

개발자 안내서

OpenFrame OSI 7.2

TMAXSOFT

저작권 공지

Copyright 2025. TmaxSoft Co., Ltd. All Rights Reserved.

회사 정보

(주)티맥스소프트

주소 : 경기도 성남시 분당구 정자일로 45, 티맥스소프트타워

기술 서비스 센터: 1544-8629

홈페이지: <https://www.tmaxsoft.com>

제한된 권리

이 소프트웨어(Tmax OpenFrame®) 사용설명서와 프로그램은 저작권법과 국제 조약에 의해 보호됩니다. 사용설명서와 프로그램은 TmaxSoft Co., Ltd.와의 사용권 계약 하에서만 사용할 수 있으며, 사용설명서는 사용권 계약의 범위 내에서만 배포 또는 복제할 수 있습니다. 이 사용설명서의 전부 또는 일부를 TmaxSoft의 사전 서면 동의 없이 전자, 기계, 녹음 등의 수단으로 전송, 복제, 배포하거나 2차적 저작물을 작성할 수 없습니다.

이 소프트웨어 사용설명서와 프로그램의 사용권 계약은 어떠한 경우에도 사용설명서 및 프로그램과 관련된 지적 재산권(등록 여부를 불문)을 양도하는 것으로 해석되지 않으며, 브랜드나 로고, 상표 등을 사용할 권한을 부여하지 않습니다. 사용설명서는 오로지 정보 제공만을 목적으로 하며, 이로 인한 계약상의 직접적 또는 간접적 책임을 지지 않습니다. 또한 사용설명서 상의 내용이 법적 또는 상업적인 특정 조건을 만족시킬 것을 보장하지 않습니다. 사용설명서는 제품의 업그레이드나 수정에 따라 예고 없이 변경될 수 있으며, 내용상의 오류가 없음을 보장하지 않습니다.

상표 공지

Tmax®와 Tmax OpenFrame®은 TmaxSoft Co., Ltd.의 등록 상표입니다. 본 사용설명서에 기재된 모든 제품과 회사 이름은 각각 해당 소유주의 상표로서 참조용으로만 사용되며 반드시 상표 표시 (™, ®)를 하지는 않습니다.

오픈소스 소프트웨어 공지

본 제품은 "OpenSSL", "RSA Data Security, Inc.", "Apache Foundation" 및 "Jean-loup Gailly와 Mark Adler"에 의해 개발 또는 라이선스된 오픈 소스 소프트웨어를 포함합니다. 관련 상세 정보는 제품의 다음 디렉터리에 기재된 사항을 참고하시기 바랍니다. : \${INSTALL_PATH}/license/oss_licenses

안내서 이력

| 제품 버전 | 안내서 버전 | 발행일 | 비고 |
|-------------------|--------|------------|----|
| OpenFrame OSI 7.2 | 3.2.1 | 2025-08-14 | |
| OpenFrame OSI 7.2 | 3.1.2 | 2023-12-29 | |
| OpenFrame OSI 7.2 | 3.1.1 | 2023-08-30 | |

목차

| | |
|------------------------------|----|
| 1. 소개 | 1 |
| 1.1. 개요 | 1 |
| 1.2. 데이터베이스 | 1 |
| 1.3. 데이터 통신 | 1 |
| 1.4. 애플리케이션 | 1 |
| 2. 애플리케이션 인터페이스 | 4 |
| 2.1. 개요 | 4 |
| 2.2. DL/I 인터페이스 | 4 |
| 2.3. AIBTDLI 인터페이스 | 4 |
| 2.4. 데이터베이스 Call 함수 | 4 |
| 2.5. 데이터 통신 Call 함수 | 6 |
| 3. 애플리케이션 작성 | 9 |
| 3.1. 애플리케이션 작성 과정 | 9 |
| 3.1.1. 애플리케이션 설계 | 9 |
| 3.1.2. 프로그래밍 | 10 |
| 3.1.3. Batch 애플리케이션 | 10 |
| 3.2. MPP 애플리케이션 | 10 |
| 3.2.1. MPP 사용자 서버 준비 | 11 |
| 3.2.2. MPP 프로그래밍 | 12 |
| 3.2.3. 메시지 세그먼트 포맷 | 13 |
| 3.3. BMP 애플리케이션 | 15 |
| Appendix A: DL/I Status code | 16 |
| Appendix B: IO-PCB Mask | 19 |
| Appendix C: AIB Mask | 20 |
| Appendix D: DL/I Call | 21 |
| D.1. 시스템 서비스 DL/I Call | 21 |
| D.1.1. CHKP (basic) | 21 |
| D.1.2. CHKP (symbolic) | 21 |
| D.1.3. INIT | 22 |
| D.1.4. INQY | 22 |
| D.1.5. LOG | 24 |
| D.1.6. ROLB | 24 |
| D.1.7. SYNC | 24 |
| D.1.8. XRST | 25 |
| D.2. 트랜잭션 관리 DL/I Call | 25 |
| D.2.1. CHNG | 25 |
| D.2.2. CMD | 26 |
| D.2.3. GCMD | 26 |
| D.2.4. GN | 26 |

| | |
|-------------------|----|
| D.2.5. GU..... | 27 |
| D.2.6. ISRT..... | 27 |
| D.2.7. PURG | 28 |

1. 소개

본 장에서는 OpenFrame OSI 애플리케이션 프로그래밍에 대한 개념과 특징적인 구조에 대해 기술한다.

1.1. 개요

개발자는 OpenFrame OSI(이하 OSI) 시스템에서 운영되는 새로운 애플리케이션을 개발하거나, 기존 IMS/DC에서 운영하던 애플리케이션들을 마이그레이션하여 트랜잭션 서비스를 구성할 수 있다. OSI 시스템에서 제공하는 많은 리소스들은 IMS/DC가 제공하는 기능 및 서비스를 동일하게 제공한다.

OSI 프로그래밍은 **DL/I** 함수를 기본으로 해서 각각의 함수들의 기능들을 조합하여 업무 프로그램을 구성한다. DL/I 함수의 종류와 기능은 [애플리케이션 인터페이스](#)에서 설명한다.

1.2. 데이터베이스

IMS/DC 시스템이 사용하는 데이터베이스는 계층적 데이터베이스 관리 시스템으로 고객의 업무 데이터를 구조적으로 저장하고 관리하는 기능을 제공한다. 이 데이터베이스를 IMS/DB라고 부르는데 IMS/DC에서 실행되는 애플리케이션이나 순수 Batch 애플리케이션이 이 데이터베이스(IMS/DB)에 접근하기 위해서는 표준 DL/I 함수를 호출해야 한다.

각 프로그램 언어에서 제공되는 표준 애플리케이션 인터페이스인 **DL/I**를 통해 애플리케이션은 표준 DL/I 함수를 사용할 수 있다.

OSI에서는 데이터베이스 구조를 UNIX에서 재구현하며 IMS/DC와 모든 기능을 동일하게 제공하도록 OpenFrame HiDB를 연동한다.

1.3. 데이터 통신

사용자가 단말을 통해 애플리케이션이 처리한 데이터의 결과값을 볼 수 있도록 IMS/DC는 시스템 서버(Control Region)과 사용자 서버(Dependent Region)을 제공한다.

OSI는 Tmax와의 연동을 통해 IMS/DC의 기능을 OSI에서도 동일하게 제공하고 효율적으로 트랜잭션을 처리한다. OSI는 IMS/DC에서 운영 중인 고객의 프로그램을 OSI에서 동일하게 운영할 수 있도록 IMS/DC에서 제공하는 DL/I 인터페이스와 동일한 인터페이스를 제공한다.

DL/I 인터페이스를 제공하기 때문에 사용자 서버(Dependent Region)의 MPP, BMP에서 운영되는 사용자 프로그램을 간단한 마이그레이션만으로 OSI에서 운영할 수 있다.

1.4. 애플리케이션

OSI는 IMS/DC에서 동작하던 애플리케이션을 동일하게 동작할 수 있도록 IMS/DC에서 제공하던 기능 및 기반 구조를 제공한다.

IMS/DC 애플리케이션은 Non IMS/DC 애플리케이션과는 다르게 반드시 PSB(Program Specification Block)를 하나씩 가지고 있어야 한다. PSB는 애플리케이션에 IMS에서 제공하는 서비스를 실행하기 위한 인터페이스 역할을 한다. 이러한 서비스에는 메시지를 단말에 보내는 방식, 데이터베이스에 접근하는 방식, IMS 명령어, IMS 서비스 Call이 있다. OSI 애플리케이션은 IMS 서비스 처리방식을 PSB를 통해서 결정할 수 있다.

