP_MAXQCOUNT   9       int
TIP_ASQCOUNT    10      int
TIP_MAXRSTART   11      int
TIP_GPERIOD     12      int
TIP_RESTART     13      int
TIP_SVRTYPE     14      string
#TIP_CPC                15      int
TIP_SCHEDULE    16      int
#TIP_MINTHR     17      int
#TIP_MAXTHR     18      int
TIP_TARGET      19      string
TIP_DEPEND      20      string
TIP_CASCADE     21      int
TIP_PROCNAME    22      string
TIP_LIFESPAN    23      string
TIP_DDRI        24      string
TIP_CURSVR      25      int
TIP_SVGNO       26      int
TIP_SVR_LOGLVL  27      string


#
#       SERVICE section fields
```

```
#
# name          number  type    flags   comments
*base 16000400
#TIP_NAME       0       string
TIP_SVRNAME     1       string
TIP_PRI         2       int
TIP_SVCTIME     3       int
TIP_ROUTING     4       string
TIP_EXPORT      5       int
TIP_AUTOTRAN    6       int

#
#       ROUTING section fields
#
# name          number  type    flags   comments
*base 16000425
#TIP_NAME       0       string
TIP_FLDTYPE     1       string
TIP_RANGES      2       string
TIP_SUBTYPE     3       string
TIP_ELEMENT     4       string
TIP_BUFTYPE     5       string
TIP_OFFSET      6       int
TIP_FLDLEN      7       int
#TIP_FLDOFFSET  8       int


#
#       RQ section fields
#
# name          number  type    flags   comments
*base 16000450
#TIP_NAME       0       string
#TIP_SVGNAME    1       string
TIP_PRESVC      2       string
TIP_QSIZE       3       int
TIP_FILEPATH    4       string
TIP_BOOT        5       string
TIP_FSYNC       6       int
TIP_BUFFERING   7       int
#TIP_ENQSVC     8       int
#TIP_FAILINTERVAL 9     int
#TIP_FAILRETRY  10      int
#TIP_FAILSVC    11      string
#TIP_AFTERSVC   12      string


#
#       GATEWAY section fields
#
# name                  number  type    flags   comments
*base 16000500
#TIP_NAME               0       string
TIP_GWTYPE              1       string
TIP_PORTNO              2       int
#TIP_CPC                        3       int
TIP_RGWADDR             4       string
TIP_RGWPORTNO           5       int
#TIP_BACKUP             6       string
```

```
#TIP_NODENAME           7       string
TIP_KEY                 8       string
TIP_BACKUP_RGWADDR      9       string
TIP_BACKUP_RGWPORTNO    10      int
TIP_TIMEOUT             11      int
TIP_DIRECTION           12      string
TIP_MAXINRGW            13      int
TIP_GWOWNER             15      string
TIP_RGWOWNER            16      string
TIP_RGWPASSWD           17      string
TIP_PTIMEOUT            18      int
TIP_PTIMEINT            19      int

#
#       FUNCTION section fields
#
# name          number  type    flags   comments
*base 16000550
#TIP_NAME       0       string
#TIP_SVRNAME    1       string
TIP_FQSTART     2       int
TIP_FQEND       3       int
TIP_ENTRY       4       string


#
#       STATISTICS segment fields
#
# name          number  type    flags   comments
*base 16000600
#TIP_NAME       0       string
TIP_STATUS      1       string
TIP_STIME       2       string
TIP_TTIME       3       int
TIP_SVC_STIME   4       int
TIP_COUNT       5       int
#TIP_NO         6       int
TIP_NUM_FREE    7       int
TIP_NUM_REPLY   8       int
TIP_NUM_FAIL    9       int
TIP_NUM_REQ     10      int
TIP_ENQ_REQS    11      int
TIP_DEQ_REQS    12      int
TIP_ENQ_REPLYS  13      int
TIP_DEQ_REPLYS  14      int
TIP_CLHNO       15      int
TIP_SVR_NAME    16      string
TIP_SVC_NAME    17      string
TIP_AVERAGE     18      float
TIP_QCOUNT      19      int
TIP_CQCOUNT     20      int
TIP_QAVERAGE    21      float
TIP_MINTIME     22      float
TIP_MAXTIME     23      float
TIP_FAIL_COUNT  24      int
TIP_ERROR_COUNT 25      int
TIP_PID         26      int
TIP_TOTAL_COUNT 27      int
TIP_TOTAL_SVCFAIL_COUNT 28      int
```

```
TIP_TOTAL_ERROR_COUNT   29      int
TIP_TOTAL_AVG   30      float
TIP_TOTAL_RUNNING_COUNT 31      int
TIP_TMS_NAME    32      string
TIP_SVG_NAME    33      string
TIP_SPRI        34      int
TIP_TI_THRI     35      int
TIP_TI_AVG      36      float
TIP_TI_XID      37      string
TIP_TI_XA_STATUS        38      string
TIP_GW_NAME     39      string
TIP_GW_NO       40      int
TIP_GW_HOSTN    41      string
TIP_GW_CTYPE    42      string
TIP_GW_CTYPE2   43      string
TIP_GW_IPADDR   44      string
TIP_GW_PORT     45      int
TIP_GW_STATUS   46      string

#
#       ADMIN segment fields
#
# name          number  type    flags   comments
*base 16000650
TIP_IPADDR      0       string
TIP_USRNAME     1       string
TIP_MODULE      2       int
TIP_LOGLVL      3       string
TIP_SPEC        4       string

#
#       boot time
#
# name          number  type    flags   comments
TIP_BOOTTIME_SEC        5       int
TIP_BOOTTIME_MSEC       6       int

