

**iGate Introduction**



**Copyright Notice**

Copyright © 2009 Inzent Co., Ltd. All Rights Reserved.

대한민국 서울특별시 영등포구 대림3동 779-10 콤텍빌딩 2층

**Restricted Rights Legend**

All Inzent Software (iGate) and documents are protected by copyright laws and the Protection Act of Computer Programs, and international convention. Inzent software and documents are made available under the terms of the Inzent License Agreement and may only be used or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of Inzent Co., Ltd.

이 소프트웨어(iGate) 사용설명서의 내용과 프로그램은 저작권법, 컴퓨터프로그램보호법 및 국제 조약에 의해서 보호받고 있습니다. 사용설명서의 내용과 여기에 설명된 프로그램은 Inzent Co., Ltd.와의 사용권 계약 하에서만 사용이 가능하며, 사용권 계약을 준수하는 경우에만 사용 또는 복제할 수 있습니다. 이 사용설명서의 전부 또는 일부분을 Inzent의 사전 서면 동의 없이 전자, 기계, 녹음 등의 수단을 사용하여 전송, 복제, 배포, 2차적 저작물작성 등의 행위를 하여서는 안 됩니다.

**Trademarks**

iGate are trademark of Inzent Co., Ltd. Other products, titles or services may be registered trademarks of their respective companies.

iGate는 Inzent Co., Ltd.의 상표입니다. 기타 모든 제품들과 회사 이름은 각각 해당 소유주의 상표로서 참조용으로만 사용됩니다.

**Open Source Software Notice**

This product includes various open source software that has been developed and/or licensed by “Spring Framework”, “Hibernate”, “Terracotta, Inc”, “Apache Foundation”, “Bela Ban”, “QOS.ch”, ”XStream”. Inzent fully respects the aforementioned parties and the open source software used in this product. More details can be found within the directory “${INSTALL\_PATH}/license/oss\_licenses”

본 제품은 “Spring Framework”, “Hibernate”, “Terracotta, Inc”, “Apache Foundation”, “Bela Ban”, “QOS.ch”, ”XStream” 에 의해 개발 또는 라이선스된 오픈 소스 소프트웨어를 포함합니다. 오픈 소스 소프트웨어와 개발자에 대해 경의를 표합니다. 관련 상세 정보는 제품의 디렉터리 “${INSTALL\_PATH}/license/oss\_licenses”에 기재된 사항을 참고해 주십시오.

**사용설명서 정보**

제목: iGate Introduction

버전: v1.0.0

**목 차**

[1 iGate 소개 12](#_Toc340603401)

[1.1 개요 12](#_Toc340603402)

[1.1.1 iGate란 12](#_Toc340603403)

[1.1.2 iGate의 기대효과 12](#_Toc340603404)

[1.2 시스템 구성도 14](#_Toc340603405)

[2 실행 환경 15](#_Toc340603406)

[2.1 iGate Architecture 15](#_Toc340603407)

[2.2 구성 요소 15](#_Toc340603408)

[2.2.1 Meta Engine 15](#_Toc340603409)

[2.2.2 Connectivity Engine 16](#_Toc340603410)

[2.2.3 Message Engine 17](#_Toc340603411)

[2.2.4 ProcessRule Engine 18](#_Toc340603412)

[2.3 거래 유형 18](#_Toc340603413)

[2.3.1 로컬 서비스 거래 18](#_Toc340603414)

[2.3.2 일반 서비스 거래 19](#_Toc340603415)

[2.3.3 복합 서비스 거래 20](#_Toc340603416)

[2.3.4 대외 거래 21](#_Toc340603417)

[2.4 iGate 주요 기능 22](#_Toc340603418)

[2.4.1 Message Process 22](#_Toc340603419)

[2.4.2 거래 제한 25](#_Toc340603420)

[2.4.3 표준 메시지 25](#_Toc340603421)

[2.4.4 거래추적로그 26](#_Toc340603422)

[3 관리 환경 27](#_Toc340603423)

[3.1 iManager Architecture 27](#_Toc340603424)

[3.2 Meta 데이터 관리 27](#_Toc340603425)

[3.3 연계 시스템 정보 관리 28](#_Toc340603426)

[3.3.1 System 설정 28](#_Toc340603427)

[3.3.2 System Node 설정 29](#_Toc340603428)

[3.3.3 Adapter 설정 29](#_Toc340603429)

[3.3.4 Connector 설정 30](#_Toc340603430)

[3.4 정책 설정 관리 31](#_Toc340603431)

[3.4.1 인터페이스정책 31](#_Toc340603432)

[3.4.2 서비스 정책 32](#_Toc340603433)

[3.4.3 헤더 맵핑 정책 33](#_Toc340603434)

[3.5 사용자 및 권한 관리 34](#_Toc340603435)

[3.5.1 권한 설정 34](#_Toc340603436)

[3.5.2 Role 설정 34](#_Toc340603437)

[3.5.3 사용자설정 34](#_Toc340603438)

[3.6 시스템 모니터링 35](#_Toc340603439)

[3.7 거래 통계 35](#_Toc340603440)

[4 개발 환경 37](#_Toc340603441)

[4.1 iTools Architecture 37](#_Toc340603442)

[4.2 Designer 37](#_Toc340603443)

[4.2.1 Service Designer 37](#_Toc340603444)

[4.2.2 Interface Designer 38](#_Toc340603445)

[4.2.3 Data-Model Designer 39](#_Toc340603446)

[4.2.4 Mapping-Rule Designer 40](#_Toc340603447)

[4.2.5 Operation Designer 41](#_Toc340603448)

[4.3 Tester 44](#_Toc340603449)

[4.3.1 TestCase 44](#_Toc340603450)

[4.3.2 TestSuit 45](#_Toc340603451)

[4.4 통합 상황판 46](#_Toc340603452)

그림 목차

[**[그림 1‑１]** 시스템 구성도 14](#_Toc340603345)

[**[그림 2‑１]** iGate Runtime Architecture 15](#_Toc340603346)

[**[그림 2‑２] 로컬 서비스 흐름도** 19](#_Toc340603347)

[**[그림 2‑３]** 일반 서비스 흐름도 20](#_Toc340603348)

[**[그림 2‑４]** 복합서비스 흐름도 20](#_Toc340603349)

[[그림 2‑５] SAF거래처리 흐름도 21](#_Toc340603350)

[**[그림 2‑６] EDI방식의 파일송수신거래** 22](#_Toc340603351)

[**[그림 2‑７] Message Process** 22](#_Toc340603352)

[**[그림 2‑８] Service 등록** 23](#_Toc340603353)

[[그림 2‑９] Interface 등록 23](#_Toc340603354)

[**[그림 2‑１０] 입출력 데이터 Parsing Composing** 24](#_Toc340603355)

[[그림 2‑１１] Mapping-Rule 처리 24](#_Toc340603356)

[[그림 2‑１２] 거래제한기능 25](#_Toc340603357)

[[그림 2‑１３] 에러발생에 따른 표준코드 및 메시지처리 흐름도 26](#_Toc340603358)

[**[그림 2‑１４] 거래추적로그 조회화면** 26](#_Toc340603359)

[**[그림 3‑１]** iManager Architecture 27](#_Toc340603360)

[[그림 3‑２] Meta정보 실시간적용 흐름도 28](#_Toc340603361)

[[그림 3‑３] 연계 설정 흐름도 28](#_Toc340603362)

[[그림 3‑４] 정책 설정 개념도 31](#_Toc340603363)

[**[그림 3‑５] 인터페이스 일반정책 화면** 32](#_Toc340603364)

[**[그림 3‑６]** 인터페이스 개별정책 화면 32](#_Toc340603365)

[**[그림 3‑７]** 서비스 일반정책 화면 33](#_Toc340603366)

[**[그림 3‑８] 서비스 개별정책 화면** 33](#_Toc340603367)

[**[그림 3‑９]** 헤더맵핑정책 등록화면 34](#_Toc340603368)

[**[그림 3‑１０]** 사용자권한에 따른 메뉴화면 35](#_Toc340603369)

[**[그림 3‑１１]** 시스템모니터링 및 서버모니터링 35](#_Toc340603370)

[**[그림 3‑１２]** 거래 통계 흐름도 36](#_Toc340603371)

[**[그림 3‑１３]** 거래통계 조회화면 36](#_Toc340603372)

[**[그림 4‑１]** iTools Architecture 37](#_Toc340603373)

[**[그림 4‑２]** Service Designer 38](#_Toc340603374)

[**[그림 4‑３]** Interface Designer 39](#_Toc340603375)

[**[그림 4‑４]** Data-Model Designer 40](#_Toc340603376)

[**[그림 4‑５]** Mapping-Rule Designer 41](#_Toc340603377)

[**[그림 4‑６]** Operation Designer 43](#_Toc340603378)

[**[그림 4‑７]** Operation Component 43](#_Toc340603379)

[**[그림 4‑８]** Test Case 45](#_Toc340603380)

[**[그림 4‑９]** Test Suite 46](#_Toc340603381)

[**[그림 4‑１０]** 통합상황판 47](#_Toc340603382)

**서문**

**사용설명서의 대상**

본 사용설명서는 iGate를 사용하는 관리자, 개발자들을 대상으로 기술한다.

**사용설명서의 구성**

본 사용설명서는 총 4개의 장으로 구성되어 있다.

* 1장 iGate 소개

iGate에 대한 기본적인 이해와 iGate의 개념과 아키텍처가 기술되어 있다.

* 2장 실행 환경

iGate의 실행 환경 아키텍처와 그 기능들이 기술되어 있다.

* 3장 관리 환경  
  iGate의 관리 환경 아키텍처와 그 기능들이 기술되어 있다.
* 4장 개발 환경  
  iGate의 개발 환경 아키텍처와 그 기능들이 기술되어 있다.

**사용설명서의 표현 규칙**

|  |  |
| --- | --- |
| **표현** | **의미** |
| **굵은 텍스트** | 사용자 명령어 및 User Interface menu |
| *기울임 텍스트* | 직접 입력하는 text |
| *|* | 선택 인자 구분 |

# iGate 소개

본 장에서는 iGate에 대한 기본적인 이해와 iGate의 개념과 아키텍처를 기술한다.

## 개요

다양한 채널과 서비스를 통합하는 인프라 구축을 통해 업무시스템이 채널로부터 독립적일 수 있게 보장하며, 일관성 있는 정보 제공으로 IT 비즈니스의 효율성을 극대화한다.

### iGate란

iGate 4.0 는 SOA(Service-Oriented Architecture)기반의 ESB 엔진을 근간으로 하고 있으며, 대/내외의 채널별 공통 업무와 채널 고유 업무를 효율적으로 처리하는 중요 영역과 Application 의 인터페이스 표준화를 지원함으로써 Middle Layer Integration 의 실현을 위해 다음과 같은 사상을 기본으로 하는 솔루션이다.

* 서비스 통합
  + 기존의 각 어플리케이션들은 멀티채널 통합 아키텍처 상에서 통합되어야 한다.
  + 코어의 어플리케이션들은 SOA 상에서 데이터 로직을 제공하는 서비스 프로바이더로 존재하고, 비즈니스 프로세스는 멀티채널 통합 layer에 구현된다. 이러한 비즈니스 프로세스는 각 채널에서 재사용, 공유하여야 한다.
* 재사용성 극대화
  + 프레젠테이션 로직과 비즈니스 로직을 분리하여 구현되어야 한다.
  + 특정 기능을 수행하는 비즈니스 로직은 전사적으로 하나만 존재하며, 프레젠테이션 로직은 채널 및 사용자 특성별로 다양하게 구성할 수 있어야 한다.
* 메시지 및 인터페이스 표준화
  + 전사적인 표준 메시지를 통한 Integration Tier의 일관성 있는 비즈니스를 지원한다
  + ESB를 기반으로 인터페이스를 표준화한다.
* 고가용성 확보
  + iGate를 통해 제공되는 모든 서비스는 24x365 운영을 고려하여 설계 및 구현한다.
  + 대용량 세션 및 거래를 처리할 수 있는 안정성이 검증된 아키텍처로 구현한다.