PSB와 애플리케이션은 OSI 시스템에 의해서 동시에 로딩되며, 애플리케이션 내부에서 실행되는 Call 함수를 통해 OSI의 실행 모듈들은 애플리케이션이 요청하는 기능을 수행한다. 각각의 MPP 애플리케이션들은 MPP 타입 사용자 서버를 통해 OSI의 기능들(DL/I Call 함수, 명령어 등)을 실행할 수 있으며, 이를 위해서 필요한 시스템 정의들은 미리 등록되어 있어야 한다. 시스템 정의는 IMS/DC에서 사용하였던 시스템 매크로를 통해 할 수 있다.

Batch 애플리케이션은 위의 경우와는 다른 방식으로 실행된다. Batch 애플리케이션이 OSI 시스템에서 실행되기 위해서는 BMP(Batch Message Processing Region) 서버가 필요하다. 이 서버는 Batch JOB(JCL 기동)을 통해서 요청되는 애플리케이션의 함수들을 처리할 수 있다.

애플리케이션을 작성할 때 ENTRY 문에는 프로그램에서 사용하는 PCB를 정의하는데, 정의된 PCB를 통해 OSI 시스템의 기능을 이용할 수 있다. PCB는 애플리케이션에서 사용되는 OpenFrame HiDB에 대한 VIEW나 메시지 소스 또는 메시지 대상과 통신하기 위해 시스템에서 제공하는 컨트롤 블록이다. OSI에서 PCB는 400개까지 사용할 수 있다.

다음은 애플리케이션의 PCB에 대한 설명이다.

- Database Program Control Block (DB-PCB)

애플리케이션이 데이터베이스에 저장되어 있는 정보를 어떤 방식으로 읽어올 것인가를 기술한다. 똑같은 데이터베이스라도 DB-PCB에 기술되어 있는 방식에 따라서 다르게 읽혀진다.

- IO Program Control Block (IO-PCB)

OSI는 기본적으로 단말을 사용하여 동작하고 이러한 Online 방식으로 기동되는 애플리케이션은 IO-PCB를 필요로 한다.

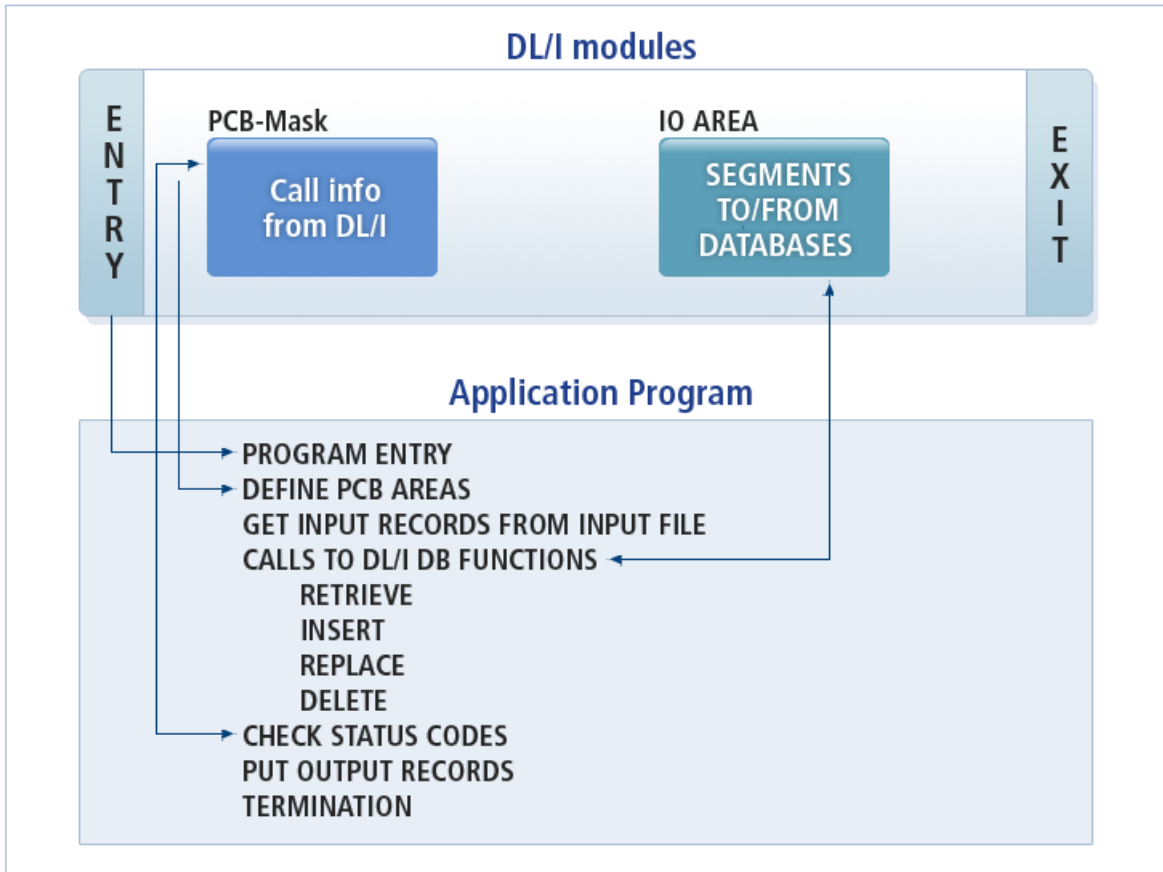
- Alternative Program Control Block (ALT-PCB)

애플리케이션 안에서 다른 프로그램 혹은 다른 터미널로 메시지를 전달하고자 할 때 ALT-PCB를 사용한다. ALT-PCB는 목적지를 변경하여 사용이 가능하다.



PSB와 DBD의 자세한 문법에 대한 자세한 내용은 OpenFrame HiDB "HiDB안내서"를 참고한다.

다음은 앞에서 설명한 애플리케이션의 일반적인 동작 구조를 그림으로 나타낸 것이다.



애플리케이션 동작 구조

2. 애플리케이션 인터페이스

본 장에서는 OpenFrame OSI 환경에서 작성하는 COBOL 프로그램이 사용하는 인터페이스의 종류와 사용법에 대해서 설명한다.

2.1. 개요

OSI 환경에서 애플리케이션은 데이터베이스나 MQ에 접근을 하기 위해서 반드시 정해진 인터페이스(**DL/I Call 함수**)를 지켜야 한다. 각 함수는 데이터베이스뿐만 아니라 OSI 영역(Online 프로그램)에서도 사용되며 사용 방식은 동일하다. SSA(Segment Search Argument)와 같은 특정한 옵션은 Online에서는 사용되지 않는다. 데이터베이스를 Online에서는 MQ(Message Queue)라고 생각하고 함수의 기능을 대입하면 된다.

인터페이스를 지키지 않았을 경우에는 사용자가 인식할 수 있는 Status code를 반환값으로 돌려준다. Status code에 대한 자세한 내용은 [DL/I Status code](#)를 참고한다.



DL/I Call 함수에 대한 자세한 사항은 OpenFrame HiDB "HiDB 안내서"를 참고한다.

2.2. DL/I 인터페이스

애플리케이션이 DL/I Call 함수를 사용하기 위해서는 DL/I 인터페이스를 사용해야 한다. 애플리케이션 언어에 따라 CBLTDLI, PLITDLI, ASMTDLI 등 다양한 DL/I 인터페이스를 사용할 수 있다.

다음은 DL/I 인터페이스의 사용법이다. argument 1에 DL/I Call 함수 이름을, argument 2에는 PCB를 지정한다.

```
CALL 'CBLTDLI' USING argument 1, argument 2, ..., argument n
```

2.3. AIBTDLI 인터페이스

AIB(Application Interface Block)는 PCB 주소를 알 수 없을 때 대신 사용하거나, DL/I Call 함수가 PCB를 필요로 하지 않는 경우에 사용할 수 있다. DL/I Call 함수에서 PCB가 필요한 경우에는, AIB의 리소스 이름과 psbgen 시 등록된 PCB 이름이 동일해야 한다.

다음은 애플리케이션이 AIBTDLI 인터페이스를 통해 DL/I Call 함수를 사용하는 방법이다. argument 1에는 DL/I Call 함수를, argument 2에는 PCB 대신 AIB를 지정한다.

```
CALL 'AIBTDLI' argument 1, argument 2, ..., argument n
```

2.4. 데이터베이스 Call 함수

다음은 애플리케이션이 데이터베이스에 접근하기 위하여 사용하는 함수의 사용법이다.

- 사용법

Function Code(argument 1) PCB(argument 2) 데이터 영역(argument 3) SSA(argument 4)

- argument

- argument 1

DL/I 함수를 지칭한다. 함수 코드는 데이터베이스 세그먼트의 처리의 종류를 결정하는 코드이며, 이 필드는 4Byte 길이의 영역으로서 애플리케이션 내에서 선언한다.