#
#       EXTRA flag fields
#
# name          number  type    flags   comments
*base 16000700
TIP_EXTRA_OPTION        0       int
TIP_SVRI        1       int
TIP_QPCOUNT     2       int
TIP_EMCOUNT     3       int
TIP_SVR_STATUS  4       string
```

5.5.4. 시스템 환경 정보 조회 프로그램

다음은 시스템의 환경 정보를 확인하는 클라이언트 프로그램의 예제이다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <usrinc/tip.h>

#define SEC_DOMAIN          1
#define SEC_NODE            2
#define SEC_SVGROUP         3
#define SEC_SERVER          4
#define SEC_SERVICE         5
#define SEC_ROUTING         6
#define SEC_RQ              7
#define SEC_GATEWAY         8

main(int argc, char *argv[])
{
        FBUF    *sndbuf, *rcvbuf;
        TPSTART_T *tpinfo;
        int   i, n, sect;
        long  rcvlen;
        char  nodename[NAMELEN];
        int   pid, count = 0;

        if (argc != 3) {
            printf("Usage: %s section nodename\n", argv[0]);
            printf("section:\n");
            printf("\t1: domain\n");
            printf("\t2: node\n");
            printf("\t3: svrgroup\n");
            printf("\t4: server\n");
            printf("\t5: service\n");
            printf("\t6: routing\n");
            printf("\t7: rq\n");
            printf("\t8: gateway\n");
            exit(1);
        }

        if (!isdigit(argv[3][0])) {
            printf("fork count must be a digit\n");
            exit(1);
        }
        count = atoi(argv[3]);

        sect = atoi(argv[1]);
        if (sect < SEC_DOMAIN || sect > SEC_GATEWAY) {
            printf("out of section [%d - %d]\n", SEC_DOMAIN, SEC_GATEWAY);
            exit(1);
        }

        strncpy(nodename, argv[2], sizeof(nodename) - 1);

        n = tmaxreadenv("tmax.env", "TMAX");
        if (n < 0) {
           fprintf(stderr, "can't read env\n");
           exit(1);
        }

        tpinfo = (TPSTART_T *)tpalloc("TPSTART", NULL, 0);
        if (tpinfo == NULL) {
            printf("tpalloc fail tperrno = %d\n", tperrno);
```

```
        exit(1);
    }
    strcpy(tpinfo->usrname, ".tpadmin");
    if (tpstart((TPSTART_T *)tpinfo) == -1){
        printf("tpstart fail [%s]\n", tpstrerror(tperrno));
        exit(1);
    }

    if ((sndbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    if ((rcvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
        printf("tpalloc failed! errno = %d\n", tperrno);
        tpend();
        exit(1);
    }

    n = fbput(sndbuf, TIP_OPERATION, "GET", 0);
    n = fbput(sndbuf, TIP_SEGMENT, "CONFIGURATION", 0);
    switch (sect) {
        case SEC_DOMAIN:
            n = fbput(sndbuf, TIP_SECTION, "DOMAIN", 0);
            break;
        case SEC_NODE:
            n = fbput(sndbuf, TIP_SECTION, "NODE", 0);
            break;
        case SEC_SVGROUP:
            n = fbput(sndbuf, TIP_SECTION, "SVGROUP", 0);
            break;
        case SEC_SERVER:
            n = fbput(sndbuf, TIP_SECTION, "SERVER", 0);
            break;
        case SEC_SERVICE:
            n = fbput(sndbuf, TIP_SECTION, "SERVICE", 0);
            break;
        case SEC_ROUTING:
            n = fbput(sndbuf, TIP_SECTION, "ROUTING", 0);
            break;
        case SEC_RQ:
            n = fbput(sndbuf, TIP_SECTION, "RQ", 0);
            break;
        case SEC_GATEWAY:
            n = fbput(sndbuf, TIP_SECTION, "GATEWAY", 0);
            break;
    }

    n = fbput(sndbuf, TIP_NODENAME, nodename, 0);

    n = tpcall("TIPSVC", (char *)sndbuf, 0, (char **)&rcvbuf,  &rcvlen,
                TPNOFLAGS);

    if (n < 0) {
            printf("tpcall fail [%s]\n", tpstrerror(tperrno));
            fbprint(rcvbuf);
            tpfree((char *)sndbuf);
            tpfree((char *)rcvbuf);
```

```
                tpend();
                exit(1);
        }

#if 1
        fbprint(rcvbuf);
#endif

        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
}
```

[결과]

다음은 해당 프로그램의 결과(Domain Conf)이다.

```
$ client 1 tmaxh4
fkey = 217326601, fname = TIP_ERROR, type = int, value = 0
...
fkey = 485762214, fname = TIP_CRYPT, type = string, value = NO
    fkey = 485762215, fname = TIP_DOMAIN_TMMLOGLVL, type = string, value = DEBUG1
    fkey = 485762216, fname = TIP_DOMAIN_CLHLOGLVL, type = string, value = DEBUG2
    fkey = 485762217, fname = TIP_DOMAIN_TMSLOGLVL, type = string, value = DEBUG3
    fkey = 485762218, fname = TIP_DOMAIN_LOGLVL, type = string, value = DEBUG4
    fkey = 217326763, fname = TIP_DOMAIN_MAXTHREAD, type = int, value = 128
```

5.5.5. 시스템 통계 정보 조회 프로그램

다음은 시스템의 통계 정보를 확인하는 프로그램의 예제이다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <usrinc/tip.h>

#define SEC_NODE        1
#define SEC_TPROC       2
#define SEC_SPR         3
#define SEC_SERVICE     4
#define SEC_RQ          5
#define SEC_TMS         6
#define SEC_TMMS        7
#define SEC_CLHS        8
#define SEC_SERVER        9

#define NODE_NAME_SIZE 32

main(int argc, char *argv[])
{
        FBUF    *sndbuf, *rcvbuf;
```

```
        TPSTART_T *tpinfo;
        int   i, n, sect;
        long  rcvlen;
        char  nodename[NODE_NAME_SIZE];
        int stat;

        if (argc != 3) {
            printf("Usage: %s section node\n", argv[0]);
            printf("section:\n");
            printf("\t1: node\n");
            printf("\t2: tproc\n");
            printf("\t3: spr\n");
            printf("\t4: service\n");
            printf("\t5: rq\n");
            printf("\t6: tms\n");
            printf("\t7: tmms\n");
            printf("\t8: clhs\n");
            printf("\t9: server\n");
            exit(1);
        }

        sect = atoi(argv[1]);
        if (sect < SEC_NODE || sect > SEC_SERVER) {
            printf("out of section [%d - %d]\n",SEC_NODE, SEC_SERVER);
            exit(1);
        }