### iGate의 기대효과

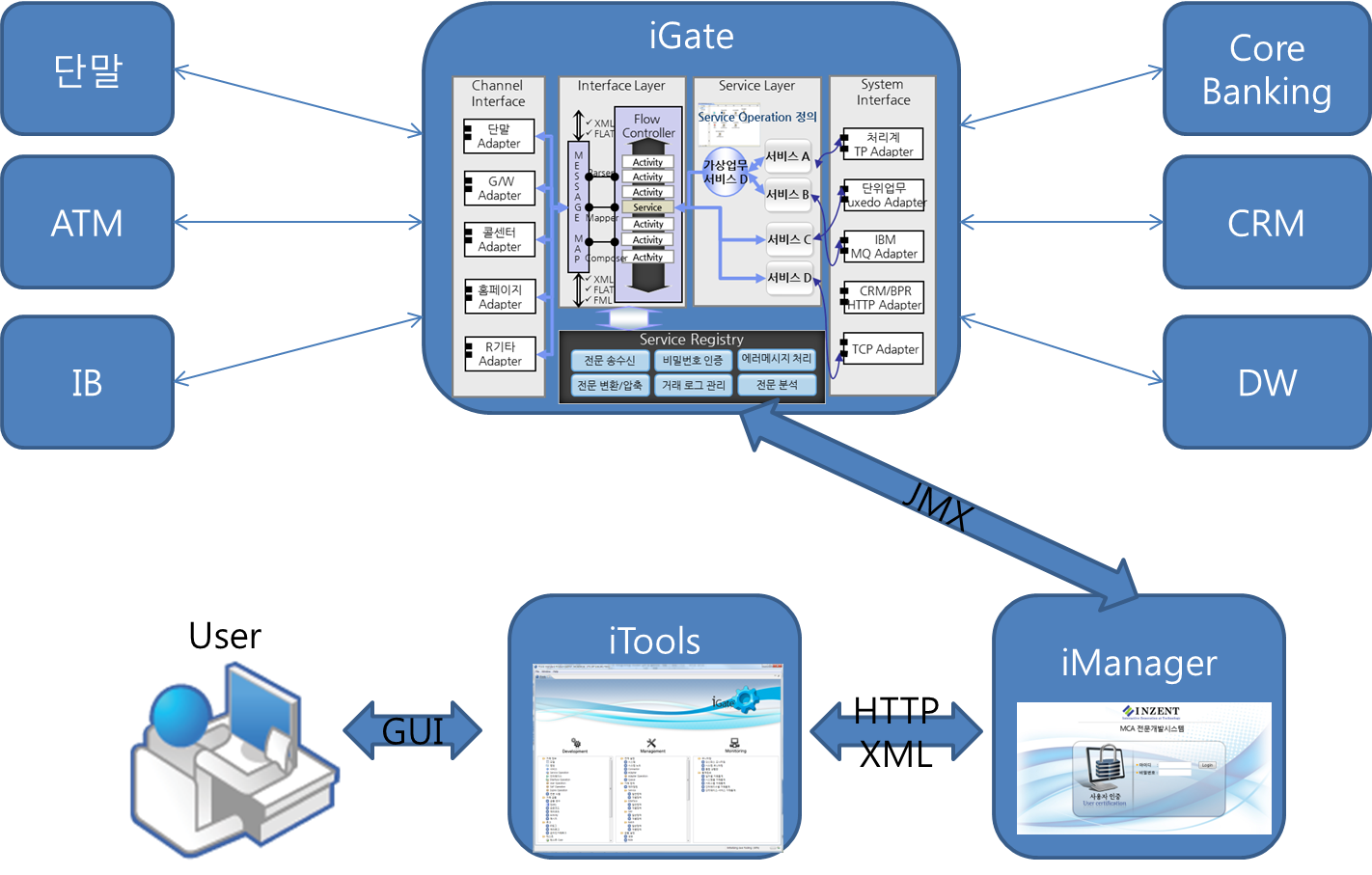
iGate는 내/외부 서비스의 유연한 통합 실현을 통해 업무 특성에 맞는 Business Agility 확보하고, 서비스 Reusability를 극대화하여 Time to Market을 실현한다.

다음과 같은 효과를 얻을 수 있다**.**

* 비즈니스 경쟁력 강화
  + 분산된 Business Middle Layer 통합
  + 고객 반응정보, 접촉이력에 대한 통합된 채널 로그를 통해 고객분석을 위한 기초 데이터 제공
  + 일관된 Customer Experience 를 제공하여 고객 중심 세일즈 효과 및 교차판매 역량 강화
  + 수용 가능한 비즈니스 요건의 범위 확대함으로써 다양한 Business 모델 구성
* Middle layer Integration
  + 인터페이스 단순화 및 채널별 기능 중복 제거를 통한 효율성 확보
  + 다양한 형태의 통합 요건 수용
* 운영 및 유지보수 용이성 확보
  + 24\*365 서비스가 가능한 안정된 통합시스템 구축
  + Middle Layer의 통합으로 중앙집중적인 통합 운영관리가 가능함
  + 운영관리 및 개발환경 제공

## 시스템 구성도

솔루션은 실행환경인 iGate와 관리환경인 iManager 그리고 개발환경인 iTools로 구성되어있다.



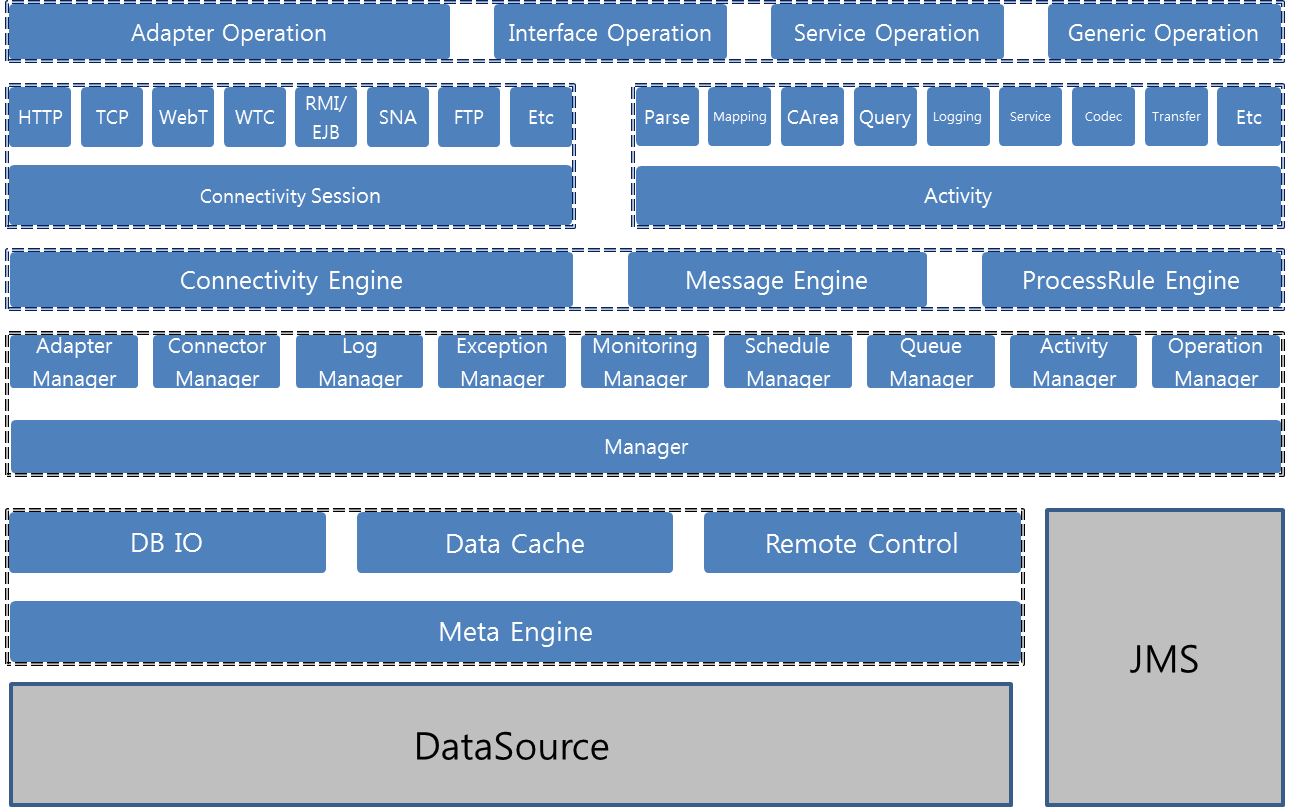
[그림 ‑] 시스템 구성도

# 실행 환경

본 장에서는 iGate의 실행 환경 아키텍처와 그 기능들을 기술한다.

## iGate Architecture

iGate는 [그림 2.1] iGate Runtime Architecture와 같이 여러 구성요소들이 계층을 이루어 상호작용을 한다.



**[그림 2‑１]** iGate Runtime Architecture

## 구성 요소

### Meta Engine

Meta Engine은 iGate가 동작하는데 필요한 각종 설정 데이터들을 관리하고, 이들 데이터가 필요로 하는 각종 구성 요소에 제공하는 역할을 한다.

|  |  |
| --- | --- |
| **주요 기능** | **비고** |
| **DBMS IO** | DB에 저장된 각종 Meta-data를 읽어 이를 Value-Object로 변환시킨다. |
| **Data Cache** | DB 장애 대응 및 DB IO 개선을 위해 각종 Meta-data를 JVM heap에 cache한다. |
| **Remote Control** | 거래처리 중에도 각종 Meta-data의 변경 내용이 거래에 적용된다. |

### Connectivity Engine

Connectivity Engine는 각종 연계 시스템들과 물리적인 통신을 담당하면서, 이들과의 데이터 송수신을 전담하여 처리한다.

iGate는 각종 상이한 통신 프로토콜을 보다 효율적으로 지원하기 위해서 각종 통신 프로토콜마다 이에 대응하는 Connection-Session이 제공된다.  
이를 통하여 여러 다양한 통신 프로토콜을 내부 표준으로 변환하여 일관되게 데이터 송수신을 수행한다

* In-House System

iGate는 각종 In-House System들과 연계를 위해서 아래와 같은 Connection-Session을 제공한다.

|  |  |
| --- | --- |
| **프로토콜** | **비고** |
| **HTTP** | HTTP(HyperText Transfer Protocol RFC2616)를 사용하여 전문을 주고 받는 통신 방식을 지원한다. |
| **iGate@Link** | In-House System 연계 지원용 API가 iGate@Link이다. 이 API를 사용한 통신 방식을 지원한다. |
| **TCP/IP** | TCP/IP를 사용하여 일정한 길이정보를 전송데이터에 포함시키는 등의 방식으로 전문을 주고 받는 통신 방식을 지원한다. |
| **Java™ RMI** | Java™ Remote Method Invocation을 사용하여 연계시스템과 전문을 주고 받는 통신 방식을 지원한다. |
| **FTP** | FTP(File Transfer Protocol RFC959)를 사용하여 연계 시스템과 파일을 주고 받는 통신 방식을 지원한다. |

* Middle-Ware

iGate는 다양한 Well-Known Middle-Ware들과의 연동을 지원한다.

|  |  |  |
| --- | --- | --- |
| **이름** | **비고** | |
| **Tmax** | Tmax사의 Java 라이브러리인 WebT, WebTAsync을 사용하여 Tmax Service로 전문을 주고 받는 통신 방식을 지원한다. | |
| **Tuxedo** | Oracle사의 WTC(WebLogic Tuxedo Connector), Jolt와 Tmax사의 JTC(JEUS-Tuxedo Connector)를 사용하여 Tuxedo Service와 전문을 주고 받는 통신 방식을 지원한다. | |
| **VTAM** | IBM사의 SNA LU0 프로토콜을 사용하여 Mainframe Service와 전문을 주고 받는 통신 방식을 지원한다. | |
| **Web Application Server** | HTTP | 보편적인 WAS(JEUS, Weblogic, Websphere)에서 기동되는 서비스들과 연계를 하기 위해 HTTP 통신 프로토콜을 지원한다. |
| EJB | EJB(Enterprise JavaBeans)을 사용하여 WAS에 설치된 서비스들과 전문을 주고 받는 통신 방식을 지원한다 |
| RMI | Java™ Remote Method Invocation을 사용하여 WAS에 설치된 서비스들과 전문을 주고 받는 통신 방식을 지원한다. |

* Application Framework

iGate는 다양한 Application Framework들과의 연동을 지원한다.

|  |  |
| --- | --- |
| **이름** | **비고** |
| **ProFrame** | Tmax사의 Application Framework 제품인 ProFrame과 전문을 주고 받는 통신 방식을 지원한다. |
| **AnyFrame** | SDS사의 Java 기반 Application Framework 제품인 AnyFrame과 전문을 주고 받는 통신 방식을 지원한다. |
| **SYSTEMiER** | SDS사의 C 기반 Application Framework 제품인 SYSTEMiER와 전문을 주고 받는 통신 방식을 지원한다. |
| **DevOn-Java** | LG CNS사의 Java 기반 Application Framework 제품인 DevOn-Java와 전문을 주고 받는 통신 방식을 지원한다. |
| **DevOn-C** | LG CNS사의 C 기반 Application Framework 제품인 DevOn-C와 전문을 주고 받는 통신 방식을 지원한다. |
| **BANCS** | Financial Network Services사의 Application Framework 제품인 BANCS와 전문을 주고 받는 통신을 지원한다. |

### Message Engine

Message Engine은 iGate에서 다루는 데이터 처리 기능이 집약된 Component이다.

데이터 해석, 데이터 조작, 데이터 생성 등의 기능을 각각 Parser, Mapper, Composer 수행하고 이들이 거래 처리 과정에 참여하여 이기종간의 데이터 교환을 보다 효율적으로 처리한다.

* Parser

iGate의 Parser는 Connection-Session을 통해서 수신한 데이터를 해석하여 이를 Data-Object로 변환하는 역할을 한다.

이를 통해 각각의 연계 시스템들의 고유 전문 포맷을 모두 수용하고, 이를 표준화하여 해당 요청을 효율적으로 처리한다.

* Composer

iGate의 Composer는 표준화된 Data-Object로부터 Connection-Session을 통해서 송신할 데이터를 생성하는 역할을 한다.

이를 통해 각각의 연계 시스템들의 고유 전문 포맷을 모두 수용하고, 이를 표준화하여 해당 요청을 효율적으로 처리한다.

Parser와 Composer는 항상 짝을 이루어 동작한다.

* Mapper

iGate의 Mapper는 표준화된 Data-Object를 생성하는 역할을 수행한다. Mapper는 이를 위해서 사칙 연산 및 함수 호출과 같은 다양한 연산을 수행한다.

iGate는 Mapper를 통해서 연계 시스템의 요청 혹은 응답 전문을 새로운 Data-Object를 만들고, Composer를 통해서 최종 송신 데이터를 작성한다.