다음은 argument 1에 지정할 수 있는 함수이다.

| 함수 | 설명 |
|---|--|
| GU(GET UNIQUE) / GHU(GET HOLD UNIQUE) | 데이터베이스의 특정의 세그먼트를 검색하는 함수 코드이며, SSA(argument 4번째부터 n번째까지 지정)를 지정하여 세그먼트를 얻을 수 있다. Online 프로그램에서는 MQ에 적재되어 있는 메시지를 읽어오도록 지시한다. |
| GN(GET NEXT) / GHN(GET HOLD NEXT) | 현재 포지셔닝되고 있는 세그먼트(Call 발행 직전에 검색된 세그먼트)의 다음 세그먼트를 검색하는 함수 코드로, 데이터베이스의 세그먼트를 계층 순서에 검색하는 경우에 사용한다. Online 프로그램에서는 MQ가 데이터베이스 역할을 하며, 함수의 기능 또한 데이터베이스가 아닌 MQ를 타겟으로 한다. |
| GNP(GET NEXT WITHIN PARENT) / GHNP(GET HOLD NEXT WITHIN PARENT) | 데이터 구조의 특정 세그먼트 타입을 부모로 하고 그 세그먼트의 하위 레벨에 있는 세그먼트를 포함할 수 있다. 이것을 패밀리 세그먼트라고 하는데 GNP, GHNP 함수 코드는 이 패밀리 세그먼트의 계층 순서에 검색한다. Online 프로그램(IO-PCB)를 사용하는 경우에는 사용하지 않는 함수이다. |
| ISRT(INSERT) | argument 3에 입력하는 데이터를 지정한 위치의 데이터베이스 혹은 MQ에 Insert 한다. |
| DLET(DELETE) | 세그먼트를 삭제하는 경우에 사용하는 함수 코드이다. 세그먼트를 삭제하는 경우 홀드계의 검색 Call을 사용해 세그먼트의 포지셔닝을 한 후에 DLET Call을 사용한다. 삭제되는 세그먼트는 데이터베이스로부터 삭제 규칙에 따라 삭제된다. |
| REPL(REPLACE) | 세그먼트의 치환을 수행하는 경우에 사용하는 함수 코드이다. 세그먼트를 치환하는 경우도 DLET Call과 동일하게 홀드계의 검색 Call을 사용해 포지셔닝을 한 후에 REPL Call을 사용한다. REPL Call에 의해 세그먼트의 키를 변경하면 안된다. Online 프로그램(IO-PCB)를 사용하는 경우에는 사용하지 않는 함수이다. |

- argument 2

애플리케이션이 데이터 액세스와 인터페이스를 사용하기 위해서 PCB Mask를 DB-PCB, IO-PCB, ALT-PCB 중 하나로 정의한다. PCB는 애플리케이션이 COBOL 언어인 경우 LINKAGE 절에 정의한다. 애플리케이션에 정의된 PCB가 Call 기능을 실행하는 경우에는 OSI 내의 CONTROL 절과 링크된다.

| PCB Mask | 설명 |
|----------|--|
| DB-PCB | 데이터베이스 Call 기능에 대해 정의된 PCB를 DB-PCB 라고 한다. |
| IO-PCB | Online 기능을 사용할 경우에 사용한다. |
| ALT-PCB | 다른 트랜잭션 혹은 터미널로 보내고 싶을 경우에 사용한다. |

◦ argument 3

애플리케이션 내의 데이터 영역을 지정한다. 애플리케이션과 데이터 액세스의 데이터의 처리는 모두 데이터 영역에 지정해서 사용한다. 사용자는 세그먼트의 길이와 SSA의 Command Code에 의해 필요한 길이를 애플리케이션 내에 XX 영역에 선언해야 한다.

◦ argument 4

SSA의 처리의 대상이 되는 세그먼트와 Sensitive 세그먼트에 대한 조건을 규정하는 영역이다. SSA의 선두 명칭을 데이터 구조로 정의한 계층 레벨에 따라서 지정한다. n의 값은 최대 18까지이다. IO-PCB Mask를 사용할 경우에는 MOD 명칭을 설정하여 단말에 표시되는 화면을 변경할 수 있다.

SSA는 데이터베이스 Call 기능의 특유 argument이며, 세그먼트에게 전달하는 조건의 유무에 의해 unqualified SSA와 qualified SSA로 분류된다.

| SSA | 설명 |
|-----------------|--|
| unqualified SSA | 처리하는 세그먼트의 명칭만으로 SSA 처리를 하고 세그먼트 내의 필드 내용에 조건을 붙이지 않는 SSA이다. |
| qualified SSA | 세그먼트 내의 필드 내용에 조건을 붙이는 SSA이다. |



데이터베이스 Call 함수의 자세한 정보는 "IBM IMS Application Programming:Database Manual"을 참고한다.

2.5. 데이터 통신 Call 함수

다음은 데이터 통신 Call 함수의 기능이다.

- 메시지 세그먼트의 송수신
- 메시지의 송신 목표의 변경
- SPA(Scratch Pad Area)의 송신

데이터 통신 Call 함수는 데이터베이스 Call 함수와 기능면에서는 동일하다. 하지만 Online에서 MQ에 적재되어 있는 메시지와 SPA 영역의 메시지는 계층 구조(OpenFrame HiDB가 가지는 특징적인 구조)를 가지고 있지 않기 때문에 계층적 검색 조건인 데이터베이스 Call의 SSA는 필요하지 않다.



SPA(Scratch Pad Area)에 대한 자세한 정보는 "[IBM IMS Conversational Transactions](#)"를 참고한다.

• 사용법

Function Code(argument 1) PCB(argument 2) 데이터 영역(argument 3) MOD 명칭(argument 4)

• argument

◦ argument 1

함수 코드를 지정하는 argument로, 함수 코드는 메시지 세그먼트, SPA 처리의 종별을 지정하는 코드이다. 데이터베이스 Call 기능과 동일하게 4Byte의 영역으로 애플리케이션에서 선언한다.

다음은 argument 1에 지정할 수 있는 함수이다.

| 함수 | 설명 |
|----------------|--|
| GU(GET UNIQUE) | 메시지의 최초의 세그먼트(헤더 세그먼트)를 수신한다. 애플리케이션은 메시지의 수신에 있어서 GU를 최초로 호출해야 한다. 대화 모드(Conversation Mode)로 처리되는 경우는 SPA가 수신된다. |
| GN(GET NEXT) | 헤더 세그먼트에 이어지는 메시지 세그먼트를 수신한다. |
| ISRT(INSERT) | 메시지 세그먼트 혹은 SPA를 송신한다. |
| PURG(PURGE) | ISRT에 의해 송신된 메시지 세그먼트를 1개의 메시지로 단말에 출력한다. |
| CHNG(CHANGE) | ALT-PCB를 사용하고, 다른 터미널 혹은 트랜잭션에 메시지를 보내고 싶을 때 사용한다. |

◦ argument 2

데이터 통신 Call 함수를 위한 PCB는 데이터베이스 PCB와 같은 방법으로 정의하고 **논리 단말 PCB**라고 한다.

다음은 argument 2에 지정할 수 있는 논리 단말 PCB의 종류이다.

| PCB | 설명 |
|-------------------------|--------------------------------------|
| IO-PCB | 트랜잭션을 요청한 논리 단말의 메시지 입/출력을 위해서 사용한다. |
| Alternative PCB | 기본 단말 이외의 다른 단말로 메시지 송신하기 위해서 사용한다. |
| Alternative Program PCB | 메시지를 다른 프로그램으로 송신하기 위해서 사용한다. |



1. Alternative Program PCB는 ALT-PCB라고 한다. 애플리케이션 내에서 이러한 논리 단말 PCB는 IO-PCB, ALT-PCB의 순서로 정의해야 한다.

2. IO-PCB는 OSI 시스템에서 자동으로 관리되고 ALT-PCB는 프로그램 정의 유틸리티로 정의된다.

- argument 3

메시지 세그먼트, Command, SPA의 송수신을 위해 사용되는 데이터 영역이다.

- argument 4

MFS를 수행하는 단말에 대해 송신하는 메시지의 포맷을 미리 메시지 포맷 정의 유틸리티로 정의한 MOD 명칭(8Byte)이 설정된 영역의 주소를 지정한다. MOD 명칭이 지정되는 것은 ISRT하는 경우 뿐이다.



데이터 통신 Call 함수의 자세한 정보는 "IBM IMS Application Programming: Transaction Manager Manual"을 참고한다.

3. 애플리케이션 작성

본 장에서는 애플리케이션 프로그램을 작성하는 방법에 대해서 기술한다.