        memset(nodename, 0x00, NODE_NAME_SIZE);
        strncpy(nodename, argv[2], NODE_NAME_SIZE - 1);

        n = tmaxreadenv("tmax.env", "TMAX");
        if (n < 0) {
           fprintf(stderr, "can't read env\n");
           exit(1);
        }

        tpinfo = (TPSTART_T *)tpalloc("TPSTART", NULL, 0);
        if (tpinfo == NULL) {
            printf("tpalloc fail tperrno = %d\n", tperrno);
            exit(1);
        }
        strcpy(tpinfo->dompwd, "xamt123");

        if (tpstart((TPSTART_T *)tpinfo) == -1){
            printf("tpstart fail tperrno = %d\n", tperrno);
            exit(1);
        }

        if ((sndbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
            printf("tpalloc failed! errno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        if ((rcvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
            printf("tpalloc failed! errno = %d\n", tperrno);
            tpend();
            exit(1);
        }
```

```
        n = fbput(sndbuf, TIP_OPERATION, "GET", 0);
        n = fbput(sndbuf, TIP_SEGMENT, "STATISTICS", 0);
        switch (sect) {
            case SEC_NODE:
                n = fbput(sndbuf, TIP_SECTION, "NODE", 0);
                break;
            case SEC_TPROC:
                n = fbput(sndbuf, TIP_SECTION, "TPROC", 0);
                break;
            case SEC_SPR:
                n = fbput(sndbuf, TIP_SECTION, "SPR", 0);
                break;
            case SEC_SERVICE:
                n = fbput(sndbuf, TIP_SECTION, "SERVICE", 0);
                break;
            case SEC_RQ:
                n = fbput(sndbuf, TIP_SECTION, "RQ", 0);
                break;
            case SEC_TMS:
                stat = 1;
                n = fbput(sndbuf, TIP_SECTION, "TMS", 0);
                n = fbput(sndbuf, TIP_EXTRA_OPTION, (char *)&stat, 0);
                break;
            case SEC_TMMS:
                n = fbput(sndbuf, TIP_SECTION, "TMMS", 0);
                break;
            case SEC_CLHS:
                n = fbput(sndbuf, TIP_SECTION, "CLHS", 0);
                break;
            case SEC_SERVER:
                n = fbput(sndbuf, TIP_SECTION, "SERVER", 0);
                break;
        }
        n = fbput(sndbuf, TIP_NODENAME, nodename, 0);

        n = tpcall("TIPSVC", (char *)sndbuf, 0, (char **)&rcvbuf,
                   &rcvlen, TPNOFLAGS);
        if (n < 0) {
            printf("tpcall fail [%s]\n", tpstrerror(tperrno));
            fbprint(rcvbuf);
            tpfree((char *)sndbuf);
            tpfree((char *)rcvbuf);
            tpend();
            exit(1);
        }

        fbprint(rcvbuf);

        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
}
```

[결과]

다음은 해당 프로그램의 결과(TMS STATISTICS)를 나타내는 예제이다.

```
$  client 3000 1 2
fkey = 217326601, fname = TIP_ERROR, type = int, value = 0
fkey = 485762680, fname = TIP_TMS_NAME, type = string, value = tms_ora2
fkey = 485762681, fname = TIP_SVG_NAME, type = string, value = xa1
fkey = 217327226, fname = TIP_SPRI, type = int, value = 0
fkey = 485762649, fname = TIP_STATUS, type = string, value = RUN
fkey = 217327197, fname = TIP_COUNT, type = int, value = 0
fkey = 351544938, fname = TIP_AVERAGE, type = float, value = 0.000000
fkey = 217327212, fname = TIP_CQCOUNT, type = int, value = 0
fkey = 217327227, fname = TIP_TI_THRI, type = int, value = 1
fkey = 351544956, fname = TIP_TI_AVG, type = float, value = 0.000000
fkey = 485762685, fname = TIP_TI_XID, type = string, value = 00000013664
fkey = 485762686, fname = TIP_TI_XA_STATUS, type = string, value = COMMIT
```

5.5.6. 서버 프로세스 기동 및 종료 프로그램

예제1

다음은 서버 프로세스를 추가로 기동하거나 종료하는 프로그램 예제이다.

< cli.c >

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <usrinc/tmaxapi.h>
#include <usrinc/tip.h>

#define NODE_NAME_SIZE  32

main(int argc, char *argv[])
{
        FBUF    *sndbuf, *rcvbuf;
        TPSTART_T *tpinfo;
        int   i, n, type, clid, count, flags;
        long  rcvlen;
        char  svrname[TMAX_NAME_SIZE];
        char  svgname[TMAX_NAME_SIZE];
        char  nodename[NODE_NAME_SIZE];
        int   pid, forkcnt;

        if (argc != 6) {
            printf("Usage: %s type svrname count nodename forkcnt\n", argv[0]);
            printf("type 1: BOOT, 2: DOWN, 3: DISCON\n");
            exit(1);
        }

        type = atoi(argv[1]);
        if ((type != 1) && (type != 2) && (type != 3)) {
            printf("couldn't support such a type %d\n", type);
            exit(1);
```

```
        }

        if (strlen(argv[2]) >= TMAX_NAME_SIZE) {
            printf("too large name [%s]\n", argv[1]);
            exit(1);
        }
        strcpy(svrname, argv[2]);
        count = atoi(argv[3]);
        flags = 0;

        strncpy(nodename, argv[4], NODE_NAME_SIZE - 1);
        forkcnt = atoi(argv[5]);

        n = tmaxreadenv("tmax.env", "TMAX");
        if (n < 0) {
           fprintf(stderr, "can't read env\n");
           exit(1);
        }

        tpinfo = (TPSTART_T *)tpalloc("TPSTART", NULL, 0);
        if (tpinfo == NULL) {
            printf("tpalloc fail tperrno = %d\n", tperrno);
            exit(1);
        }
        strcpy(tpinfo->usrname, ".tpadmin");

        for (i = 1; i < forkcnt; i++) {
            if ((pid = fork()) < 0)
                exit(1);
            else if (pid == 0)
                break;
        }

        if (tpstart((TPSTART_T *)tpinfo) == -1){
            printf("tpstart fail tperrno = %d\n", tperrno);
            exit(1);
        }

        if ((sndbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
            printf("tpalloc failed! errno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        if ((rcvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
            printf("tpalloc failed! errno = %d\n", tperrno);
            tpend();
            exit(1);
        }

        n = fbput(sndbuf, TIP_OPERATION, "GET", 0);
        n = fbput(sndbuf, TIP_SEGMENT, "ADMINISTRATION", 0);
        if (type == 1)
            n = fbput(sndbuf, TIP_CMD, "BOOT", 0);
        else if (type == 2)
            n = fbput(sndbuf, TIP_CMD, "DOWN", 0);
        else
            n = fbput(sndbuf, TIP_CMD, "DISCON", 0);
```

```
        if (type == 3) {
            clid = count;       /* at type 3 */
            flags |= TIP_SFLAG;
            n = fbput(sndbuf, TIP_CLID, (char *)&clid, 0);
            n = fbput(sndbuf, TIP_FLAGS, (char *)&flags, 0);
        } else {
            flags |= TIP_SFLAG;
            n = fbput(sndbuf, TIP_SVRNAME, svrname, 0);
            n = fbput(sndbuf, TIP_COUNT, (char *)&count, 0);
            n = fbput(sndbuf, TIP_FLAGS, (char *)&flags, 0);
        }

        n = fbput(sndbuf, TIP_NODENAME, nodename, 0);

        n = tpcall("TIPSVC", (char *)sndbuf, 0, (char **)&rcvbuf,
                   &rcvlen, TPNOFLAGS);
        if (n < 0) {
           printf("tpcall failed! errno = %d[%s]\n", tperrno, tpstrerror(tperrno));
           fbprint(rcvbuf);
           tpfree((char *)sndbuf);
           tpfree((char *)rcvbuf);
           tpend();
           exit(1);
        }

        fbprint(rcvbuf);

        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
```