### ProcessRule Engine

iGate에서는 각종 비즈니스 로직을 Activity 혹은 Operation 형태로 작성한다. 이렇게 사용자가 작성한 각종 비즈니스 로직을 물리적으로 실행하는 Component가 ProcessRule Engine이다.

iGate는 거래 처리에 필요한 각종 로직들이 담겨 있는 Activity 혹은 Operation을 ProcessRule Engine을 통해서 실행함으로 사용자가 원하는 임의의 요구를 실현시킨다.

* Activity

Activity는 가장 기초 단위가 되는 비즈니스 로직이 담겨지는 Component인다. 이 Activity는 Java™ Language로 작성된다.

Java™로 작성하기 때문에 Java™로 작성된 각종 외부 Library를 참조할 수 있을 뿐만 아니라, Java™ Language에서 얻은 수 있는 각종 장점(예들 들어 Object-Oriented)들을 사용하여 비즈니스 로직을 구현할 수 있다.

* **Operation**

Operation은 iTools를 사용하여 작성되는 비즈니스 로직이다. 이는 Activity / Operation간의 흐름을 정의 함으로써 비즈니스 로직을 작성한다.

Operation은 Activity와 달리 Component-Oriented 방식으로 작성된다. 모든 Activity와 Operation은 쉽게 참조될 수 있도록 각종 정보들이 공유되고, 사용자는 iTools(GUI기반의 개발환경)에서 제공되거나 작성된 Activity와 Operation들 참조하여 직관적으로 표현되는 처리 흐름을 작성한다.

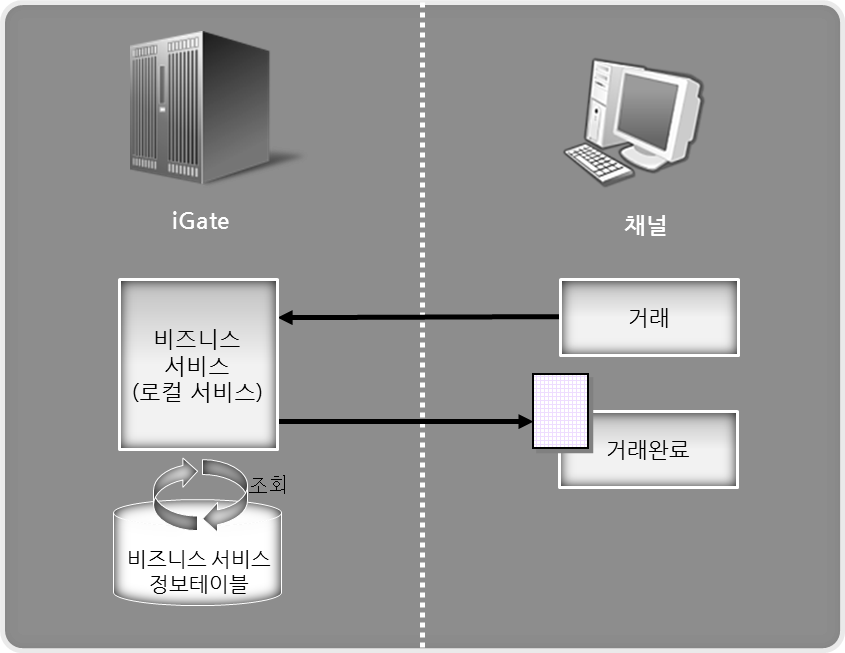
이처럼 직관적으로 작성된 비즈니스 로직을 iGate는 ProcessRule Engine를 통해서 물리적으로 실행하고, 이를 통해서 거래가 수행된다.

## 거래 유형

### 로컬 서비스 거래

로컬 서비스는 iGate내 정의되어 있는 서비스이다.

외부 시스템과 이러한 로컬 서비스를 연계하는 거래 유형이다.

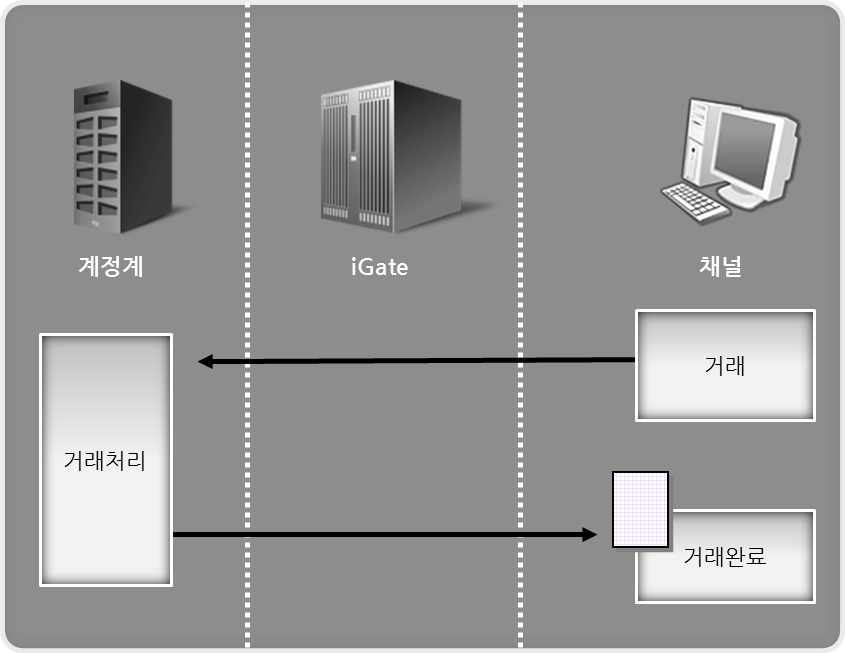


**[그림 2‑２] 로컬 서비스 흐름도**

### 일반 서비스 거래

일반 서비스는 외부 시스템에 존재하는 서비스이다.

일반 서비스 거래는 기타 외부 시스템에서 일반 서비스를 사용하여 거래하는 유형이다. 인터페이스 시스템으로부터 요청 메시지가 입력되면, 해당 서비스가 수행된 후 요청한 시스템으로 응답 메시지를 출력한다.  
동기, 비동기, PUSH방식의 거래가 있으며 책임자 승인거래, 대량 출력 거래, 단순출력 거래 등이 일반거래유형에 해당한다.



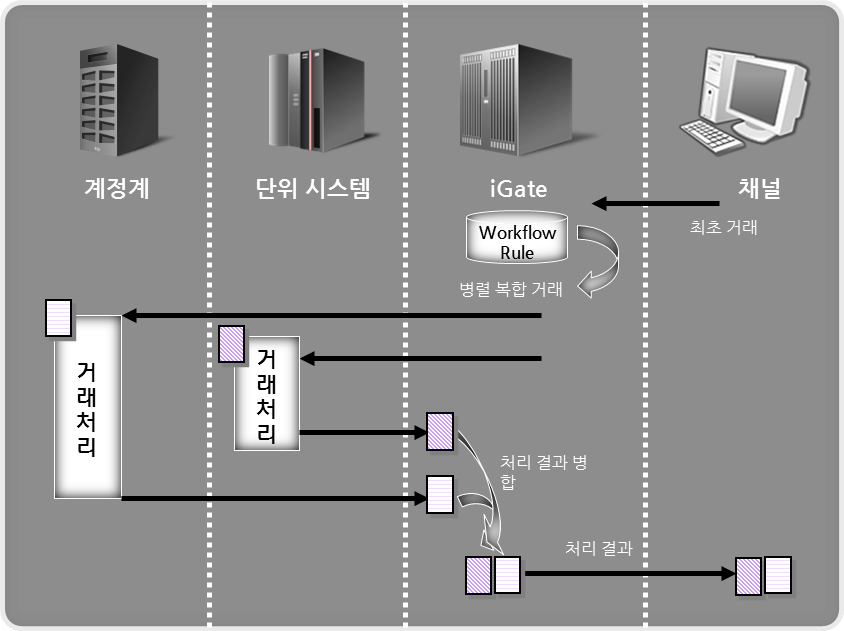
[그림 ‑] 일반 서비스 흐름도

### 복합 서비스 거래

복합 서비스는 여러 서비스들을 조합하여 정의되는 가상의 서비스이다.

iGate는 SOA을 기반으로 작성된 서비스(재사용이 가능한 세분화된 서비스)들을 유기적으로 조합하여 다양한 복합 거래를 지원한다.  
업무 개발자는 Service Operation Designer를 이용하여 편리하게 복합 서비스를 작성할 수 있다.

복합 서비스는 비즈니스 변화를 신속하게 수용하고 적용할 수 있는 유연성을 제공한다.



[그림 ‑] 복합서비스 흐름도

### 대외 거래

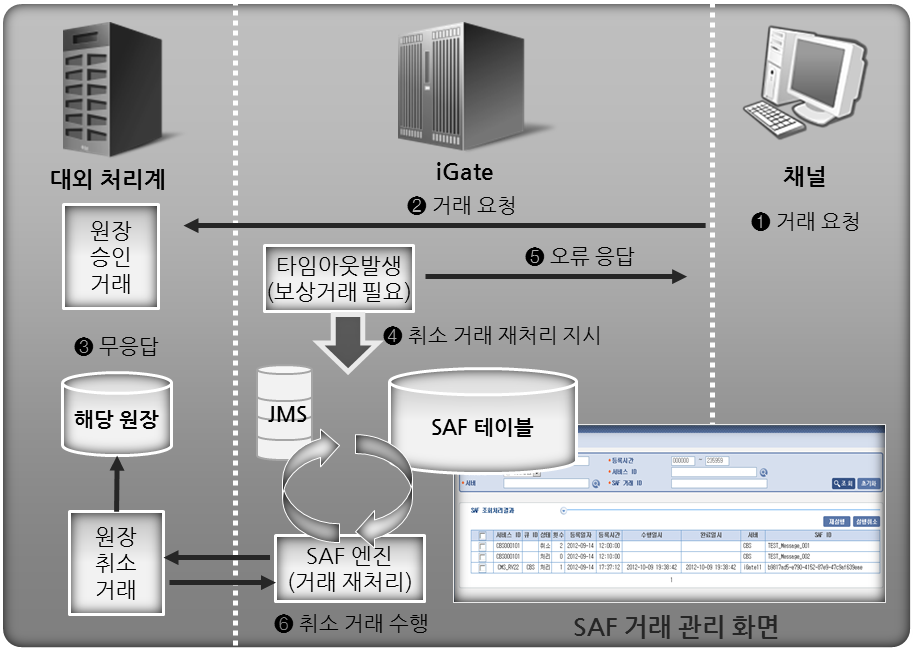
대외 거래는 대외기관과 수행하는 거래이다.

iGate는 거래보장 및 정합성을 제공하기 위한 SAF(Store And Forward)거래와 대외기관이 필요로 하는 파일을 처리하기 위한 파일송수신거래를 지원 한다.

SAF(Store and Forward)거래

거래보장 및 거래정합성을 지원하기 위해서 SAF엔진 기반의 재처리기능을 제공한다.

iGate는 거래보장을 지원하기 위해 메시지를 SAF테이블에 저장하고, 거래처리 중 에러가 발생 하면 메시지를 재전송한다. 또한 거래정합성을 지원하기 위해 응답 처리 중 에러가 발생하면 SAF테이블에 취소전문을 저장하고 취소거래를 재전송한다.



[그림 ‑] SAF거래처리 흐름도

파일송수신거래

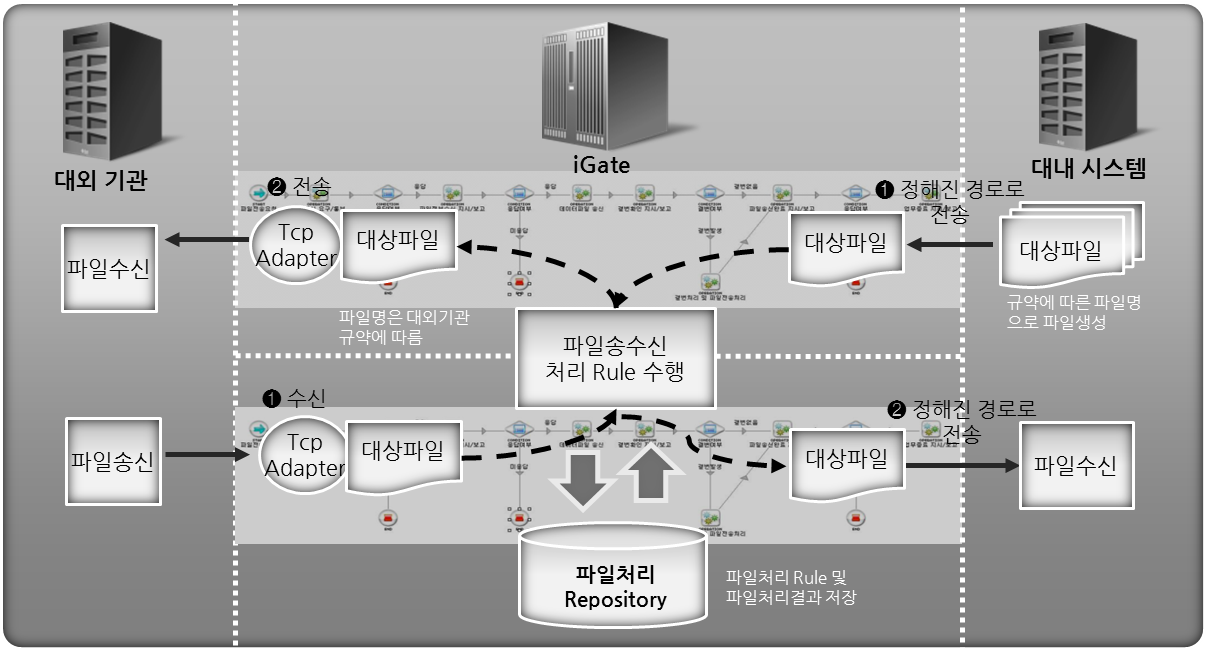
대외기관과의 파일전송처리를 위하여 지원되는 거래로서 EDI방식과 표준FTP방식이 있다.