3.1. 애플리케이션 작성 과정

애플리케이션의 작성에는 애플리케이션의 설계, 프로그래밍, 디버그 및 실행의 단계로 나눌 수 있다.

각각의 단계에는 OSI에 필수로 정의되어야 하는 리소스부터 사용자가 임의로 작성하는 프로그래밍까지 포함한다.

3.1.1. 애플리케이션 설계

애플리케이션의 설계는 다음의 과정으로 진행된다.

1. 개발자는 애플리케이션이 어떠한 타입의 사용자 서버에서 구동될 것인지를 결정해야 한다.

애플리케이션의 처리를 담당하는 사용자 서버에는 MPP 사용자 서버와 BMP 사용자 서버가 있는데, 애플리케이션이 위 2개의 방식 중 어떤 방식으로 실행될 것인가를 결정해야 한다.

2. 서버의 타입이 정해지면 애플리케이션과 사용자 서버와의 관계를 정해야 한다.

애플리케이션과 사용자 서버와의 관계를 정하는 방법은 시스템 리소스 정의(System Resource Definition)에서 정의할 수 있다.

다음은 시스템 정의에 대한 예제이다.

```

APPLCTN PSB=OIVPI001,PGMTYPE=(,,),SCHDTYP=PARALLEL
TRANSACT CODE=OIVPMPP1,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5),      X
      MODE=SNGL
APPLCTN PSB=OIVPI002,PGMTYPE=(TP,,1),SCHDTYP=PARALLEL
TRANSACT CODE=OIVPMPP2,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5),      X
      MODE=SNGL,MAXRGN=1

APPLCTN PSB=OIVPIL05,PGMTYPE=BATCH,SCHDTYP=PARALLEL
TRANSACT CODE=OIVPBMP5,MSGTYPE=(SNGLSEG,RESPONSE,1),PRTY=(1,5),      X
      MODE=SNGL

```

3. 애플리케이션을 설계한다. 애플리케이션은 유지보수, 확장성을 고려한 프로그램 구조, 모듈 설계가 이루어져야 한다. 애플리케이션을 설계할 때 다음 사항을 유의해야 한다.

- DL/I 함수 호출을 효과적으로 사용해야 한다.
- 입/출력 처리 횟수를 최소화해야 한다.
- 사용 데이터베이스의 데이터 구조와 처리 옵션(검색, 추가, 삭제)에 대한 고려를 해야 한다.
- 처리 모드(대화 모드 처리, 일반 트랜잭션의 모드 등)의 설정에 대해 고려해야 한다.
- 메시지 경로에 대해 고려해야 한다.

3.1.2. 프로그래밍

OSI 애플리케이션은 COBOL로 작성하며 다음 사항에 유의하면서 작성한다.

- 프로그램의 시작 조건의 설정에 유의한다.
- DB-PCB, IO-PCB, ALT-PCB의 설정에 유의한다.
- 함수 호출 인터페이스의 설정에 유의한다.
- Status code의 체크와 에러 처리의 설정에 유의한다.
- 데이터 처리할 때의 정합성에 유의한다. (예: 데이터 길이)
- 프로그램의 종료 조건의 설정에 유의한다. (예: 반환코드)
- 프로그램 정의에서 입/출력 데이터 크기나 SSA 크기에 관계없이 프로그래밍할 때 데이터 전송 영역은 충분한 사이즈를 확보해야 한다.

3.1.3. Batch 애플리케이션

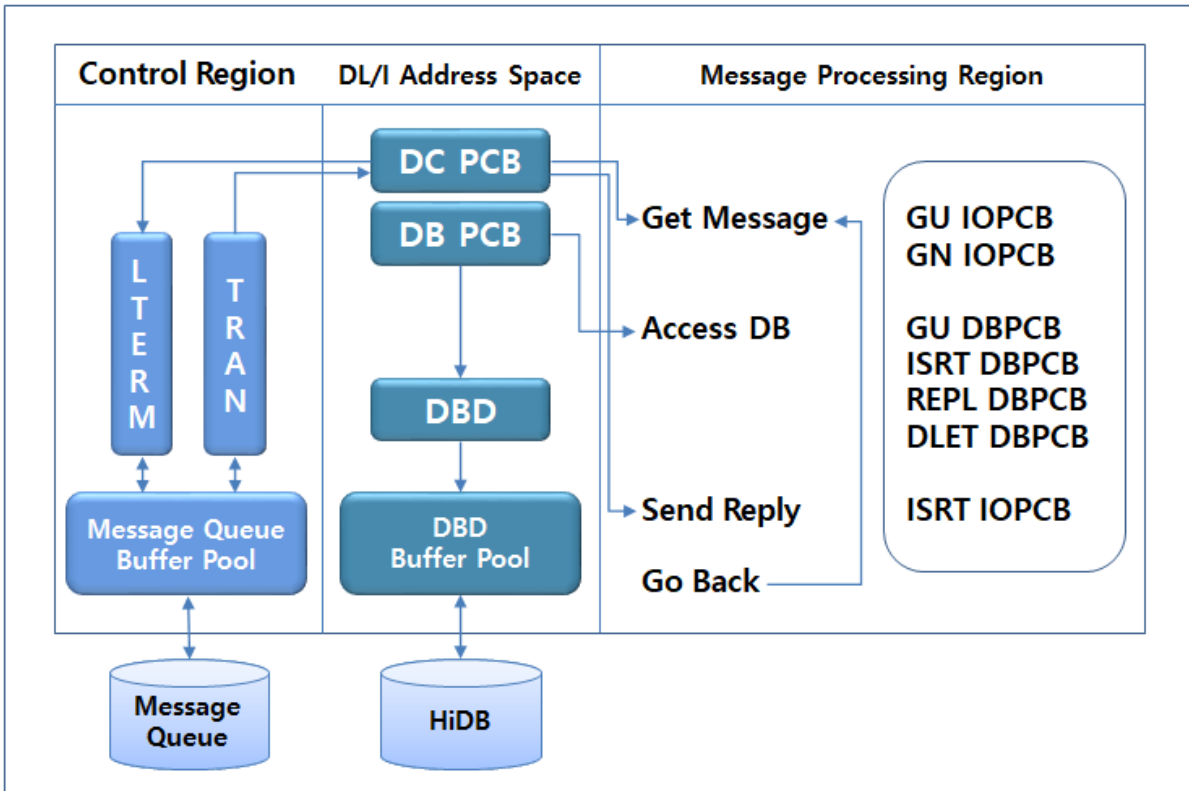
Batch 애플리케이션은 OpenFrame의 유틸리티인 **tjesmgr**에 의해서 실행된다. Batch 애플리케이션의 작성 방법은 기본적으로 MPP 애플리케이션과 동일하다. 단지 JCL을 통해서 실행된다는 점이 다르다. tjesmgr은 JCL을 효과적으로 실행할 수 있는 유틸리티이다.

Batch 애플리케이션은 Online 업무를 위한 프로그램이 아니므로 단말과의 통신을 위한 데이터 영역을 정의할 필요는 없다. 프로그램이 데이터베이스에 연결할 수 있는 로직과 MQ(Message Queue)에서 메시지를 로드할 수 있는 로직 또는 MQ에 데이터를 다시 쌓을 수 있는 로직만 있으면 된다.

3.2. MPP 애플리케이션

IMS/DC Online 환경에서 메시지를 처리하는 프로그램을 MPP(Message Processing Program) 또는 MPR(Message Processing Region)이라고 한다. OSI에서는 이들을 각각 시스템 서버와 사용자 서버로 분리하여 통칭한다.

다음은 Online 환경에서 OSI 시스템과 애플리케이션의 구성을 나타내는 그림이다.



OSI 시스템과 애플리케이션의 구성

일반적으로 단말(LTERM)로부터 입력된 트랜잭션은 시스템 서버(Control Region)에 의해 분석되고 이 트랜잭션을 처리하는 MPP 애플리케이션은 OSI의 사용자 서버 중 하나인 MPP 사용자 서버에 로드되어 실행된다. 이 트랜잭션이 OSI 시스템에 전달하는 데이터 영역은 트랜잭션 코드(최대 8 문자)와 데이터 영역으로 구성되는데, 시스템 서버는 전달받은 트랜잭션 코드를 이용하여 RTS(RunTime System Definition) 테이블로부터 클래스 정보를 취득하고, 해당 클래스를 처리하는 MPP 사용자 서버로 스케줄링하게 된다.

MPP 애플리케이션은 Get Unique(GU) Call로 단말로부터 입력된 메시지를 얻을 수 있다. 여러 개의 사용자 서버를 기동하여 다수의 MPP 애플리케이션 처리를 할 수 있으며, 데이터베이스를 동시에 접근하는 일도 가능하다. LTERM 뿐만 아니라 트랜잭션 간에도 ALT-PCB를 사용해서 MPP 애플리케이션을 시작할 수 있다.