예제2

다음은 svr23_stat_ins 이라는 서버의 log level을 debug4로 변경하는 예제이다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <usrinc/atmi.h>
#include <usrinc/fbuf.h>
#include <usrinc/tmaxapi.h>
#include <usrinc/tip.h>
#include "../fdl/tip_fdl.h"

#define NFLAG 32
#define GFLAG 8
#define VFLAG 1024

int case_chlog(int, char *[], FBUF *);

#define NODE_NAME_SIZE  32

main(int argc, char *argv[])
{
      FBUF    *sndbuf, *rcvbuf;
      TPSTART_T *tpinfo;
```

```
int   i, ret, n, type, clid, count, flags = 0;
long  rcvlen;
char  svrname[TMAX_NAME_SIZE];
char  svgname[TMAX_NAME_SIZE];
char  nodename[NODE_NAME_SIZE];
int   pid, forkcnt;

if (argc < 6) {
      printf("Usage: %s svgname svrname nodename  [chlogmodule] [flags]
                                                  [loglvl]\n", argv[0]);
      printf("chlogmodule 1: TIP_TMM, 2: TIP_CLH, 4: TIP_TMS, 8: TIP_SVR\n");
      printf("flags 1: NFLAGS, 2: GFLAGS, 3: VFLAGS\n");
      printf("loglvl : 1: compact, 2: basic, 3: detail, 4: debug1,
              5: debug2, 6: debug3, 7: debug4\n");
      exit(1);
}

  n = tmaxreadenv("tmax.env", "TMAX");
  if (n < 0) {
     fprintf(stderr, "can't read env\n");
     exit(1);
  }

  tpinfo = (TPSTART_T *)tpalloc("TPSTART", NULL, 0);
  if (tpinfo == NULL) {
      printf("tpalloc fail tperrno = %d\n", tperrno);
      exit(1);
  }
  strcpy(tpinfo->usrname, ".tpadmin");
  strcpy(svgname, argv[1]);
  strcpy(svrname, argv[2]);
  strncpy(nodename, argv[3], NODE_NAME_SIZE - 1);

  if (tpstart((TPSTART_T *)tpinfo) == -1){
      printf("tpstart fail tperrno = %d\n", tperrno);
      exit(1);
  }

  if ((sndbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
      printf("tpalloc failed! errno = %d\n", tperrno);
      tpend();
      exit(1);
  }

  if ((rcvbuf = (FBUF *)tpalloc("FIELD", NULL, 0)) == NULL) {
      printf("tpalloc failed! errno = %d\n", tperrno);
      tpend();
      exit(1);
  }

  ret = case_chlog(argc, argv, sndbuf);
  n = fbput(sndbuf, TIP_OPERATION, "GET", 0);
  n = fbput(sndbuf, TIP_SEGMENT, "ADMINISTRATION", 0);
  n = fbput(sndbuf, TIP_CMD, "CHLOG", 0);
  n = fbput(sndbuf, TIP_NODENAME, nodename, 0);
  n = fbput(sndbuf, TIP_SVGNAME, svgname , 0);
  n = fbput(sndbuf, TIP_SVRNAME, svrname, 0);

  n=tpcall("TIPSVC", (char *)sndbuf, 0, (char **)&rcvbuf, &rcvlen, TPNOFLAGS);
```

```
        if (n < 0) {
             printf("tpcall failed! errno = %d[%s]\n", tperrno,
                     tpstrerror(tperrno));
             fbprint(rcvbuf);
             tpfree((char *)sndbuf);
             tpfree((char *)rcvbuf);
             tpend();
             exit(1);
        }

        fbprint(rcvbuf);
        tpfree((char *)sndbuf);
        tpfree((char *)rcvbuf);
        tpend();
}

int case_chlog(int argc2, char *argv2[], FBUF *sndbuf)
{
        int  chlogmdl, loglvl, flags, n=0;
        char  cloglvl[TMAX_NAME_SIZE];
        const int true = 1, false = 0;

        chlogmdl = atoi(argv2[4]);
        if( (chlogmdl != 1) && (chlogmdl != 2) && (chlogmdl != 4) &&
            (chlogmdl != 8)
        {
                printf("couldn't support such a chlogmdl\n");
                exit(1);
        }

        flags = atoi(argv2[5]);
        if( (flags != NFLAG) && (flags != GFLAG) && (flags != VFLAG) )
        {
                printf("couldn't support such a flags\n");
                exit(1);
        }

        loglvl = atoi(argv2[6]);
        if( (loglvl < 1) || (loglvl > 7) )
        {
                printf("couldn't support such a loglvl\n");
                exit(1);
        }

        switch (loglvl)
        {
                case 1 :
                        strcpy(cloglvl, "compact");
                        break;
                case 2 :
                        strcpy(cloglvl, "basic");
                        break;
                case 3 :
                        strcpy(cloglvl, "detail");
                        break;
                case 4 :
                        strcpy(cloglvl, "debug1");
                        break;
                case 5 :
```