EDI(Electronic Data Interchange) 방식

EDI 전송 규약에 따라 파일을 정해진 크기만큼 나누어 송/수신하고 파일 송/수신 규약 처리는 업무 처리 Rule을 구성하여 처리한다. 동일한 규약의 신규 기관이 추가되면 기존의 업무처리 Rule을 재사용하여 구현한다.

표준FTP방식

기관별 정의된 규약에 따라 표준 FTP 방식의 파일 송/수신 처리한다. 대내 시스템으로부터 수신한 파일을 파일 처리 Rule에 따라 송신하고, 대외 시스템으로부터 수신한 파일은 처리 Rule에 따라 대내 시스템으로 전송한다.

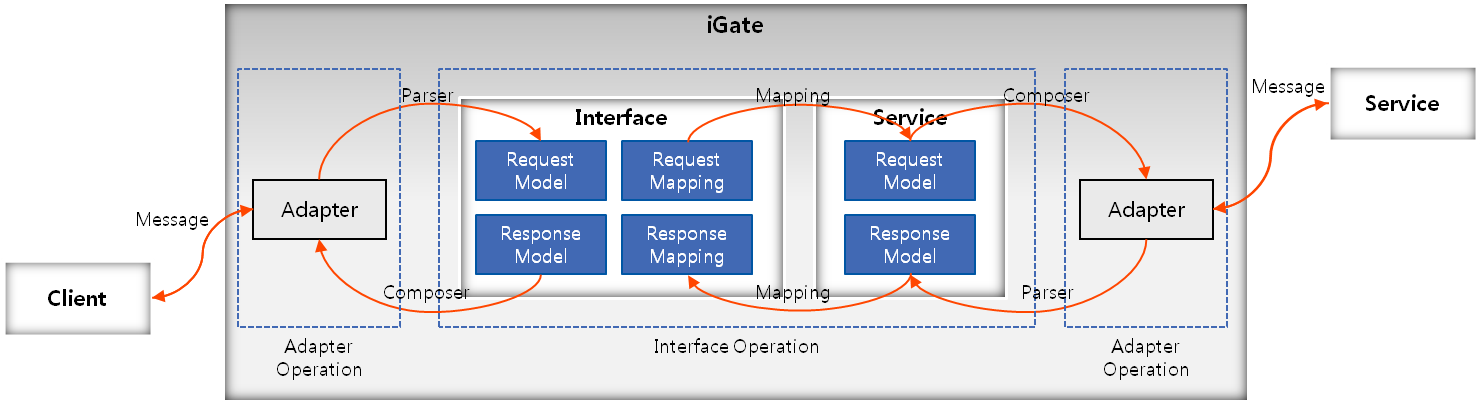


**[그림 2‑６] EDI방식의 파일송수신거래**

## iGate 주요 기능

### Message Process

iGate에서는 입력 데이터를 Data-Model로 Parsing하여 Message로 만들고, Message를 Composing하여 출력 데이터를 만든다. iGate 내부에서는 표준화된 Message를 통해서 다양한 Data의 포맷과 형식을 처리 한다.



**[그림 2‑７] Message Process**

Service

Service는 명확한 기능적인 의미를 지닌 소프트웨어 컴포넌트로, 고차원의 비즈니스 개념을 캡슐화 하고 있다. Service Provider와 Service Consumer는 메시지를 통해 서로 통신하고 Service Provider는 서비스 명세를 통해 서비스가 제공하는 기능과 이를 이용하기 위해 사용자와 주고 받아야 하는 메시지의 형식이 정의되어 있다.

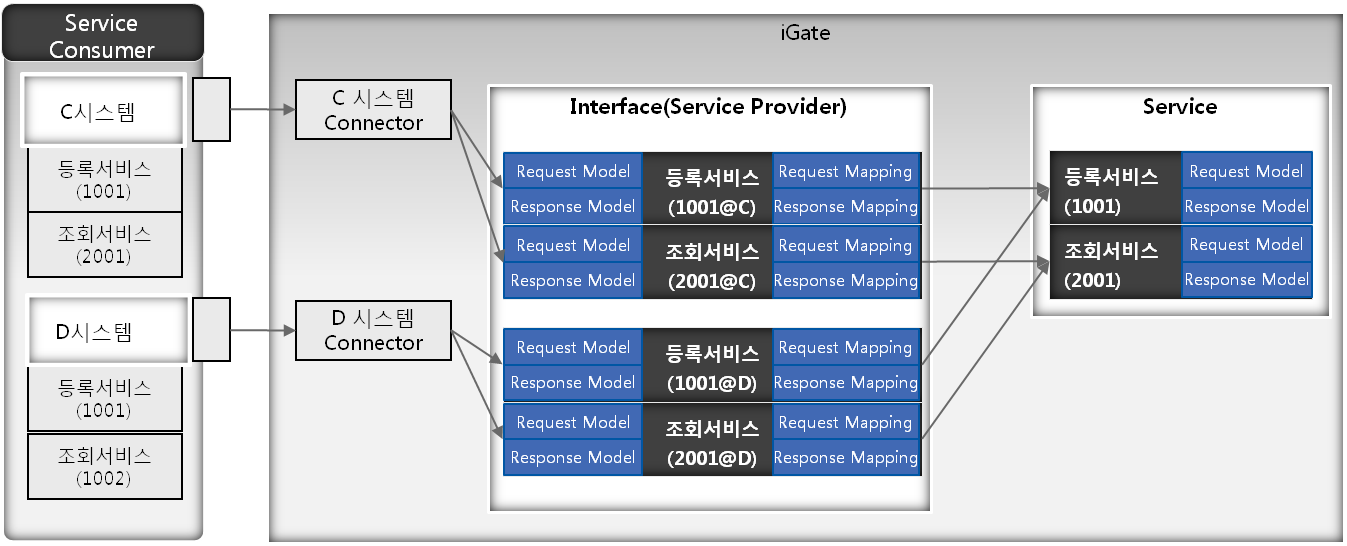
iGate에서 Service는 Service Provider가 제공한 Service의 기본정보와 메시지정보를 정의한 것이다.



**[그림 2‑８] Service 등록**

Interface

Interface는 Service Provider가 제공하는 Service를 사용할 수 있도록 Service Consumer가 정의한 Proxy-Service를 의미한다. Interface에는 Service Consumer가 사용 하는 Request / Response Data-Model 과 Service Provider에서 정의한 Data-Model 간에 Mapping-Rule이 포함되어 있다.



[그림 ‑] Interface 등록

Da**t**a-Model

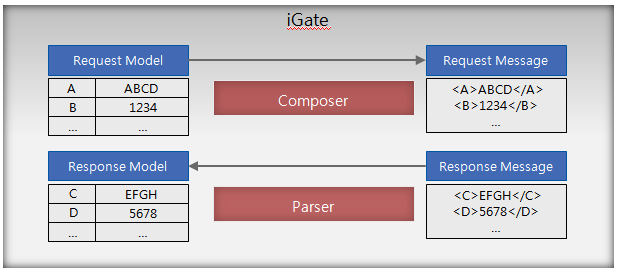
Data의 논리적인 구조를 정의한 iGate 고유에 포맷이다.

iGate는 이를 통해서 데이터를 해석하고 메시지로 변환한다.

Message

메시지는 데이터를 Data-Model로 해석한 결과로서, 해당 Data-Model의 논리적인 구조와 물리적인 정보가 결합되어 있다.

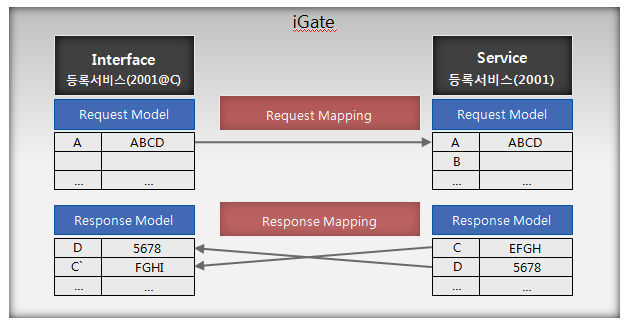
iGate에서는 다양한 유형의 데이터를 처리하기 위해 Parser, Composer을 이용하여 표준화된 메시지를 생성한다.



**[그림 2‑１０] 입출력 데이터 Parsing Composing**

Mapping-Rule

Mapping-Rule은 Interface와 Service간 Data-Model이 다르게 정의된 경우, 이 두 Data-Model 사이의 관계를 정의하고 변환하는 규칙이다.



[그림 ‑] Mapping-Rule 처리

### 거래 제한

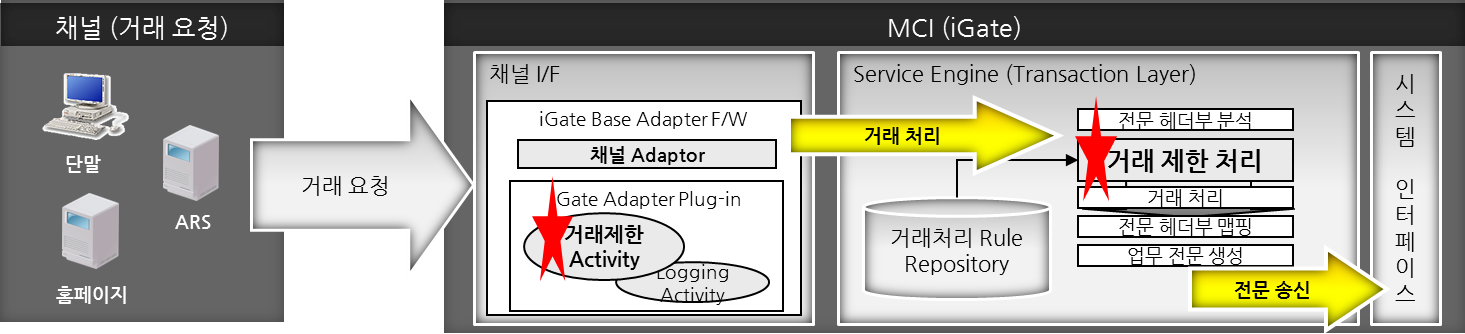
거래 제한은 특정 거래 요청을 제어하는 기능이다.

이를 통해서 특정시스템에 장애가 발생하거나 거래가 폭주하는 경우에 다른 시스템에 영향을 최소화하여 시스템 안정성을 유지할 수 있다. 또는 계획 정지와 같은 일상 업무 수행을 위해 사용된다.

시간과 메시지의 설정이 가능하여 효율적인 거래 통제기능을 제공한다.

거래제한의 종류는 다음과 같다.

|  |  |
| --- | --- |
| **종류** | **설명** |
| **인터페이스시스템** | 특정 인터페이스시스템에 거래제한을 설정한다 |
| **서비스시스템** | 특정 서비스시스템에 거래제한을 설정한다. |
| **인터페이스ID** | 특정 인터페이스ID로 거래제한을 설정한다. |
| **서비스ID** | 특정 서비스ID로 거래제한을 설정한다. |
| **IP별** | 특정 IP 또는 IP대역별로 거래제한을 설정한다. |



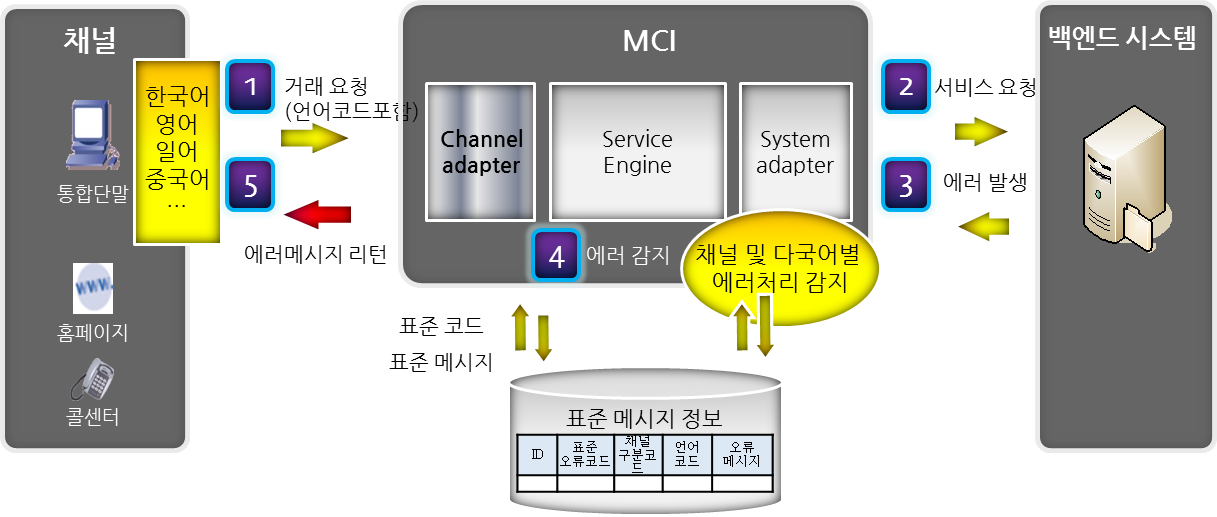
[그림 ‑] 거래제한기능

### 표준 메시지

표준 메시지는 상황에 맞게 메시지를 출력하기 위해 등록한다.