3.2.1. MPP 사용자 서버 준비

MPP 사용자 서버를 기동하는 데 필요한 사항은 Tmax 환경 파일 설정, MPP 기동 JCL 파일 준비가 있다. OSI에는 OSIMPPSVR 바이너리 파일 하나로 Tmax의 target 옵션을 사용하여 서로다른 클래스 정보를 처리하는 MPP 서버를 기동시킬 수 있다.

- Tmax 환경 파일 설정

OSI는 Tmax 환경에서 동작하기 때문에 사용자 서버도 Tmax 서버에 등록을 해야 정상적으로 동작한다. 서버 등록 방법은 Tmax에서 서버를 등록하는 일반적인 방법과 동일하다. Tmax 서버를 등록할 때 관련된 서비스도 같이 등록해야 한다.

```
#####
#  OpenFrame OSI Dependent Region Servers                               #
#####
OSIMPPSVR      SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO
#####
```

```

#   OSI, example in which IMSID is IMSA                                     #
#####
IMSASCHD      SVGNAME = svg_node1, MAX = 1, SVRTYPE = UCS, RESTART = NO,
              TARGET = osisschd
IMSACMMD      SVGNAME = svg_node1, MAX = 1, SVRTYPE = UCS, RESTART = NO,
              TARGET = osicmdsv
IMSAMPP_TCL1  SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO,
              TARGET = OSIMPPSVR
IMSAMPP_TCL2  SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO,
              TARGET = OSIMPPSVR
IMSAMPP_TCL3  SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO,
              TARGET = OSIMPPSVR
IMSAMPP_TCL4  SVGNAME = svg_node1, MIN = 1, MAX = 10, RESTART = NO,
              TARGET = OSIMPPSVR

*SERVICE
#####
#   OSI USER APPLICATION SERVER DEFAULT                                   #
#####
OSIMPPSVRSVC  SVRNAME = OSIMPPSVR, SVCTIME=60

```

• MPP 서버 기동 JCL

다음은 시스템 서버(Control Region) 이름이 IMSA인 경우 관리자가 생성한 사용자 서버(Dependent Region)에 해당하는 MPP 사용자 서버를 기동하는 JCL 파일이다. 클래스를 지정하지 않을 시 '000' 으로 설정 가능하며 첫 번째 클래스는 필수로 지정해야 한다. (본 예제에서 처리할 클래스는 1, 2, 3, 4이다)

```

//IMSMSG JOB
//STEP1 EXEC PGM=DFSRRC00,
//          PARM='MSG,001002003004,W00099000,,,R,,,IMSA,,,,D2PA'
//STEPLIB DD DISP=SHR,DSN=OSI.IMSA.STEPLIB
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSDBOUT DD SYSOUT=*
//

```

3.2.2. MPP 프로그래밍

MPP 프로그래밍의 기본은 GU와 ISRT Call에 의해 이루어진다.

메시지를 GU 처리하는 경우는 IO-PCB를 사용해서 DL/I 함수를 호출한다. 메시지가 1개 이상의 세그먼트로 구성되어 있는 경우에는 2번째 이후 세그먼트는 GN Call을 통해서 메시지를 읽어들이어야 한다(SPA를 사용하는 트랜잭션의 경우에는 GU로 SPA를 읽어들이고 그 이후에 GN로 메시지를 읽는다).

GU와 GN Call의 결과는 IO-PCB의 Status code 필드에 설정된다. MPP 프로그램은 Call 처리 후 반드시 Status code를 체크해야 한다. 메시지 세그먼트를 단말로 보내거나 다른 트랜잭션으로 보내는 경우에는 데이터 영역에 메시지 세그먼트를 정해진 포맷으로 설정해서 ISRT Call로 DL/I 함수를 호출한다.

다른 MPP 프로그램 또는 다른 단말에 메시지를 보낼 수 있는데 ALT-PCB를 사용해서 Destination을 지정해야 한다.

주의사항

다음은 MPP 프로그래밍의 주의사항이다.

- 트랜잭션이 입력될 때마다 스케줄되어 MPP가 동적으로 빈 사용자 서버에 로딩된다. MPP의 처리를 완료한 후, 사용하고 있던 사용자 서버는 다른 MPP 프로그램에 할당되거나 IDLE 상태로 되기 때문에 MPP 사용자 서버 내에 정보를 저장할 수 없다. 데이터의 저장이 필요한 경우에는 SPA를 사용한다.
- MPP의 실행을 위해서 OS가 해야 하는 사전 처리가 많기 때문에 한번 MPP가 로드되면 다수의 메시지를 처리하는 프로그램을 작성한다. 하지만 메시지의 처리를 위해서 너무 많은 시간을 소비하면 시스템을 정의할 때 지정한 우선순위에 따라서 균일화된 트랜잭션 스케줄링을 못할 수 있다. 이 경우는 시스템 정의에서 TRANSACT 매크로의 PROCLIM 지정을 활용한다.
- MPP로 데이터베이스를 갱신하는 경우에는 데이터베이스에 대한 처리를 수행하기 때문에 주의가 필요하다. 애플리케이션을 필요 이상으로 너무 크게 해서 데이터베이스의 액세스를 독점하는 경우가 없도록 주의해야 한다.
- 균등한 스케줄링 서비스를 수행하기 위해서 1개의 트랜잭션으로 많은 시간을 필요로 하는 MPP를 작성하지 말고 여러 개의 MPP에 분할하도록 프로그래밍한다.
- MPP는 처리 중에 Status code를 검사해서 문제가 있는 경우 더 이상의 처리를 불가능하게 만든다. 데이터베이스를 처리하는 중 데이터베이스의 백 아웃이 필요한 경우에는 반드시 이상종료(ABEND)시켜야 한다.
- OpenFrame 환경설정 중 hidb 서브섹트, HIDB_DEFAULT 섹션의 USE_DBD_DBMS_LOCK 키의 값이 YES로 설정되어 있을 경우, 트랜잭션이 시작될 때 트랜잭션이 사용하는 DBD 명으로 Lock이 생성되고, 트랜잭션이 종료될 때 Lock이 해제된다. 자세한 사항은 OpenFrame "환경설정 안내서"를 참조한다.

3.2.3. 메시지 세그먼트 포맷

다음은 수신 메시지 세그먼트, 송신 메시지 세그먼트, 프로그램 간 메시지 세그먼트에 대한 설명이다.

3.2.3.1. 수신 메시지 세그먼트

사용자 프로그램(애플리케이션)에서는 단말 혹은 다른 트랜잭션으로부터 메시지를 전달받는데 DL/I 함수의 GU, GN Call을 사용하여 COBOL에 정의한 데이터 영역으로 읽어들이는 데이터를 수신한다.

일반적으로 애플리케이션이 수신한 메시지 세그먼트의 포맷은 다음과 같다.

```
LL ZZ DATA
```

| 필드 | 설명 |
|----|--|
| LL | LL 및 ZZ 필드의 길이를 포함한 메시지 세그먼트의 길이로 2bytes의 2진수이다. PL/I로 기술된 애플리케이션의 경우는 4byte이며, (실제의 길이-2)의 길이로 설정된다. (LL = DATA 길이+4) |
| ZZ | 2bytes로 데이터 액세스를 위한 영역이다. 이 필드는 개발자가 설정할 필요없는 필드로 임의로 수정할 경우 에러가 발생한다. |

| 필드 | 설명 |
|------|--|
| DATA | <p>단말로부터 입력된 메시지 세그먼트이다.</p> <p>메시지 세그먼트는 트랜잭션 코드와 데이터 영역으로 구성된다. 단말로 보내진 데이터에는 앞 8bytes가 트랜잭션 이름이 되며, 그 이후부터가 실제 데이터가 된다.</p> <p>복수의 메시지 세그먼트로 입력되는 메시지의 경우, 최초의 메시지 세그먼트의 선두 8bytes 이내에 반드시 트랜잭션 코드를 포함한다.</p> |

3.2.3.2. 송신 메시지 세그먼트

애플리케이션에서 단말, 다른 트랜잭션, 다른 단말로 메시지를 보내는 경우에는 DL/I 함수의 ISRT Call을 사용하여 IO-PCB 혹은 ALT-PCB에 설정된 Destination으로 메시지를 보낸다.