```
                        strcpy(cloglvl, "debug2");
                        break;
                case 6 :
                        strcpy(cloglvl, "debug3");
                        break;
                case 7 :
                        strcpy(cloglvl, "debug4");
                        break;
        }
        n = fbput(sndbuf, TIP_MODULE, (char *)&chlogmdl, 0);
        n = fbput(sndbuf, TIP_FLAGS, (char *)&flags , 0);
        n = fbput(sndbuf, TIP_LOGLVL, cloglvl , 0);
        return 1;
}
```

[결과] (TIP_SVR, => DEBUG4)

```
$ client  xa1 svr23_stat_ins $HOSTNAME 8 1024 7
fkey = 217326601, fname = TIP_ERROR, type = int, value = 0
>>> tmadmin (cfg -v)

         loglvl = DEBUG4
```

5.6. Local recursive call

서버에서 tpcall()을 수행할 경우 같은 서버에 존재하는 서비스일 경우에도 내부에서 Recursive하게 호출할 수 있도록 기능이 추가되었다. 이는 서버에서 Multicontext 기법을 통해 tpcall()에 한해서 Local Recursive call이 가능하도록 한 것으로 무한 loop를 방지하기 위해서 최대 depth는 8로 제한한다.

Local Recursive call을 사용하기 위해서는 서버 프로그램을 컴파일할 때 반드시 CFLAGS에 –D_MCONTEXT를 추가해야 하며, libsvr.so 대신 libsvrmc.so 서버 라이브러리를 이용해 컴파일해야 한다.

서버 프로그램

다음은 서버 프로그램의 예제이다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <usrinc/atmi.h>


SVC15004_1(TPSVCINFO *msg)
{
         int     i;
         char    *rcvbuf;
         long    rcvlen;
```

```
        if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL)
             printf("rcvbuf tpalloc fail[%s]\n",tpstrerror(tperrno));
        if (tpcall("SVC15004_2", msg->data, 0, &rcvbuf, &rcvlen, 0) == -1)
        {
             printf("tpcall fail [%s]\n", tpstrerror(tperrno));
             tpfree((char *)rcvbuf);
             tpreturn(TPFAIL, 0, 0, 0, 0);
        }

        strcat(rcvbuf, "_Success");

        tpreturn(TPSUCCESS,0,(char *)rcvbuf, 0,0);
}

SVC15004_2(TPSVCINFO *msg)
{
        int     i;
        char    *rcvbuf;
        long    rcvlen;

        if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL)
            printf("rcvbuf tpalloc fail \n");
}

        if (tpcall("SVC15004_3", msg->data, 0, &rcvbuf, &rcvlen, 0) == -1)
        {
            printf("tpcall fail [%s]\n", tpstrerror(tperrno));
            tpfree((char *)rcvbuf);
            tpreturn(TPFAIL, 0, 0, 0, 0);
        }
        strcat(rcvbuf, "_Success");
        tpreturn(TPSUCCESS,0,(char *)rcvbuf, 0,0);
}

SVC15004_3(TPSVCINFO *msg)
{
        int     i;
        char    *rcvbuf;
        long    rcvlen;

        if ((rcvbuf = (char *)tpalloc("STRING", NULL, 0)) == NULL)
             printf("rcvbuf tpalloc fail \n");

        if (tpcall("SVC15004_4", msg->data, 0, &rcvbuf, &rcvlen, 0) == -1)
        {
            printf("tpcall fail [%s]\n", tpstrerror(tperrno));
            tpfree((char *)rcvbuf);
            tpreturn(TPFAIL, 0, 0, 0, 0);
         }

        strcat(rcvbuf, "_Success");
        tpreturn(TPSUCCESS,0,(char *)rcvbuf, 0,0);
}
```

Makefile

다음은 Makefile의 예제이다.

<Makefile.c.mc>

```
# Server makefile

TARGET  = $(COMP_TARGET)
APOBJS  = $(TARGET).o
NSDLOBJ = $(TMAXDIR)/lib64/sdl.o

LIBS    = -lsvrmc -lnodb
OBJS    = $(APOBJS) $(SVCTOBJ)

SVCTOBJ = $(TARGET)_svctab.o

CFLAGS  = -O -Ae -w +DSblended +DD64 -D_HP -I$(TMAXDIR) -D_MCONTEXT

APPDIR  = $(TMAXDIR)/appbin
SVCTDIR = $(TMAXDIR)/svct
LIBDIR  = $(TMAXDIR)/lib64

#
.SUFFIXES : .c

.c.o:
        $(CC) $(CFLAGS) -c $<

#
# server compile
#

$(TARGET): $(OBJS)
        $(CC) $(CFLAGS) -L$(LIBDIR) -o $(TARGET) $(OBJS) $(LIBS) $(NSDLOBJ)
        mv $(TARGET) $(APPDIR)/.
        rm -f $(OBJS)

$(APOBJS): $(TARGET).c
        $(CC) $(CFLAGS) -c $(TARGET).c

$(SVCTOBJ):
        cp -f $(SVCTDIR)/$(TARGET)_svctab.c .
        touch ./$(TARGET)_svctab.c
        $(CC) $(CFLAGS) -c ./$(TARGET)_svctab.c

#
clean:
        -rm -f *.o core $(APPDIR)/$(TARGET)
```

6. 안내서 구성

본 장은 Tmax 전체 안내서에 대한 목록과 각 안내서에 수록하고 있는 내용에 대해서 설명한다.