시스템간 거래 시 발생하는 에러에 대응하는 표준 메시지를 시스템이나 언어코드에 따라 가변적으로 출력하여 메시지의 가독성을 높여주는 기능을 제공한다.

또한 출력 메시지가 변경되는 경우 적용된 표준 메시지를 변경하여 일괄적용이 가능하다.



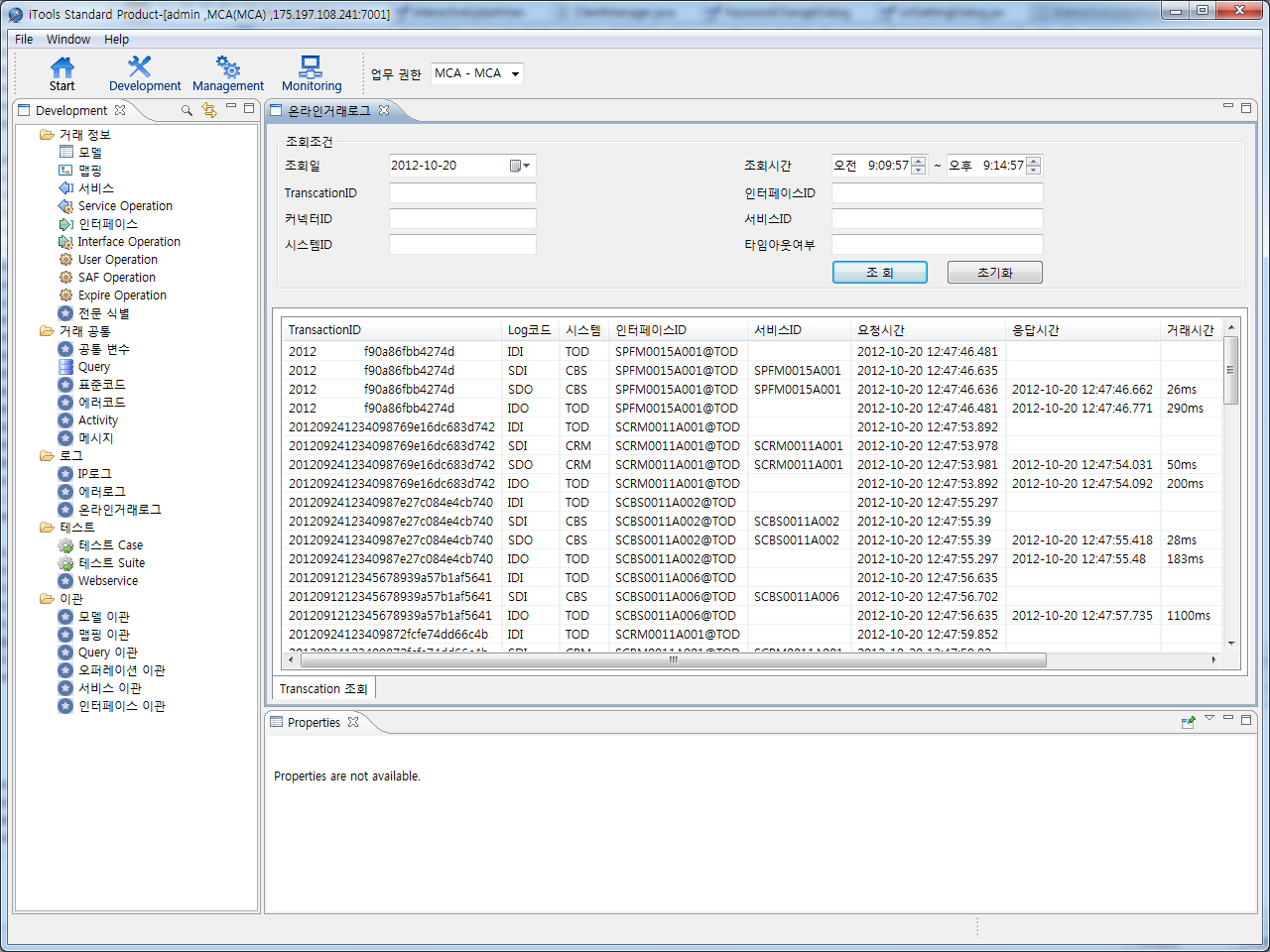
[그림 ‑] 에러발생에 따른 표준코드 및 메시지처리 흐름도

### 거래추적로그

인터페이스 시스템과 iGate 또는 iGate와 서비스 시스템 간의 거래정보를 전송단계별로 적재하는 로그이다.

표준헤더에서 필요한 정보를 거래추적로그로 저장하여 거래 분석에 편리함을 제공한다.

DB장애 시에도 거래원문은 File로 저장되어 있어, 장애 복구 후 거래추적로그를 적재하여 확인할 수 있다.



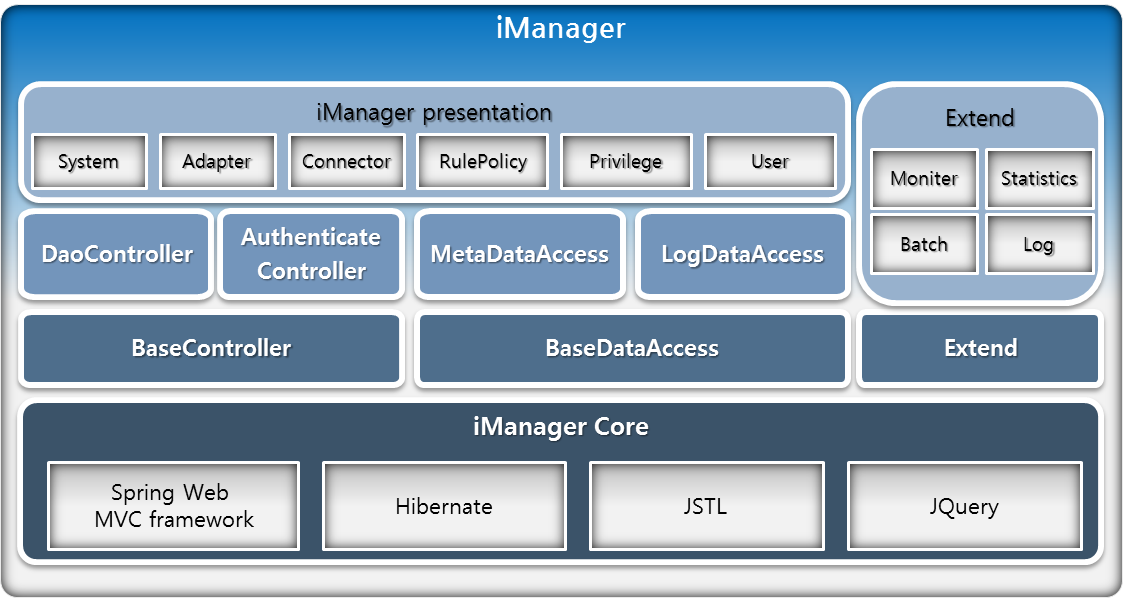
**[그림 2‑１４] 거래추적로그 조회화면**

# 관리 환경

본 장에서는 iGate의 관리 환경 아키텍처와 그 기능들을 기술한다.

## iManager Architecture

iManager은 Spring Web MVC를 Base로 하여 다양한 데이터의 조회 및 관리를 유연하게 처리할 수 있도록 개발됨

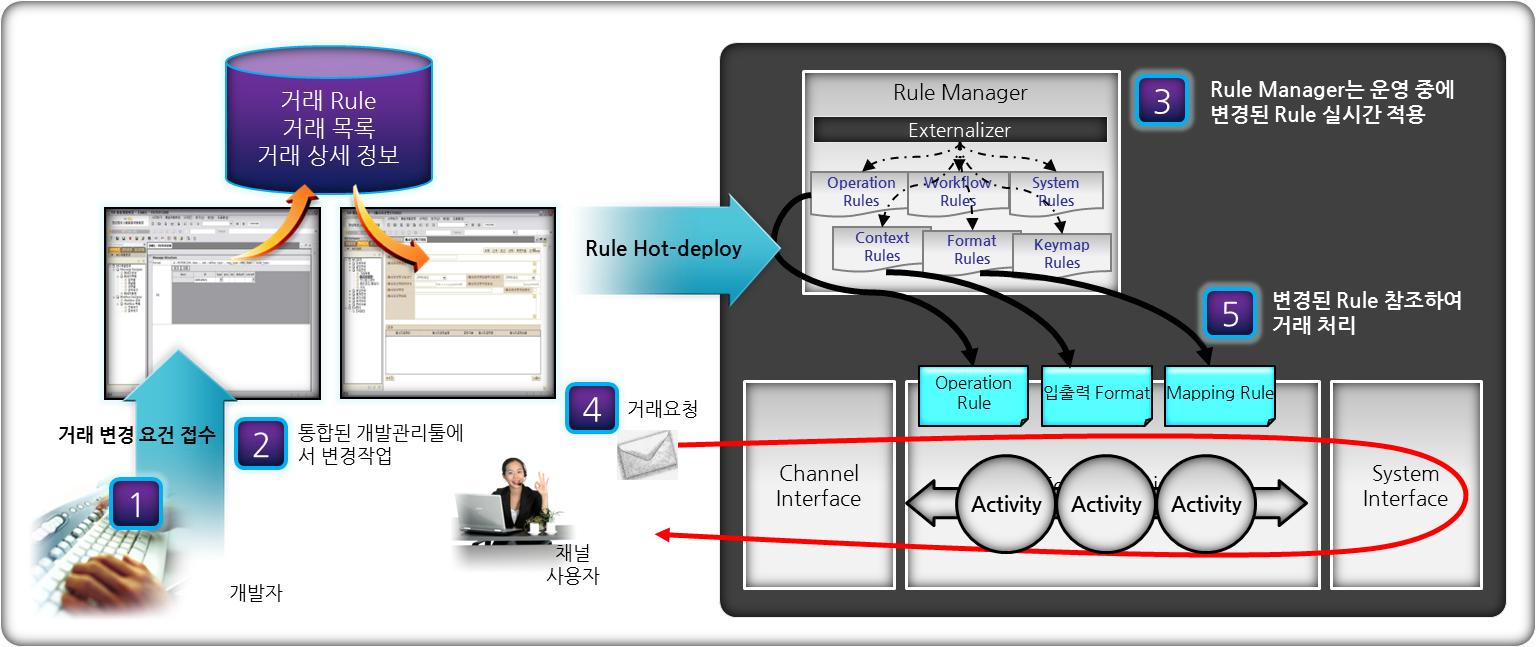


**[그림 3‑１]** iManager Architecture

## Meta 데이터 관리

iGate에서 사용되는 모든 Meta 데이터는 각각에 대응되는 Meta 관리 Component가 있으며, 이를 통해서 Meta 정보가 관리된다. Meta 관리 Component를 통하여 시스템 중단 없이 Meta 정보를 실시간으로 적용할 수 있다.

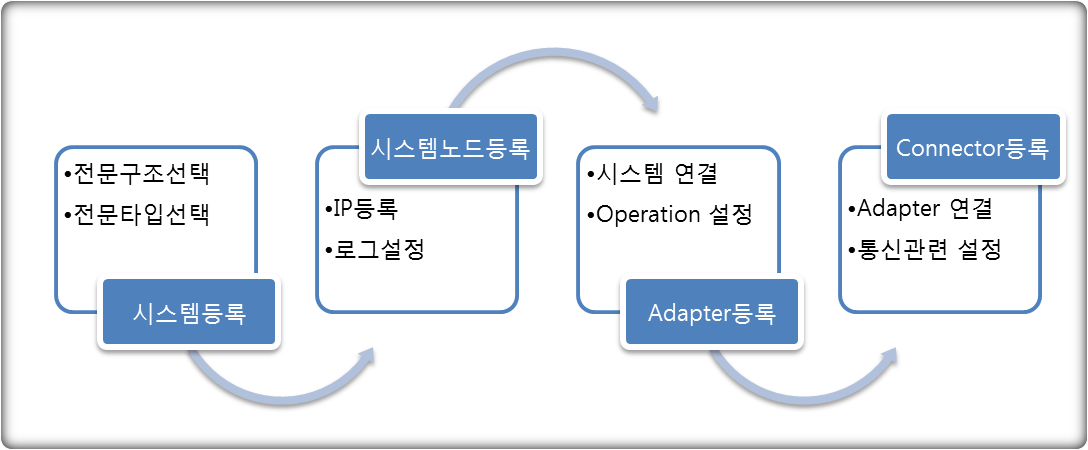
대표적인 Meta 데이터는 Rule Policy, Data-Model, Mapping-Rule, Operation 등이 있다.



[그림 ‑] Meta정보 실시간적용 흐름도

## 연계 시스템 정보 관리

iManager는 iGate와 연계할 시스템의 정보를 정의하기 위해 System, System Node, Adapter 그리고 Connector 정보를 설정한다.