송신 메시지 세그먼트의 포맷은 다음과 같다.

```
LL ZZ DATA
```

| 필드 | 설명 |
|------|---|
| LL | <p>LL 및 ZZ 필드의 길이를 포함한 메시지 세그먼트의 길이로 2bytes의 2진수이다.</p> <p>PL/I로 기술된 애플리케이션의 경우는 4bytes이다. 애플리케이션에서 반드시 이 필드를 설정해야 한다. (LL = DATA 길이 + 4)</p> |
| Z1 | 1Byte로 데이터 액세스를 위한 영역이다. 애플리케이션에서는 이 필드에 2진수로 0을 입력해야 한다. |
| Z2 | 1Byte로 논리 페이징을 할 때 새로운 페이지의 시작을 MFS에게 알려야 하는 경우 X'40'을 입력한다. 이 외의 경우에는 2진수로 0을 입력해야 한다. 자세한 내용은 OpenFrame OSI "MFS 참조 안내서"를 참고한다. |
| DATA | 특정 Destination(현재 단말, 다른 트랜잭션, 다른 단말)로 보내는 데이터 영역이다. 메시지 세그먼트의 길이는 Destination에 따라 다르다. 메시지는 여러 메시지 세그먼트라도 상관없다. |

3.2.3.3. 프로그램 간 메시지 세그먼트

특정 프로그램으로부터 다른 프로그램으로 메시지를 보낼 수 있다. 프로그램 간 메시지를 전달하는 경우 앞에서 설명한 송신 메시지 세그먼트와 방법이 동일하다.

다음은 프로그램 간 전달하는 메시지 세그먼트의 포맷이다.

```
LL ZZ DATA
```

| 필드 | 설명 |
|--------|--------------------------|
| LL, ZZ | 송신 메시지 세그먼트의 포맷과 같다. |
| DATA | 상대 프로그램에 메시지를 보내는 데이터이다. |

주의사항

다음은 프로그램 간 전달하는 메시지 세그먼트를 사용할 때 주의해야 하는 사항이다. 애플리케이션에서 사용하는 ALT-PCB 는 PSB에 미리 정의되어 있어야 하며, 애플리케이션에서 사용할 PCB들(LINKAGE SECTION 및 ENTRY 구문)은 PSB에 정의된 PCB들의 순서대로 정의되어 있어야 한다.

- 애플리케이션 AP1으로부터 다른 애플리케이션 AP2에게 메시지를 보내는 경우 ALT-PCB를 사용하며 ALT-PCB의 Destination을 보내려는 AP2의 이름으로 설정한 다음 CHNG, ISRT, PURG 순서로 DL/I 함수를 사용한다.
- AP2에서 메시지를 수신할 때 IO-PCB를 지정하여 해당 메시지를 수신할 수 있다. IO-PCB에는 AP1의 ALT-PCB의 내용이 설정되어야 정상적으로 데이터(데이터 구조 적합성)를 받을 수 있다.
- AP2에서 수신 메시지를 처리한 후, 다시 메시지를 AP1에 송신하는 경우에는 AP1의 ALT-PCB를 지정한다. 이 경우 AP1이 AP2로부터 메시지를 받아들여 처리를 하고, 메시지를 단말에 출력하는 경우에는 IO-PCB를 지정해 메시지를 송신한다. AP2가 메시지를 수신할 수 있는 것은 MPP나 BMP의 경우이다.

3.3. BMP 애플리케이션

MPP와 달리 사용자는 데이터베이스 액세스 기능 외에 데이터 관리를 사용하고, OSI에서 사용자 데이터셋을 액세스할 수 있다. 데이터베이스나 메시지를 처리하는 프로그램 방식은 MPP와 같다. MPP 프로그래밍과 차이점은 MQ에 적재되어 있는 데이터를 실시간으로 처리하는 것이 아니고 사용자가 원하는 때에 JCL을 통해 BMP 애플리케이션을 기동하여 처리한다는 점이다.

예를 들어 MPP 애플리케이션을 통해 처리된 결과물을 ALT-PCB(Alternative Program PCB)를 사용해 BMP 애플리케이션으로 보내면 OSI는 MQ 테이블에 그 결과를 적재하고 대기하다가 사용자가 JCL로 해당 BMP 애플리케이션을 실행시킬 때 MQ 테이블에 적재되어있는 모든 메시지를 Destination인 해당 BMP 애플리케이션으로 보낸다.

다음은 BMP 프로그램을 기동시키는 JCL의 예제이다.

<OSIBMP.jcl>

```
//OSIBMPT JOB CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//TSTEP1 EXEC PGM=DFSRR00,
// PARM=(BMP,OIVPIL04,,OIVPBMP4,,,,,,,,,IMSA,)
//SYSOUT DD SYSOUT=*
```

다음은 앞의 JCL을 실행하는 tjesmgr 유틸리티를 실행하는 예제이다.

```
$ tjesmgr run OSIBMP.jcl
> Command : [run OSIBMP.jcl]
Node name : NODE1
/home/oframe/OpenFrame/volume_default/SYS1.JCLLIB/OSIBMP.jcl is submitted as OSIBMPT(JOB00001).
$ tjesmgr ps
> Command : [ps]
JOBNAME JOBID CLASS STATUS RC NODE JCL
-----
OSIBMPT JOB00001 A Done R00000 NODE1 OSIBMP.jcl
```

Appendix A: DL/I Status code

본 부록에서는 DL/I Status code의 내용과 대응 방법에 대해 기술한다.

DL/I Call 처리에 대한 결과는 PCB Status code 필드의 코드값으로 확인할 수 있다. 애플리케이션 프로그램 작성자는 Status code 필드를 참조해 처리 결과에 대한 원인 및 대응 방법을 체크할 수 있다.

- AB

| | |
|--------------|---|
| 원인 | I/O Area가 필요한 함수 호출에서 I/O Area가 파라미터로 지정되지 않았다. |
| 대응 방법 | 함수 호출에 맞는 올바른 데이터 영역을 지정한다. |

- AD

| | |
|--------------|---|
| 원인 | 해당 함수 호출에 알맞지 않는 PCB를 파라미터로 지정하였다. (1) 시스템 서비스 Call에서 IO-PCB를 사용하지 않았다. (2) 메시지 GU Call 또는 GN Call에서 IO-PCB가 아닌 ALT-PCB를 이용하였다. (3) BMP JCL에서 IN=trancode 정의없이 메시지 GU Call을 IO-PCB에 사용하였다. |
| 대응 방법 | 해당 함수 호출에 알맞는 PCB인지 확인한다. |

- AZ

| | |
|--------------|---|
| 원인 | (1) PURG Call을 SPA 송신에 사용하였다. (2) ISRT Call을 ALT-PCB의 송신지가 회화형 트랜잭션인데 처음의 메시지 세그먼트를 SPA로 하지 않았다. |
| 대응 방법 | PURG, ISRT Call을 올바르게 사용한다. |

- A2

| | |
|--------------|--|
| 원인 | CHNG Call을 잘못 사용하였다. (1) ALT-PCB가 아닌 경우에 CHNG Call을 사용하였다. (2) ALT-PCB이나 modifiable이 아닌 PCB에 대해 CHNG Call을 사용하였다. (3) 메시지 세그먼트의 송신이 완료되지 않은 PCB에 대해 CHNG Call을 사용하였다. |
| 대응 방법 | (1) CHNG Call은 modifiable ALT-PCB를 사용한다. (2) ISRT Call로 메시지 세그먼트를 송신한 후에 송신지를 변경하고 싶은 경우에는 메시지 Purge 후 CHNG Call로 송신지를 변경한다. |

- A3

| | |
|--------------|--|
| 원인 | 송신지 지정이 없는 modifiable ALT-PCB에 대해 ISRT 또는 PURG Call을 사용하였다. PURG Call은 오직 하나의 I/O Area에 대한 파라미터로 처리한다. |
| 대응 방법 | CHNG Call로 지정한 송신지에서 사용하는 PCB에 대해 ISRT 또는 PURG Call을 사용한다. |

• A4

| | |
|--------------|---|
| 원인 | CHNG Call로 지정한 송신지에 대한 접근 권한이 없다. |
| 대응 방법 | CHNG Call로 지정한 송신지가 올바른지 확인하고, 송신지에 대한 권한 여부를 점검한다. |

• A5

| | |
|--------------|--|
| 원인 | 메시지 ISRT Call에서 잘못된 파라미터 리스트를 사용하였다. 출력 메시지의 첫 번째 세그먼트가 아닌데 4번째 파라미터(MOD 이름)를 지정하였다. |
| 대응 방법 | 메시지의 첫 번째 세그먼트가 아닌 경우에는 4번째 파라미터를 지정하지 않는다. |

• A6

| | |
|--------------|--|
| 원인 | 메시지 ISRT Call 또는 PURG Call로 송신한 메시지 세그먼트의 크기가 TRANSACT 매크로의 SEGSIZE 정의보다 크다. |
| 대응 방법 | 송신 메시지 세그먼트의 크기는 TRANSACT 매크로의 SEGSIZE 이하로 한다. |