6.1. 개요

제품에 대한 안내서가 있어도 제품에 대해 여러 가지 안내서가 존재하다 보니 제품을 처음 접하거나 제품에 익숙하지 않은 사용자는 원하는 안내서를 찾는 데 문제를 겪을 수 있다.

예를 들어, Tmax를 사용하는 환경에서 Delphi와 같은 4GL 언어를 이용해서 프로그램을 개발하는 경우 해당 프로그램의 흐름이나 사용해야 하는 API에 대해 알고 있어야 하는 데 필요한 자료가 어떤 안내서를 참고해야 하는지 알기 어렵다. 본 장에서는 안내서 사용에 있어 사용자의 편의를 돕기 위해 Tmax 안내서에는 어떤 것들이 있고, 각 안내서에서 어떤 내용을 기술하고 있으며, 각 안내서가 다른 안내서와 어떻게 연관을 맺고 있는지를 설명한다.

Tmax 안내서를 처음 접하는 사용자는 본 장을 주의 깊게 읽어볼 것을 권장한다. 비록 본 장이 실제 제품 사용법과 직접적인 관련이 있는 것은 아니지만, 안내서 구성을 전체적으로 이해하는 데 도움이 되도록 작성되었다.

6.2. 안내서 구성과 내용

Tmax의 안내서 목록은 다음과 같다.

NO	안내명
1	Tmax Getting Started Guide
2	Tmax Installation Guide
3	Tmax Administration Guide
4	Tmax Application Development Guide
5	Tmax Error Message Reference Guide
6	Tmax Reference Guide
7	Tmax JTmaxServer Guide
8	Tmax FDL Reference Guide
9	Tmax Host-link Guide(SNA LU0, SNA LU6.2)
10	Tmax HMS User Guide
11	Tmax WebT User Guide
12	Tmax COBOL User Guide
13	Tmax Gateway Guide(SERIAL)
14	Tmax Gateway Guide(TCP/IP)
15	Tmax Gateway Guide(TCP/IP Thread)
16	Tmax Gateway Guide(TCP/IP Service)

NO	안내명
17	Tmax Gateway Guide(WebService)
18	Tmax Gateway Guide(X.25)
19	Tmax Programming Guide(4GL)
20	Tmax Programming Guide(Dynamic Library)
21	Tmax Programming Guide(MultipleRM)
22	Tmax Programming Guide(RCA)
23	Tmax Programming Guide(RPC)
24	Tmax Programming Guide(RQ)
25	Tmax Programming Guide(SQ)
26	Tmax Programming Guide(UCS)
27	TmaxGrid User Guide
28	Tmax TChache Guide
29	Tmax XA Library and XA Gateway Guide
30	Tmax WebT User Guide
31	Tmax WebtAsync User Guide
32	Tmax WebTJCA User Guide
33	Tmax JTC User Guide

Tmax는 총 33권의 안내서를 제공한다. 각 안내서에 대한 내용은 본 장에 설명되어 있으니 특정한 내용에 대해 빨리 찾기를 원한다면 다음의 내용을 먼저 확인하기 바란다.

- Tmax Getting Started Guide

  본 안내서이다. Tmax에 대한 기본 개념과 구성에 대한 기본적인 사항을 기술한다.

- Tmax Installation Guide

  Tmax를 설치하는 과정에 대해서 설명한다.

- Tmax Administration Guide

  Tmax를 이용하기 위한 환경 설정 파일과 시스템 운영방식에 대해서 기술한다.

  - 환경변수에 대한 설명과 설정하는 방법
  - 환경파일 설정 방법과 사용 환경별 절의 설정 방법
  - 시스템의 기동과 종료 방법
  - Tmax 시스템을 운영하기 위한 어드민 툴의 사용법, 정보 조회 방법, 운용 관리 방법

- Tmax Application Development Guide

  Tmax를 사용하여 프로그램을 개발하는 사용자를 위해 기술된 안내서이다. 프로그램 개발에 대한 기본적인

내용을 기술한다.

  - Tmax를 사용하는 애플리케이션의 개념, 구성, 특징
  - 클라이언트 프로그램의 흐름, 특징, 개발 환경, 컴파일 방법
  - 서버 프로그램의 흐름, 특징, 개발 환경, 컴파일 방법
  - 클라이언트/서버의 통신 유형에 대한 설명
  - 클라이언트/서버의 통신 유형에 사용되는 버퍼의 종류와 관리 방법
  - 트랜잭션에 대한 특징과 관리 방법
  - Multi Thread 및 멀티 컨텍스트 환경에서 프로그램의 흐름, 구현 방법, 예제
  - Tmax에서 제공하는 보안 시스템에 대한 설명과 각 단계별 특징
  - 클라이언트에서 사용이 가능한 API와 사용 방법
  - 서버에서 사용이 가능한 API와 사용 방법
  - 다양한 환경에 대한 애플리케이션 예제
  - Tmax에서 발생하는 에러에 대한 처리 방법과 디버그
  - Tmax 환경 설정 예제 파일와 Makefile 예제

- Tmax Error Message Reference Guide

  Tmax 제품을 사용하는 중에 발생할 수 있는 에러와 해당 에러의 대응 방법 등에 대해 기술한다.

  - Tmax 에러 메시지에 대한 구조
  - 공통 에러 메시지에 대한 설명과 대응 방법
  - 각 모듈별 발생하는 에러 메시지에 대한 설명과 대응 방법

- Tmax Reference Guide

  Tmax를 사용하여 프로그램을 개발하는 사용자를 위해 기술된 안내서로 Tmax 애플리케이션 개발에 사용하는 명령어의 개념과 사용법 및 예제에 대해서 설명한다. 클라이언트와 서버의 연결, 통신에서 사용하는 함수에 대한 사용 방법과 예제에 대해서 기술한다.

  - 명령어에 대한 개념과 사용 방법, 예제
  - 함수에 대한 설명과 사용 방법, 예제
  - 에러별 설명과 대응 방법

- Tmax JTmaxServer Guide

  Tmax에서 제공하는 JTmaxServer를 사용하는 개발자를 위해 기술된 안내서로 환경 구성방법, 환경 설정 방법, API와 예제에 대해서 기술한다.