[그림 ‑] 연계 설정 흐름도

### System 설정

연계할 시스템 및 전문 포맷에 대하여 정의하는 정보이다.

시스템정보에는 요청/응답 Data-Model, Charset, Parser, Composer등에 정보를 설정한다.

|  |  |
| --- | --- |
| 등록항목 | **설명** |
| **ID** | 연계 시스템의 식별자 |
| **Charset** | Parser 와 Composer에서 인코딩시 사용되는 문자셋을 입력한다. |
| **Endian** | Byte Order 방식을 지정한다. |
| **요청 Data-Model** | 요청 Data를 분석할 수 있는 Data-Model을 선택한다. |
| **응답 Data-Model** | 응답 Data를 분석할 수 있는 Data-Model을 선택한다. |
| **Parser** | 연계시스템에서 사용할 Parser를 선택한다. |
| **Composer** | 연계시스템에서 사용할 Composer를 선택한다. |
| **설명** | 연계시스템에 대한 설명을 입력한다. |

### System Node 설정

연계 시스템의 물리적인 정보를 정의한다.

System Node 정보에는 SystemID, SystemNode, NodeIP, IP로깅여부, 접속허용여부 정보를 설정한다.

|  |  |
| --- | --- |
| **등록항목** | **설명** |
| **SystemID** | System의 식별자. |
| **SystemNode** | 연계 시스템의 SystemNode의 식별자. |
| **Node IP** | **시스템노드의** IP주소를 입력한다. |
| **IP로깅여부** | IP**로그의** 출력 여부를 선택한다. |
| **접속허용여부** | Client의 접속 허용 여부를 선택한다. |

### Adapter 설정

연계 시스템에서 발생하는 통신 Event에 따라 통신 처리 로직을 선택한다.

AdapterID, 시스템ID, 설명, 이벤트List를 설정한다.

|  |  |
| --- | --- |
| **등록항목** | **설명** |
| **AdapterID** | AdapterID를 입력한다. |
| **시스템ID** | 연계할 시스템ID를 선택한다. |
| **설명** | Adapter에 대한 설명을 입력한다. |
| **이벤트리스트** | Event에 따른 Operation을 선택한다. |

Event유형은 아래와 같다.

|  |  |
| --- | --- |
| **Event** | **설명** |
| **create** | Session이 생성될 때 처리하는 이벤트 |
| **read.response** | 응답 데이터를 읽었을 때 처리하는 이벤트 |
| **read.request** | 요청 데이터를 읽었을 때 처리하는 이벤트 |
| **write.response** | 응답 데이터를 생성할 때 처리하는 이벤트 |
| **write.request** | 요청 데이터를 생성할 때 처리하는 이벤트 |
| **extract** | 연계 시스템에서 데이터를 추출할 때 처리하는 이벤트 |
| **exception** | 예외상황이 발생했을 때 처리하는 이벤트 |
| **close** | Session이 종료될 때 처리하는 이벤트 |

### Connector 설정

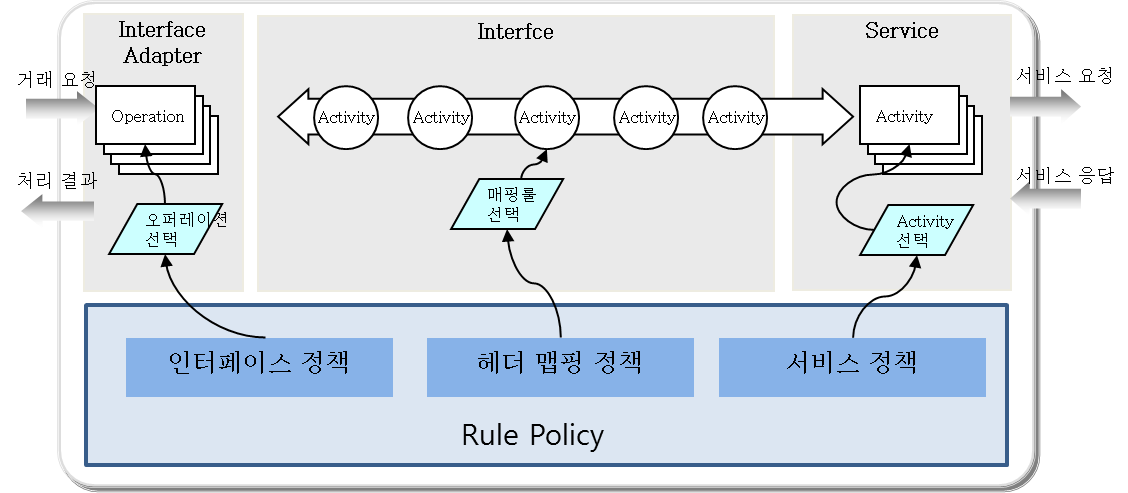
연계할 시스템에 대한 통신관련 정보를 설정한다.

Connector 설정에는 Connector 유형 및 Timeout, Port 등의 통신속성에 관하여 등록한다.

|  |  |
| --- | --- |
| **등록항목** | **설명** |
| **ConnectorID** | Connector 식별자. |
| **Class** | Connector의 유형을 선택한다. |
| **Log Level** | Connector에 적용될 로그레벨을 선택한다. |
| **RequestDirection** | Connector의 입/출력 방향을 선택한다. |
| **시스템ID** | 연계시스템ID를 선택한다. |
| **시스템노드** | 연계시스템노드를 선택한다. |
| **AdapterID** | AdapterID를 선택한다. |
| **자동기동여부** | iGate시스템이 기동할 때 Connector의 자동기동여부를 선택한다. |
| **설명** | Connector에 대한 설명을 입력한다. |
| **속성리스트** | Connector에 대한 통신속성을 입력한다. |
| **적용서버ID** | Connector을 적용할 서버ID를 선택한다. |

## 정책 설정 관리

거래처리의 유연성을 향상시키고 시스템 관리자의 관리 편의성을 위하여 거래처리 관련 정책을 제공한다.



[그림 ‑] 정책 설정 개념도

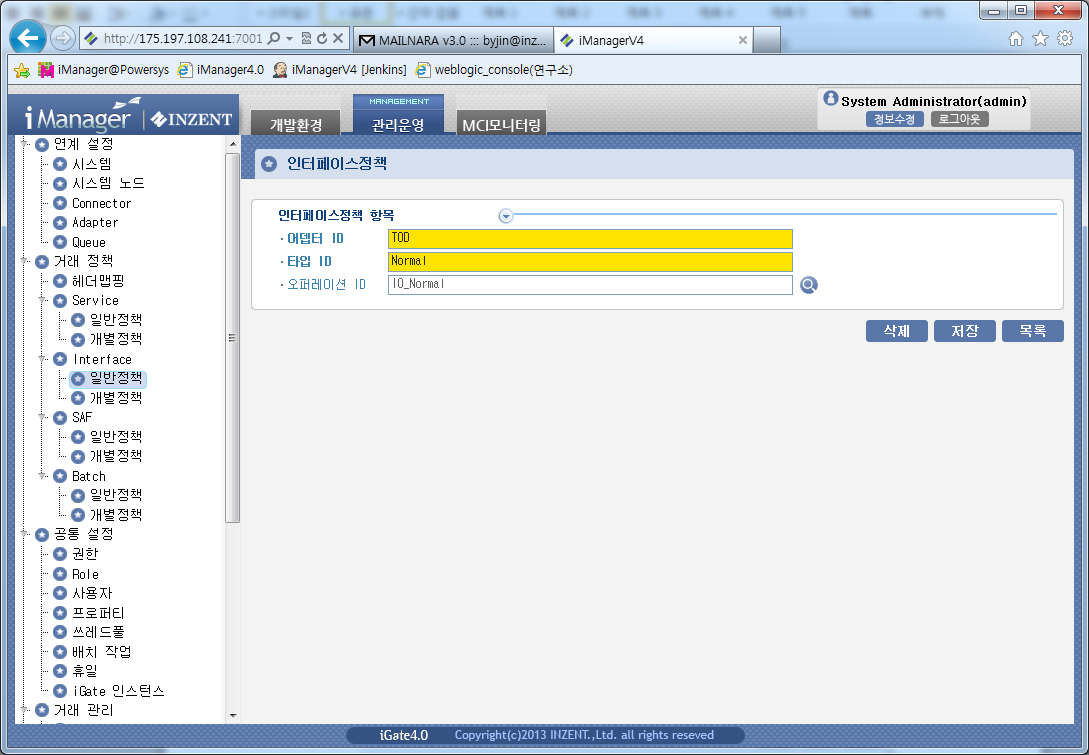
### 인터페이스정책

인터페이스 시스템과의 연계를 처리하기 위하여 생성하는 정책으로 일반정책과 개별정책이 있다.

일반정책

임의의 거래 요청을 적절한 Operation을 사용하여 거래처리 되도록 설정하는 정책이다.

|  |  |
| --- | --- |
| **등록정보** | **설명** |
| **Adapter ID** | 인터페이스 시스템의 Adapter ID를 지정한다. |
| **Type ID** | 인터페이스정책을 그룹화하여 관리하기 위해 선택한다 |
| **Operation ID** | 인터페이스 시스템과 연계 시 수행해야 할 비즈니스 로직을 가지고 있는 OperationID를 지정한다. |

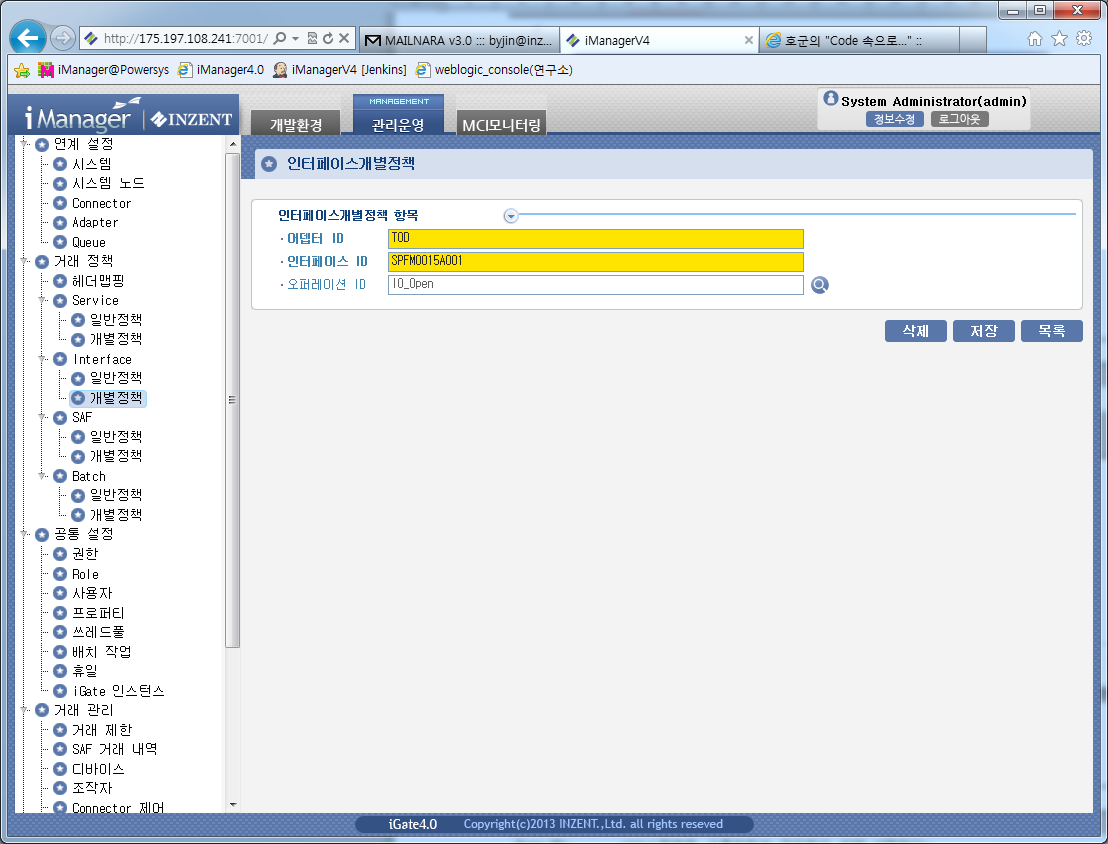


**[그림 3‑５] 인터페이스 일반정책 화면**

개별정책

특정 거래 요청을 지정된 Operation을 사용하여 거래처리 되도록 설정하는 정책이다.

|  |  |
| --- | --- |
| **등록정보** | **설명** |
| **Adapter ID** | 인터페이스 시스템의 Adapter ID를 지정한다. |
| **Interface ID** | 개별처리 할 인터페이스ID를 지정한다. |
| **Operation ID** | 인터페이스 시스템과 연계 시 수행해야 할 비즈니스 로직을 가지고 있는 OperationID를 지정한다. |