• QC

| | |
|--------------|--|
| 원인 | 입력 큐에 해당 프로그램에 대한 메시지가 존재하지 않는 경우 GU Call을 사용하였다. |
| 대응 방법 | 해당 코드는 애플리케이션 프로그램이 종료하면서 상태 정보를 알리기 위한 목적으로 출력된다. |

• QD

| | |
|--------------|---|
| 원인 | GN Call을 사용하였는데 해당 메시지에 더이상의 세그먼트가 존재하지 않는다. |
| 대응 방법 | 세그먼트가 더 이상 없으므로 해당 메시지에 대해 애플리케이션 프로그램에서 적절하게 처리한다. |

• QE

| | |
|--------------|--|
| 원인 | 메시지 GU Call을 사용하지 않고 메시지 GN Call을 사용하였다. |
| 대응 방법 | 메시지 GN Call을 사용하기 위해서는 메시지 GU Call을 먼저 사용한다. |

• QH

| | |
|-----------|--|
| 원인 | 출력 논리 단말의 이름 또는 트랜잭션 코드가 IMS에 등록되지 않았다. 메시지 ISRT Call 또는 PURG Call로 송신한 메시지 세그먼트 길이가 5Byte 미만이다. |
|-----------|--|

| | |
|--------------|---------------------------|
| 대응 방법 | 논리 단말의 이름과 트랜잭션 코드를 확인한다. |
|--------------|---------------------------|

- XA

| | |
|--------------|--|
| 원인 | originating 터미널에 응답 후에 다른 프로그램으로 SPA를 송신하였다. |
| 대응 방법 | 애플리케이션 프로그램을 수정한다. |

- XB

| | |
|--------------|--|
| 원인 | 다른 프로그램으로 SPA를 송신한 후 originating 터미널로 출력 메시지를 송신하였다. |
| 대응 방법 | 애플리케이션 프로그램을 수정한다. |

- X2

| | |
|--------------|---|
| 원인 | ALT-PCB의 송신지가 회화용 트랜잭션인데 처음 ISRT Call을 SPA로 송신하지 않았다. |
| 대응 방법 | SPA를 먼저 ISRT한 후 메시지를 ISRT한다. |

- X3

| | |
|--------------|---|
| 원인 | SPA의 선두 6Byte 필드가 변경되어 SPA 메시지로 정의되지 않는다. |
| 대응 방법 | 처음 가져온 SPA 메시지의 선두 6Byte는 변경하지 않는다. |

- X4

| | |
|--------------|-----------------------------------|
| 원인 | 회화 처리가 정의되지 않은 트랜잭션으로 SPA를 송신하였다. |
| 대응 방법 | SPA가 아닌 데이터 메시지만 ISRT Call한다. |

- X5

| | |
|--------------|---------------------------------|
| 원인 | ISRT Call로 SPA 송신을 2번 이상 시도하였다. |
| 대응 방법 | 하나의 메시지에 하나의 SPA를 송신한다. |

- X6

| | |
|--------------|--------------------------------------|
| 원인 | ISRT Call로 SPA 송신하는데 잘못된 트랜잭션 코드이다. |
| 대응 방법 | IMS 시스템에 등록된 회화형 트랜잭션 코드로 송신지를 지정한다. |

Appendix B: IO-PCB Mask

다음은 IBM Mainframe의 IMS/DC에서 제공하고 있는 IO-PCB MASK의 필드 항목이다. 필드의 길이와 OSI 제품의 지원 여부에 대한 내용을 포함하였다.

| Descriptor | 길이(Byte) | 지원 여부 |
|--------------------------------|--------------------------------|-------|
| Logical terminal name | 8 | 지원 |
| Reserved for IMS | 2 | 지원 |
| Status code | 2 | 지원 |
| Local date and time | 4(Date)+4(Time) | 지원 |
| Input message sequence number | 4 | 미지원 |
| Message output descriptor name | 8 | 지원 |
| Userid | 8 | 지원 |
| Group name | 8 | 미지원 |
| Time Stamp | 4(Date)+6(Time6)+2(UTC Offset) | 미지원 |
| Userid Indicator | 1 | 미지원 |
| Reserved for IMS | 3 | 지원 |

Appendix C: AIB Mask

다음은 IBM Mainframe의 IMS/DC에서 제공하고 있는 AIB MASK의 필드 항목이다. 필드의 길이와 OSI 제품의 지원 여부에 대한 내용을 포함하였다.

| Descriptor | 길이(Byte) | 지원 여부 |
|----------------------------|----------|-------|
| AIB identifier | 8 | 지원 |
| DFSAIB allocated length | 2 | 미지원 |
| Subfunction code | 8 | 지원 |
| Resource name 1 | 8 | 지원 |
| Resource name 2 | 8 | 미지원 |
| Reserved | 8 | 지원 |
| Maximum output area length | 4 | 지원 |
| Output area length used | 4 | 지원 |
| Resource field | 4 | 미지원 |
| Optional area length | 4 | 미지원 |
| Reserved | 4 | 미지원 |
| Return code | 4 | 지원 |
| Reason code | 4 | 지원 |
| Error code extension | 4 | 미지원 |
| Resource address 1 | 4 | 지원 |
| Resource address 2 | 4 | 미지원 |
| Resource address 3 | 4 | 미지원 |
| User defined token | 16 | 미지원 |
| Return token | 8 | 미지원 |
| Reserved | 16 | 미지원 |

Appendix D: DL/I Call

애플리케이션 프로그램에서 시스템 서비스 함수를 사용하기 위해서는 DL/I Call을 사용한다.

본 장에서는 OSI가 지원하는 시스템 서비스 Call에 대해 설명한다.



DL/I Call의 자세한 설명은 "IMS V7 Application Programming: Transaction Manager"을 참고한다.

D.1. 시스템 서비스 DL/I Call

D.1.1. CHKP (basic)

Basic CHKP Call은 애플리케이션 프로그램이 변경 사항을 반영하거나 또는 애플리케이션 프로그램이 이상 종료했을 경우 복구 시점을 지정하기 위해 사용한다.

- 포맷

```
>>-CHKP---i/o pcb---i/o area-----<<
```

- 파라미터

| 파라미터 | 설명 |
|----------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| i/o area | 입력 매개변수로 8byte 크기의 checkpoint ID를 지정한다. |

D.1.2. CHKP (symbolic)

Symbolic CHKP Call은 애플리케이션 프로그램이 변경 사항을 반영하려고 할 때 또는 애플리케이션 프로그램이 이상 종료했을 경우 복구 시점을 지정하기 위해 사용한다. 이상 종료한 프로그램 재시작하기 위해 Extended Restart(XRST) Call을 사용할 수 있다. 7개까지 데이터를 저장할 수 있으며 프로그램이 재시작될 때 복구된다.

- 포맷

```
>>-CHKP---i/o pcb---i/o area length--i/o area--+-----+--<<
                                     | .----- . |
                                     | V           | |
                                     '---area length--area+-'
```

- 파라미터

| 파라미터 | 설명 |
|-------------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| i/o area | 입력 매개변수로 8byte 크기의 checkpoint ID를 지정한다. |
| area length | 입력 매개변수로 area의 길이를 4byte크기로 지정한다. |
| area | 입력 매개변수로 area length만큼 데이터를 저장한다. |

D.1.3. INIT

IBM 메인프레임의 IMS/DC에서 INIT Call은 애플리케이션 프로그램은 각 DB PCB의 데이터 가용성을 확인하여 데이터 가용성 상태 코드를 확인할 수 있다.

OSI에서는 INIT Call은 애플리케이션 프로그램의 INIT Call 호출 여부 확인 플래그를 설정하는 용도로만 사용한다.

- 포맷

```
>>-INIT---+i/o pcb---+---i/o area-----<<
      '---aib-----'
```

- 파라미터

| 파라미터 | 설명 |
|----------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| aib | 입출력 매개변수로 전달할 aib를 지정한다. |
| i/o area | 입출력 매개변수로 세그먼트 송신에 충분한 크기의 I/O Area를 지정한다. |

D.1.4. INQY

INQY Call은 실행 환경, 목적지 타입 및 상태, 그리고 세션 상태 등을 포함하는 정보를 요청하기 위해 사용한다. OSI에서는 AIBSFUNC으로 ENVIRON과 FIND를 사용할 수 있다.

- 포맷

```
>>-INQY---+aib---+---i/o area-----<<
```

- 파라미터

| 파라미터 | 설명 |
|----------|--|
| aib | 입출력 매개변수로 전달할 aib를 지정한다. |
| i/o area | 입출력 매개변수로 세그먼트 송신에 충분한 크기의 I/O Area를 지정한다. |

D.1.4.1. ENVIRON

ENVIRON Subfunction은 애플리케이션의 실행 환경 정보를 얻는다.