- Tmax FDL Reference Guide

  Tmax FDL 함수 정의와 예제 프로그램을 통하여 FDL의 모든 기능을 잘 활용 할 수 있도록 방법을 기술한 안내서이다.

  - FDL의 개념, 필드 버퍼에 대한 설명과 FDL 테이블 생성

- FDL 함수의 사용법, 예제

- Tmax Host-link Guide(SNA LU0, SNA LU6.2)

  Tmax와 호스트를 연결하는 Host-link 개발자를 위한 안내서이다.

  - Host-link에 대한 개념 설명, 구조, 기능, 기동 및 종료 방법
  - INBOUND, OUTBOUND 세션 관리 방법
  - INBOUND, OUTBOUND 서비스
  - Host-link을 사용하는 중에 발생하는 문제에 대한 설명과 해결 방법
  - 환경 설정 방법, 상태 모니터링 API, 에러 코드, 사용자 함수, Host-link에서 사용하는 파일 목록

- Tmax HMS User Guide

  HMS를 이용하여 프로그램을 사용하는 개발자와 운영자를 대상으로 작성된 안내서이다.

  - HMS의 개념, 구조, 프로그래밍을 위한 개념 설명
  - 환경파일에 HMS 설정 방법, 예제
  - HMS API 함수의 사용법과 예제, HMS 빌드, 기동 및 종료, 조회 등 관리 방법
  - HMS를 사용하기 위한 환경 설정, 프로그램 예제, 컴파일 방법

- Tmax COBOL User Guide

  COBOL을 사용하여 Tmax 서버 프로그램을 작성하는 방법을 설명한다.

  - Tmax 환경파일 생성 방법, 컴파일, 서비스 테이블 생성, 엔진 기동 방법
  - 서버 프로그램 작성 방법, Makefile 예제, 서버 프로그램 기동 및 테스트 방법
  - 서버 프로그램에서 FDL 사용 방법, 전역 트랜잭션 프로그램 방법, TPSVRINIT / TPSVRDONE 사용 방법
  - COBOL에서 사용할 명령어와 함수에 대한 사용 방법 및 예제

- Tmax Gateway Guide(SERIAL)

  Tmax 서버와 Non-Tmax 서버가 Serial 통신을 하는 경우 인터페이스를 담당하는 Tmax에서 제공되는 SERIAL(RS232) Gateway에 대해 기술한 안내서이다.

  - SERIAL Gateway의 개념과 동작 구조
  - SERIAL Gateway의 서비스 유형
  - SERIAL Gateway를 사용하기 위한 환경 설정
  - SERIALGateway를 사용한 프로그램 예제

- Tmax Gateway Guide(TCP/IP)

  Tmax 서버와 Non-Tmax 서버의 인터페이스를 담당하는 Tmax에서 제공되는 TCP/IP Gateway에 대해 기술한 안내서이다.

  - TCP/IP Gateway의 개념과 동작 구조
  - TCP/IP Gateway의 서비스 유형

    - TCP/IP Gateway를 사용하기 위한 환경 설정
    - TCP/IP Gateway를 사용한 프로그램 예제
- Tmax Gateway Guide(TCP/IP Service)

  Tmax 서버와 Non-Tmax 서버의 TCP/IP 통신 시 인터페이스를 담당하는 Tmax에서 제공되는 TCP/IP Service Gateway에 대해 기술한 안내서이다.

    - TCP/IP Service Gateway의 개념과 동작 구조
    - TCP/IP Service Gateway의 서비스 유형
    - TCP/IP Service Gateway를 사용하기 위한 환경 설정
    - TCP/IP Service Gateway를 사용하기 위한 API
    - TCP/IP Service Gateway를 사용한 프로그램 예제
- Tmax Gateway Guide(TCP/IP Thread)

  Non-Tmax 클라이언트와 Tmax 사이에서 인터페이스 역할을 해주는 TCP/IP Thread Gateway에 대해 기술한 안내서이다.

    - TCP/IP Thread Gateway에 대한 개념과 프로세스
    - TCP/IP Thread Gateway의 유형
    - TCP/IP Thread Gateway를 사용하기 위한 환경 설정
    - TCP/IP Thread Gateway를 사용하기 위한 사용자 API 함수
    - TCP/IP Thread Gateway를 사용한 프로그램 예제
- Tmax Gateway Guide(WebService)

  Tmax 서비스를 특별한 변경 없이 웹서비스로 사용하기 위해서 제공되는 게이트웨이 서버인 WebService Gateway에 대해 기술한 안내서이다.

    - 웹서비스의 개념, WebService Gateway에 대한 소개
    - WebService Gateway를 사용하기 위한 환경 설정
    - WebService Gateway 사용방법 및 관리 방법
- Tmax Gateway Guide(X.25)

  Tmax 서버와 Non-Tmax 서버의 인터페이스를 담당하는 Tmax에서 제공되는 X.25 Gateway에 대해서 기술한 안내서이다.

    - X.25 Gateway에 대한 개념과 동작 구조
    - X.25 Gateway의 서비스 유형
    - X.25 Gateway를 사용하기 위한 환경 설정 방법
    - X.25 Gateway를 사용한 프로그램 예제
- Tmax Programming Guide(4GL)

  4GL 언어를 사용해서 Tmax 애플리케이션을 개발하는 사용자를 위해 기술된 안내서이다.

- Power Builder를 이용한 프로그램 개발 방법, 함수의 사용법과 예제, 실제 개발된 애플리케이션 예제
- Visual Basic를 이용한 프로그램 개발 방법, 함수의 사용법과 예제, 실제 개발된 애플리케이션 예제
- Delphi를 이용한 프로그램 개발 방법, 함수의 사용법과 예제, 실제 개발된 애플리케이션 예제
- Visual Basic .net를 이용한 프로그램 개발 방법, 함수의 사용법과 예제, 실제 개발된 애플리케이션 예제
- C# .net를 이용한 프로그램 개발 방법, 함수의 사용법과 예제, 실제 개발된 애플리케이션 예제
- ASP를 이용한 프로그램 개발 방법, 함수의 사용법과 예제, 실제 개발된 애플리케이션 예제

- Tmax Programming Guide(Dynamic Library)

  Tmax TDL(Tmax Dynamic Library)을 사용해서 프로그램을 개발하려는 사용자를 위해 기술된 안내서이다.