**[그림 3‑６]** 인터페이스 개별정책 화면

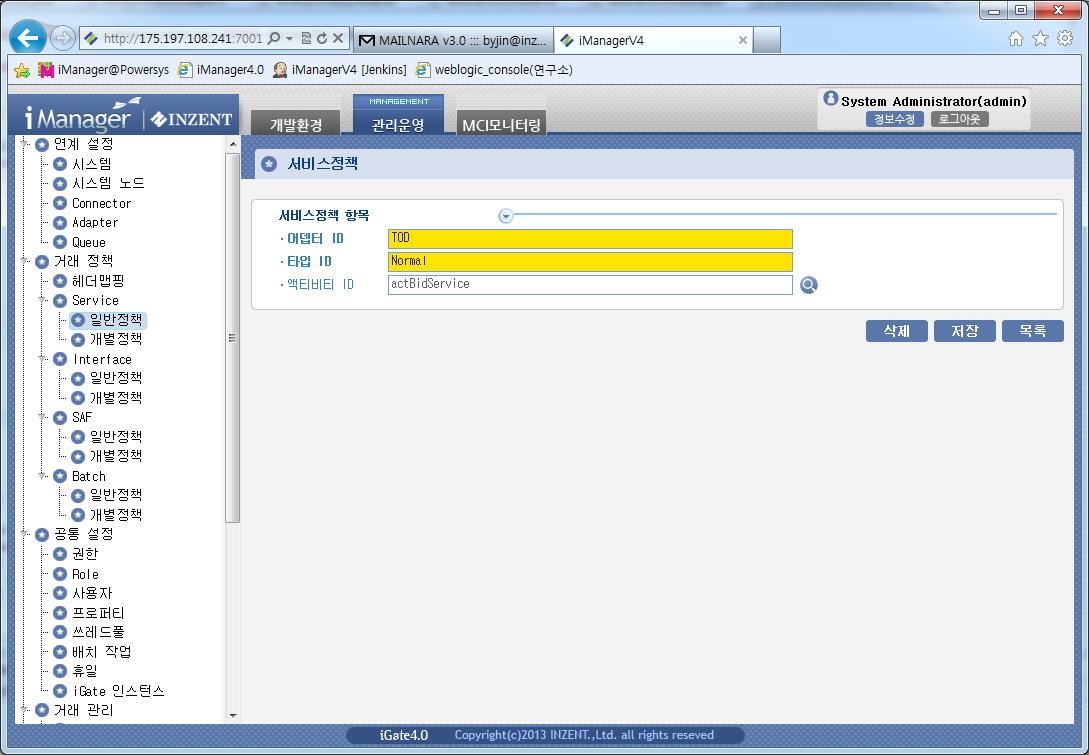
### 서비스 정책

서비스 시스템과의 연계를 처리하기 위하여 생성하는 정책으로 일반정책과 개별정책이 있다.

일반정책

임의의 서비스 요청을 적절한 Service Activity을 사용하여 처리 되도록 설정하는 정책이다.

|  |  |
| --- | --- |
| **등록정보** | **설명** |
| **Adapter ID** | 서비스 시스템의 Adapter ID를 지정한다 |
| **Type ID** | 서비스정책을 그룹화하여 관리하기 위해 선택한다 |
| **Activity ID** | 서비스와 연계방법 및 기능을 포함하는 Activity ID를 지정한다. |

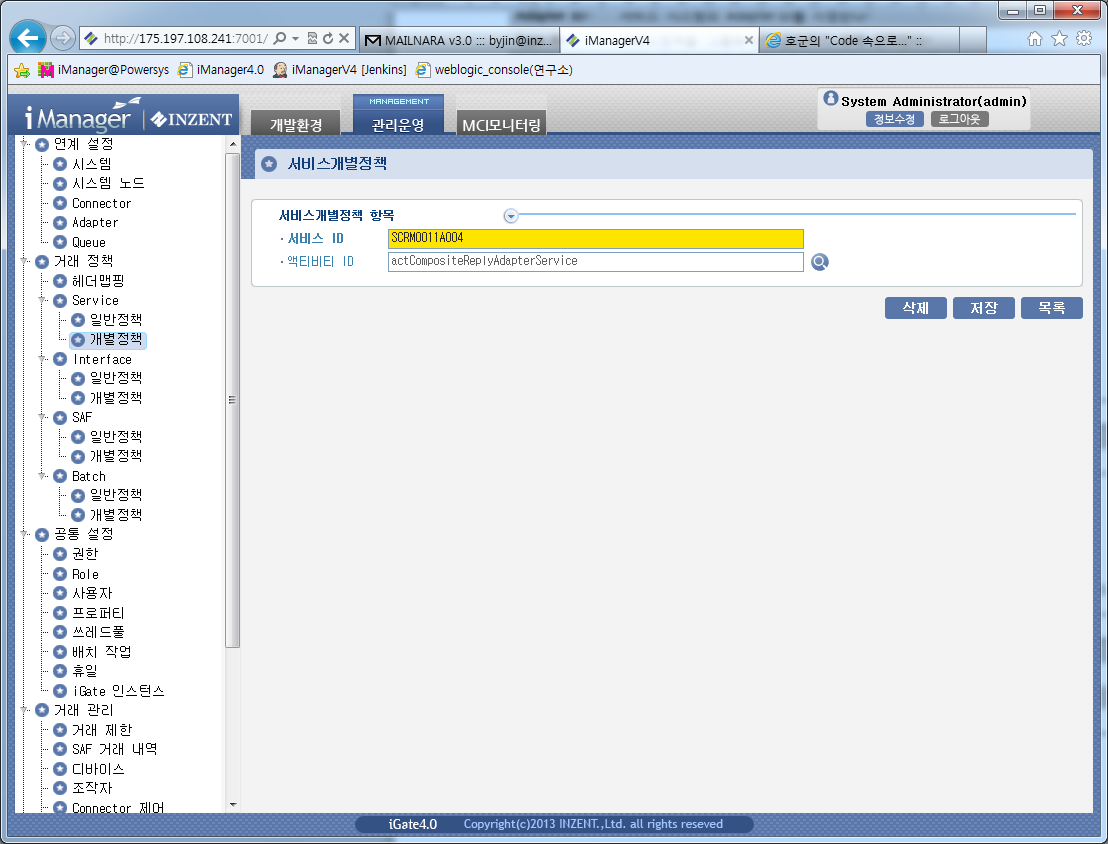


**[그림 3‑７]** 서비스 일반정책 화면

개별정책

특정 서비스 요청을 지정한 Service Activity을 사용하여 처리 되도록 설정하는 정책이다.

|  |  |
| --- | --- |
| **등록정보** | **설명** |
| **서비스 ID** | 개별처리 할 서비스 ID를 입력한다. |
| **Activity ID** | 서비스와 연계방법 및 기능을 포함하는 Activity ID를 지정한다. |

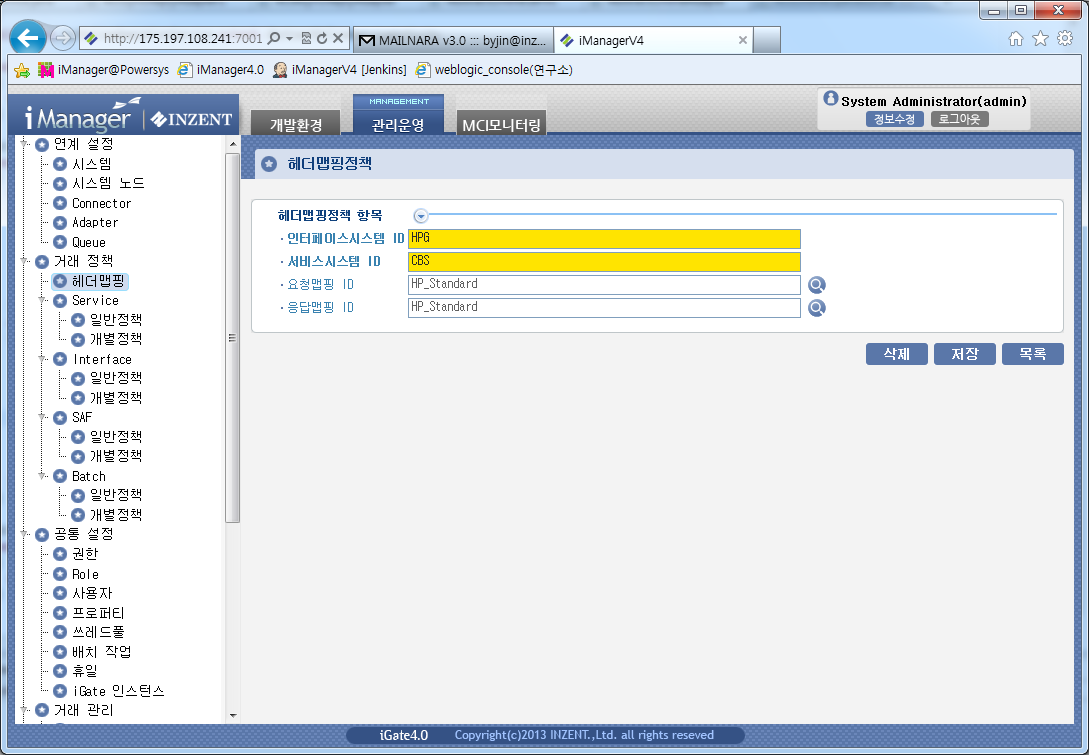


**[그림 3‑８] 서비스 개별정책 화면**

### 헤더 맵핑 정책

헤더 매핑 정책은 인터페이스 시스템과 서비스 시스템간의 헤더 Mapping-Rule을 지정하는 정책이다.

|  |  |
| --- | --- |
| **등록정보** | **설명** |
| **인터페이스 시스템ID** | 인터페이스시스템 ID를 입력한다 |
| **서비스 시스템ID** | 서비스시스템 ID를 입력한다. |
| **요청헤더 맵핑ID** | 요청전문 맵핑 시 사용할 헤더 Mapping-Rule을 등록한다. |
| **응답헤더 맵핑ID** | 응답전문 맵핑 시 사용할 헤더 Mapping-Rule을 등록한다. |



**[그림 3‑９]** 헤더맵핑정책 등록화면

## 사용자 및 권한 관리

iManager는 권한 설정, Role 설정 그리고 사용자 설정을 사용하여 다양한 수준의 권한 관리와 보안 관리를 지원한다.

### 권한 설정

권한에는 System타입과 business타입이 있으며, System타입은 iTools 및 iManager에서 사용되는 화면과 메뉴에 대하여 조회 및 편집 권한을 설정하고 Business타입은 등록된 Data-Model, Mapping-Rule, Operation등의 Meta 정보를 조회 및 편집할 수 있는 권한을 설정한다.

|  |  |
| --- | --- |
| **권한항목** | **설명** |
| **권한 ID** | 메뉴, 화면에 대한 권한 및 업무권한의 식별자. |
| **권한 Type** | System : 메뉴 및 화면에 대한 권한설정  Business : 업무에 대한 권한설정 |
| **권한 설명** | 권한에 대한 설명 |

### Role 설정

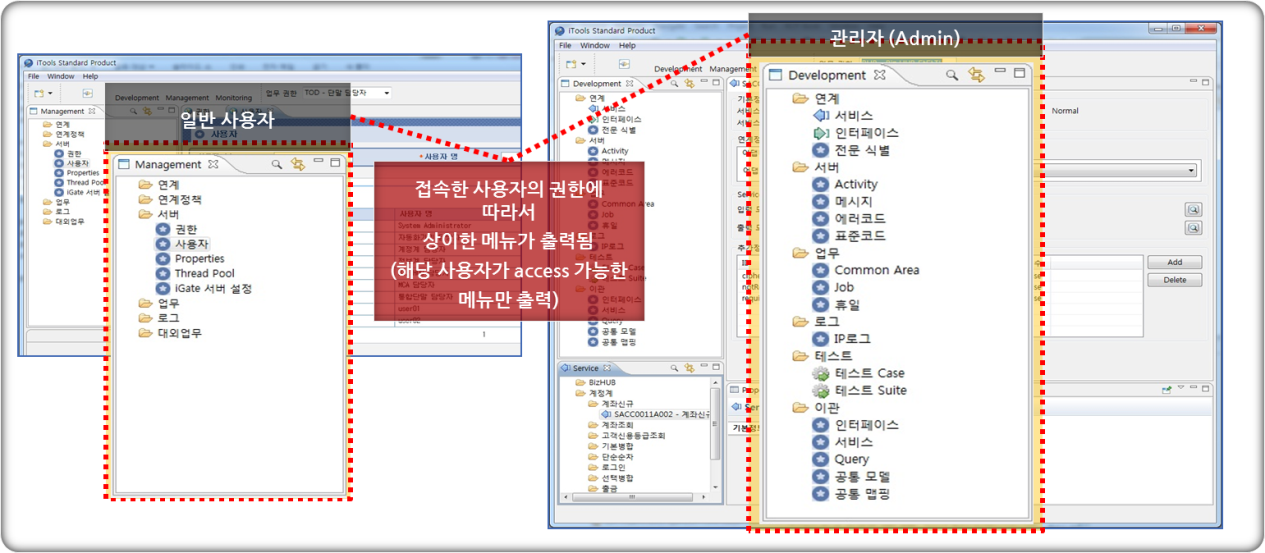
업무 Task 및 사용 Pattern에 따라 등록된 권한들을 그룹화하여 Role로 정의한다.

|  |  |
| --- | --- |
| **Role항목** | **설명** |
| **Role ID** | 권한들의 집합정보인 Role의 식별자. |
| **Role 설명** | Role에 대한 설명. |
| **권한리스트** | 해당 Role에 포함된 업무권한 및 시스템권한. |

### 사용자설정

iTools 및 iManager를 사용하는 사용자의 계정을 등록한다.