- Output

| 타입 | 길이 | 값 | 설명 |
|---------------------------|----|----------|-------------------------|
| IMS ID | 8 | | IMS ID를 나타낸다. |
| Release level | 4 | | 현재 지원하지 않는다. |
| CTL Reg Type | 8 | DB/DC | 컨트롤 리전의 타입을 나타낸다. |
| App Reg Type | 8 | MPP, BMP | 실행중인 애플리케이션의 타입을 나타낸다. |
| Reg ID | 4 | | 현재 지원하지 않는다. |
| App Pgm Name | 8 | | 실행 중인 애플리케이션의 이름을 나타낸다. |
| PSB Name | 8 | | 실행 중인 PSB 이름을 나타낸다. |
| Tran Name | 8 | | 실행 중인 트랜잭션의 이름을 나타낸다. |
| User ID | 8 | | 현재 지원하지 않는다. |
| Group Name | 8 | | 현재 지원하지 않는다. |
| Stat Grp Indicator | 4 | | 현재 지원하지 않는다. |
| Addr of Recovery Token | 4 | | 현재 지원하지 않는다. |
| Addr of the App Param Str | 4 | | 현재 지원하지 않는다. |
| Shared Queues Indicator | 4 | | 현재 지원하지 않는다. |
| User ID of Address Space | 8 | | 현재 지원하지 않는다. |
| User ID Indicator | 1 | | 현재 지원하지 않는다. |
| Res. Recov. Serv. Ind. | 3 | | 현재 지원하지 않는다. |
| catalog enablement indi. | 8 | | 현재 지원하지 않는다. |

D.1.4.2. FIND

IBM 메인프레임의 IMS/DC에서 FIND Subfunction은 요청한 PCB 이름에 대한 PCB 주소를 반환한다.

OSI의 FIND Subfunction은 애플리케이션의 실행 환경 정보를 얻는다.

D.1.5. LOG

IBM 메인프레임의 IMS/DC에서 LOG call은 IMS system log에 정보를 저장하기 위해 사용한다.

OSI에서 LOG Call은 정보를 OFM_OSI_LOG TABLE에 저장한다. 현재 OSI에서는 AIB만 지원한다.

- 포맷

```
>>-LOG---+aib---+-----<<
```

- 파라미터

| 파라미터 | 설명 |
|------|--------------------------|
| aib | 입출력 매개변수로 전달할 aib를 지정한다. |

D.1.6. ROLB

ROLB Call은 애플리케이션 프로그램에서 출력 메시지 및 데이터베이스 변경 사항을 취소하기 위해 사용한다.

- 포맷

```
>>-ROLB---+i/o pcb---+-----<<
```

- 파라미터

| 파라미터 | 설명 |
|---------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |

- 제한사항

- OSC 메시지는 ROLB Call을 사용할 수 없다.

D.1.7. SYNC

SYNC Call은 애플리케이션 프로그램이 commit 처리를 하기 위해 사용된다. BMP 프로그램에서만 적용된다.

- 포맷

```
>>-SYNC---+i/o pcb---+-----<<  
      '-aib-----'
```

- 파라미터

| 파라미터 | 설명 |
|---------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| aib | 입출력 매개변수로 전달할 aib를 지정한다. |

D.1.8. XRST

XRST Call은 이전 애플리케이션 프로그램 동작의 symbolic checkpoint로 부터 재시작하기 위해 사용된다.

- 포맷

```
>>-XRST---i/o pcb---i/o area length--i/o area-----+--<
                                     | .-----+-->
                                     | V         |
                                     |---area length--area-+-|
```

- 파라미터

| 파라미터 | 설명 |
|-----------------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| i/o area length | 입출력 매개변수로 현재 사용되지 않는다. |
| i/o area | 입력 매개변수로 checkpoint ID를 지정한다. |
| area length | 입력 매개변수로 area의 길이를 4byte크기로 지정한다. |
| area | 입출력 매개변수로 area length만큼 데이터를 복구한다. |

D.2. 트랜잭션 관리 DL/I Call

D.2.1. CHNG

CHGN (Change) Call은 메시지 세그먼트 송신지를 변경하기 위하여 사용한다. CHNG Call로 modifiable ALT-PCB의 destination을 특정 Logical terminal, LU 6.2 descriptor, 또는 트랜잭션 코드로 지정할 수 있다.

- 포맷

```
>>-CHNG---alternate pcb+--destination name-----<
```

- 파라미터

| 파라미터 | 설명 |
|---------------|---|
| alternate pcb | CHNG Call을 사용하기 위해서는 입출력 매개변수로 modifiable ALT-PCB를 지정해야 한다. |

| 파라미터 | 설명 |
|------------------|-------------------------|
| destination name | 변경할 메시지 세그먼트 송신지를 지정한다. |

D.2.2. CMD

CMD Call은 애플리케이션 프로그램이 IMS 명령어를 입력하기 위해 사용한다. GCMD Call과 함께 사용하여 OSI 커맨드 서버로 명령어를 보내고 응답을 받을 수 있다. 명령어 응답 메시지의 첫번째 세그먼트를 가져온다.

- 포맷

```
>>-CMD---i/o pcb---i/o area-----<<
```

- 파라미터

| 파라미터 | 설명 |
|----------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| i/o area | 입출력 매개변수로 세그먼트 송신에 충분한 크기의 I/O Area를 지정한다. |

D.2.3. GCMD

GCMD Call은 CMD Call을 통해 처리한 명령어 응답 메시지의 다음 세그먼트를 가져온다.

- 포맷

```
>>-GCMD---i/o pcb---i/o area-----<<
```

- 파라미터

| 파라미터 | 설명 |
|----------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| i/o area | 출력 매개변수로 세그먼트 송신에 충분한 크기의 I/O Area를 지정한다. |

D.2.4. GN

입력된 메시지가 하나 이상의 세그먼트를 포함하고 있을 경우 GU Call을 이용하여 메시지의 첫 번째 세그먼트를 얻은 후 다음 세그먼트는 GN Call을 이용하여 가져온다.

- 포맷

```
>>-GN---i/o pcb---i/o area-----<<
```

- 파라미터

| 파라미터 | 설명 |
|----------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| i/o area | 출력 매개변수로 세그먼트 송신에 충분한 크기의 I/O Area를 지정한다. |

D.2.5. GU

GU Call은 메시지의 첫 번째 세그먼트를 가져온다.

- 포맷

```
>>-GU---i/o pcb---i/o area-----<<
```

- 파라미터

| 파라미터 | 설명 |
|----------|--|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| i/o area | 출력 매개변수로 세그먼트 송신에 충분한 크기의 I/O Area를 지정한다. |

D.2.6. ISRT

ISRT Call은 특정 송신지로 하나의 메시지 세그먼트를 송신하기 위하여 사용한다. 송신지는 IO-PCB, alternate PCB의 지정에 따른다.

- 포맷

```
>>-ISRT---i/o pcb-----i/o area---+-----+-----<<
      '-alternate pcb-'          '-mod name-'
```

- 파라미터

| 파라미터 | 설명 |
|---------------|---|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| alternate pcb | 입출력 매개변수로 PCB를 지정한다. |
| i/o area | 출력 매개변수로 세그먼트 송신에 충분한 크기의 I/O Area를 지정한다. |
| mod name | 출력 메시지 송신을 위한 입력변수로 MOD 이름을 8Bytes로 지정한다. 지정한 MOD에 따라 출력 메시지가 포맷된다. |

D.2.7. PURG

PURG Call은 ISRT Call로 송신한 메시지 세그먼트를 출력하는 기능을 가진다. 애플리케이션 프로그램에서 하나 이상의 메시지 세그먼트를 송신하는 경우 이전의 메시지를 Purge하거나 메시지 송신을 완료 처리하기 위해 사용한다.

- 포맷

```
>>-PURG--+i/o pcb-----+--+-----+-----<<
      '-alternate pcb-' '-i/o area--+-----+-'
                                '-mod name-'
```

- 파라미터

| 파라미터 | 설명 |
|---------------|---|
| i/o pcb | 입출력 매개변수로 해당 프로그램의 첫 번째 PCB로 전달할 IO-PCB를 지정한다. |
| alternate pcb | 입출력 매개변수로 PCB를 지정한다. |
| i/o area | 출력 매개변수로 세그먼트 송신에 충분한 크기의 I/O Area를 지정한다. |
| mod name | 출력 메시지 송신을 위한 입력변수로 MOD 이름을 8Bytes로 지정한다. 지정한 MOD에 따라 출력 메시지가 포맷된다. PURG Call은 출력 메시지의 첫 번째 세그먼트를 위한 MOD 이름을 지정한다. |