  - TDL의 개념, 특징, 구성에 대한 설명
  - TCDL을 사용하기 위한 환경 설정, Tmax 환경 설정
  - TDL 명령어의 사용 방법, TDL API의 사용 방법과 예제

- Tmax Programming Guide(MultipleRM)

  Tmax 서버 프로세스 중 Tmax Multiple RM에 대한 이해와 기능 및 특성에 대한 습득을 위한 안내서이다.

  - Multiple RM에 대한 기본 설명
  - 환경 설정과 서버 사용법, 프로그램 예제

- Tmax Programming Guide(RCA)

  Tmax 클라이언트 라이브러리를 사용할 수 없는 기존 통신 프로그램이나 PDA 등을 Tmax 시스템과 연결할 수 있도록 지원하는 RCA 모듈의 설치 및 환경 설정 방법을 설명하는 안내서이다.

  - RCA의 기본 개념과 구조, 특징
  - RCA와 Tmax 시스템을 연동하는 방법
  - RCA를 이용하기 위한 환경변수
  - 각 플랫폼에 제공되는 파일과 Tmax의 환경파일에 설정하는 방법
  - 개발자가 RCAH를 프로그래밍할 때의 유의사항과 예제, 에러 메시지

- Tmax Programming Guide(RPC)

  Tmax 프로그램 모델 중 RPC(Remote Procedure Call)에 대한 이해와 기능 및 특성, 구성 요소에 대해 설명하는 안내서이다.

  - RPC의 기본 개념과 종류, 제약 사항
  - RPC의 구성 요소

- Tmax Programming Guide(RQ)

  Tmax의 RQ(Reliable Queue)에 대한 개념과 사용법에 대해 기술한 안내서이다.

  - RQ에 대한 정의, 개념, 특징
  - RQ를 사용하는 시스템 구성, API

    - RQ를 사용하기 위한 환경 설정, 상태 정보 관리
- Tmax Programming Guide(SQ)

  Tmax의 SQ(Session Queue)에 대한 개념과 사용법에 대해 기술한 안내서이다.

    - SQ에 대한 정의, 개념, 특징
    - SQ를 사용하는 시스템 구성, API
    - SQ를 사용하기 위한 환경 설정, 상태 정보 관리
    - SQ를 사용하는 예제
- Tmax Programming Guide(UCS)

  서버 프로세스 중 Tmax UCS에 대한 개념 및 사용 과정에 대해 기술한 안내서이다.

    - 프로세스에 대한 기본 설명과 프로그램 흐름
    - UCS 프로그램 구성, 환경 설정, 및 컴파일 방법, 함수의 사용법과 프로그램 예제
    - RDP 프로그램 구성, 환경 설정 및 컴파일 방법과 프로그램 예제
- TmaxGrid User Guide

  TmaxGrid는 멀티 노드 환경에서 in-memory 데이터를 동기화 시키는데 목적을 두고 있는 TmaxGrid를 사용하여 프로그램을 개발하는 사용자와 관리자를 위한 안내서이다.

    - TmaxGrid에 대한 기본 설명
    - 환경 설정과 서버 사용법, 프로그램 예제
- Tmax TCache Guide

  Tmax TCache의 기본 개념과 사용법에 대해 기술한 안내서이다.

    - TCache에 대한 기본 설명
    - TCache 툴 사용법
    - TCache API 사용법
    - TCache 동기화 방법
- Tmax XA Library and XA Gateway Guide

  XA 라이브러리와 XA 게이트웨이의 사용법에 대해서 기술한 안내서이다.

- Tmax WebAdmin User Guide

  Tmax를 관리하는 사용자를 위한 툴인 WebAdmin의 기본 개념과 설치, 사용법에 대해 기술한 안내서이다.

    - WebAdmin의 기본 개념과 제약 사항
    - WebAdmin을 운영하기 위한 설치 과정 및 시작 사용 방법
    - 각 메뉴별 기능 및 사용법에 대한 설명
    - WebAdmin에서 발생하는 메시지에 대한 설명
- Tmax WebT User Guide

웹을 통해 Tmax의 서비스를 제공받기 위해 JEUS의 WebT API를 사용하여 클라이언트 프로그램을 개발하려는 프로그램 개발자를 대상으로 기술한다.

  - WebT의 기능 역할, WebTConnectionPool, WebT-Server System의 개념과 구성요소 설명
  - WebT, JMax의 환경 설정 방법
  - 각 용도별 WebT API의 사용법, 예제

- Tmax WebtAsync User Guide

  Tmax의 Async Java Gateway와 인바운드/아웃바운드 통신을 비동기로 하기 위한 Java 라이브러리인 WebTAsync에 대해 설명하는 안내서이다. WebTAsync에 대한 개념과 인바운드 및 아웃바운드에 대한 설명과 예제를 기술한다.

- Tmax WebTJCA User Guide

  Tmax의 WebTJCA를 사용하여 개발하는 개발자를 대상으로 기술한 안내서이다. WebTJCA를 해서 프로그램을 개발하기 위한 사항을 기술한다.

  - JCA의 개념, 구조, WebTJCA의 개념 설명
  - CCI, 트랜잭션,로깅, 비동기 및 대화형 통신에서 사용되는 WebTJCA API에 대한 설명
  - WebTJCA 인바운드 통신
  - WebLogic, JEUS6에서 WebTJCA를 사용하는 예제

- Tmax JTC User Guide

  Tmax의 JTC(Jeus-Tuxedo Connector)는 JEUS, Tuxedo에 접속하여 서비스를 이용할 수 있는 라이브러리로 WebT 라이브러리에 포함되어 배포된다. JTC를 사용해서 프로그램을 개발하는 경우 참고할 수 있는 안내서이다.

  - JEUS, Tuxedo에 연결하기 위한 환경 설정 방법
  - JTC 클래스와 함수의 사용법, 예제
  - Tuxedo, WebT 로그 설정 방법