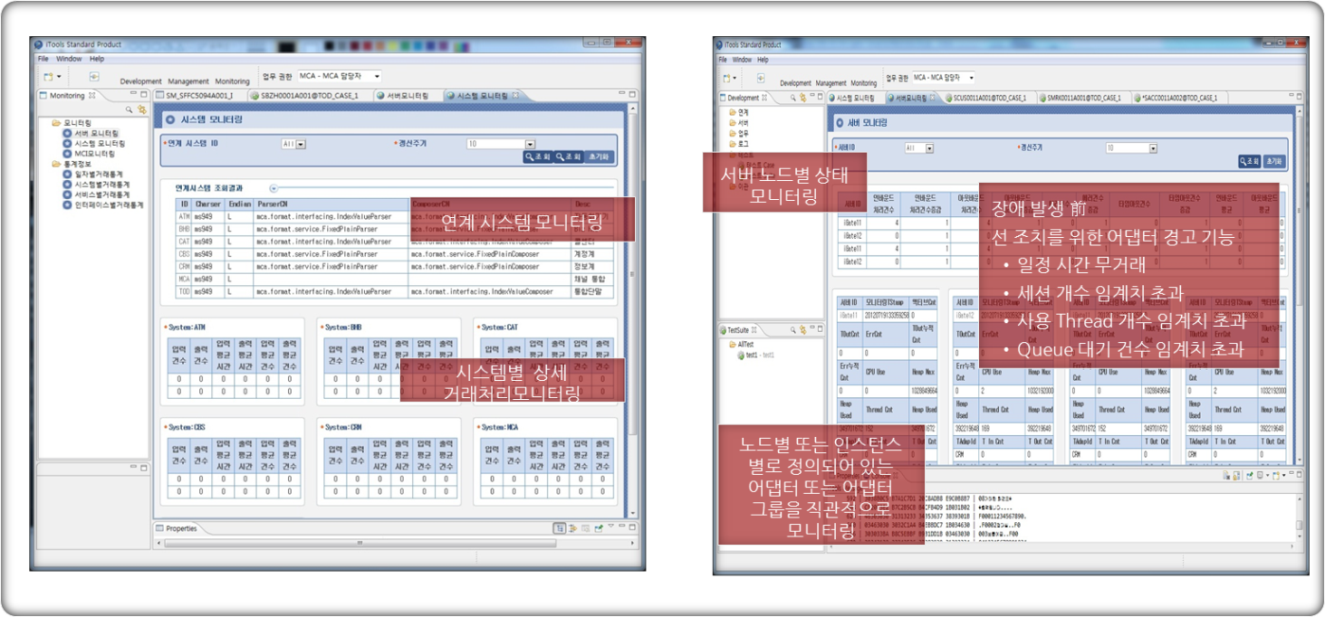
사용자 정보에는 사용자 기본정보, Role 정보 그리고 권한 정보를 설정한다.



**[그림 3‑１０]** 사용자권한에 따른 메뉴화면

## 시스템 모니터링

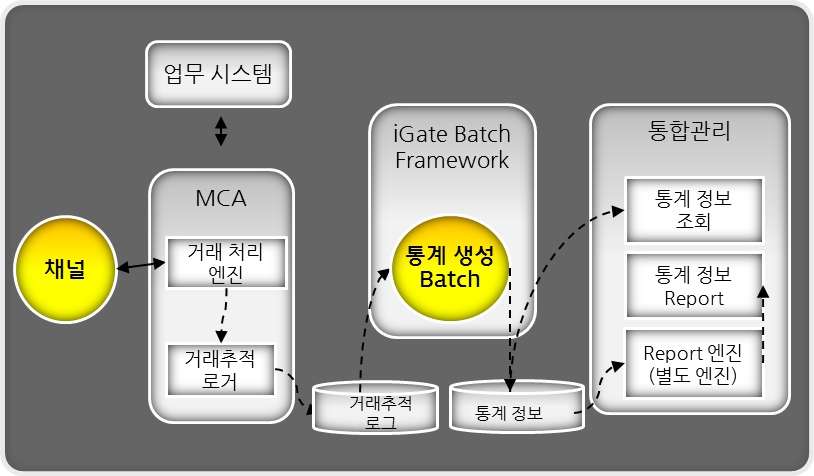
iGate는 효율적인 관리와 관리자의 편의성을 위하여 다양한 기능의 관리 및 모니터링, 원격제어 기능 등을 제공한다. iGate 시스템은 시스템 모니터링을 통하여 신속한 장애 대응 및 운영 관리가 가능하다.



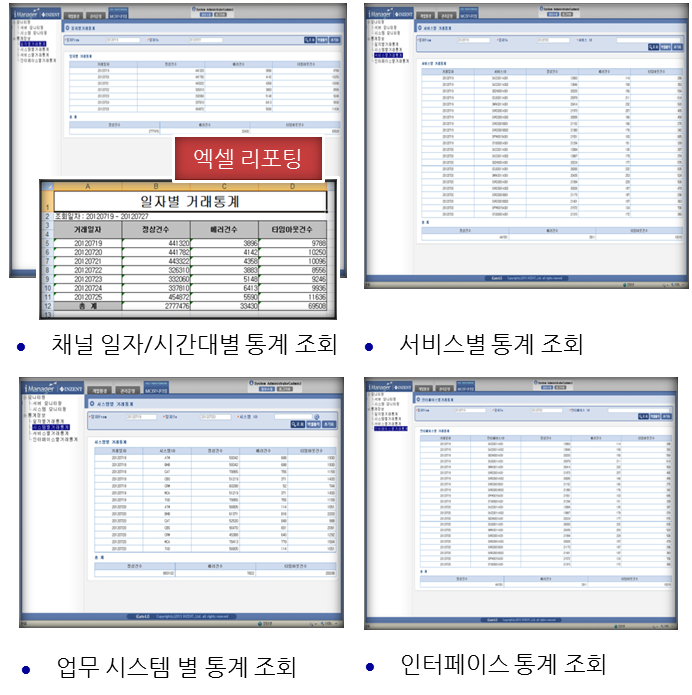
**[그림 3‑１１]** 시스템모니터링 및 서버모니터링

## 거래 통계

iGate는 일자별, 시스템별, 서비스별 그리고 인터페이스별 거래통계를 제공하여, 시스템 운영자가 원하는 통계자료를 조회 및 보고서(Excel)로 출력이 가능하다. 또한 거래 통계는 거래추적로그에서 통계생성 Batch에 의해 통계자료를 생성한다.



**[그림 3‑１２]** 거래 통계 흐름도



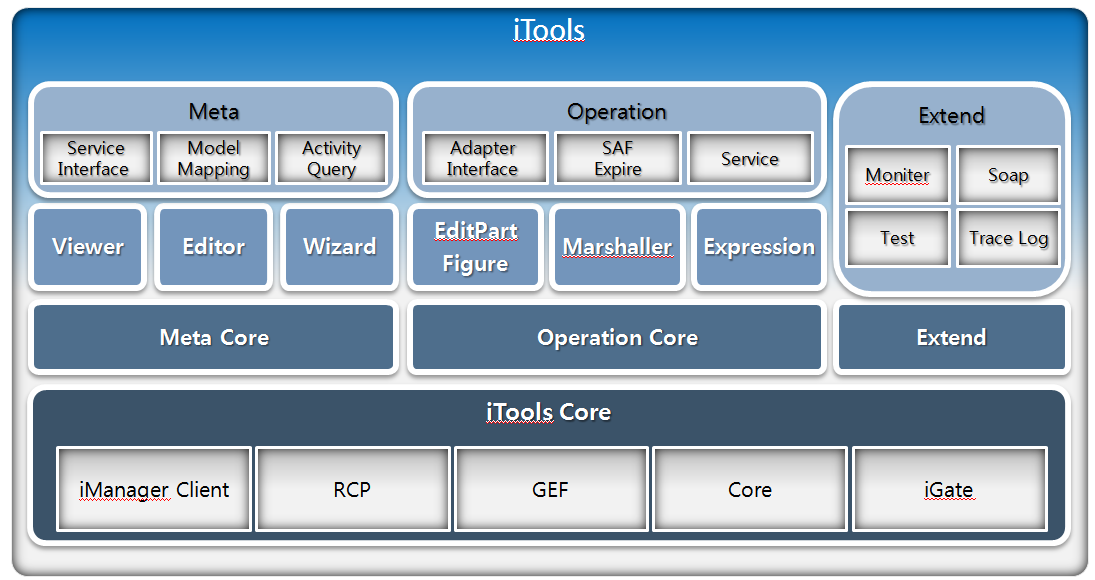
**[그림 3‑１３]** 거래통계 조회화면

# 개발 환경

본 장에서는 iGate의 개발 환경 아키텍처와 그 기능들을 기술한다.

## iTools Architecture

iTools는 Eclipse RCP 기반으로 구성되어 있으며 iMaager Client을 통해서 iManager와 연계한다. View, Editor, Wizard 로 Meta Data를 조회, 생성, 수정, 삭제를 수행합니다. Operation Core는 Eclipse GEF, Eclipse JDT 기반으로 구성되어 있고 Operation Core를 기반으로 다양한 Operation를 구성한다.



**[그림 4‑１]** iTools Architecture

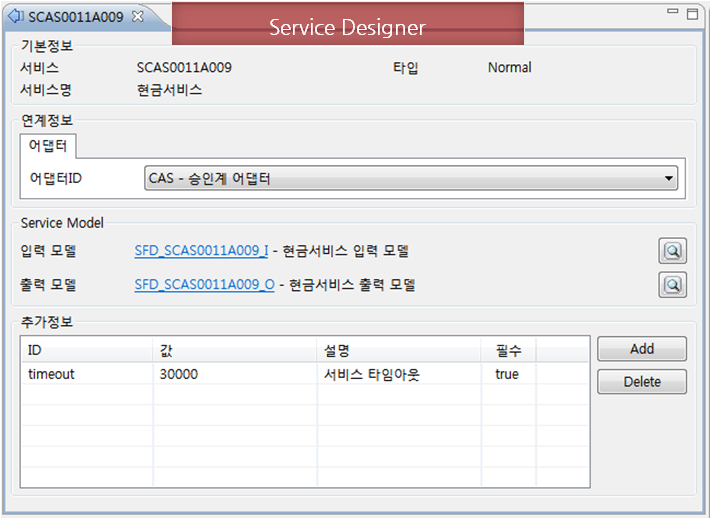
## Designer

### Service Designer

Service Designer는 Service Provider가 제공하는 단위 Service와 연계 하기 위한 정보를 정의 및 관리 하는 사용자 도구 이다.

Service Designer에서는 Service Provider와 연계하는 Connector 정보, Service ID, Request Data-Model, Response Data-Model, 기타 Service 속성을 정의한다.

또한 Virtual Service를 정의하여 Service를 제공 할 수 있게 한다.



**[그림 4‑２]** Service Designer

### Interface Designer

Interface Designer는 Service Consumer가 Service 사용 할 수 있도록 Interface를 정의 및 관리 하는 사용자 도구이다.

Interface Designer에서는 Service Consumer의 System ID, Interface ID, Request Data-Model, Response Data-Model과 Interface에서 사용하는 Service ID, 그리고 Service Model 과 Interface Model 사이 Request Mapping-Rule, Response Mapping-Rule, 기타 Interface 속성에 관련된 정보를 정의 한다.



**[그림 4‑３]** Interface Designer

### Data-Model Designer

Data-Model Designer는 Message의 구조를 정의한 Data-Model를 정의하고 관리 하는 사용자 도구 이다.

편리한 Data 구조 정의

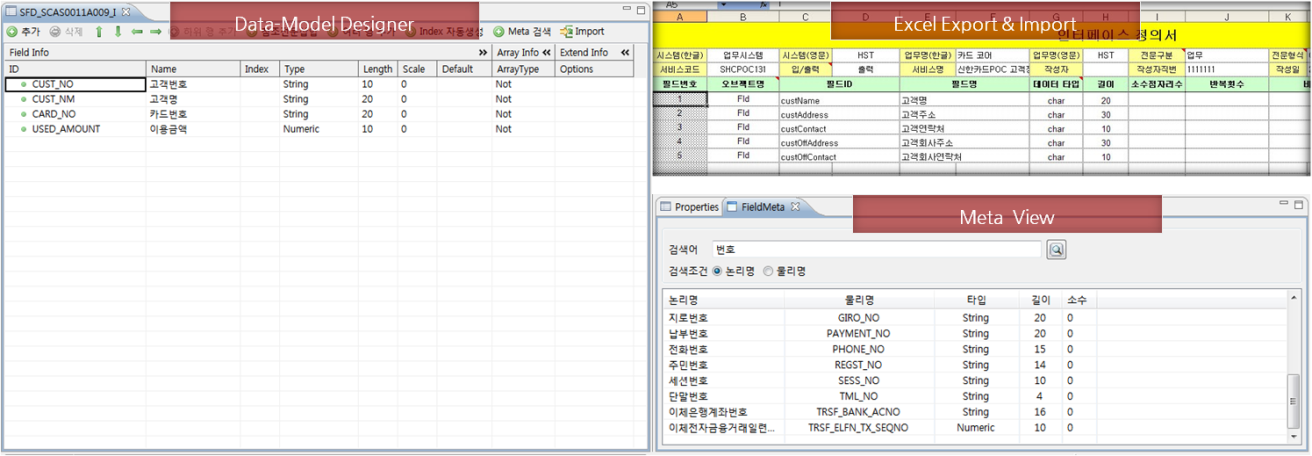
Data-Model의 계층 적인 구조를 Tree 형태로 표현 해서 Data-Model의 다양한 구조를 빠르게 정의 할 수 있고 Copy & Paste를 사용 하여 Data-Model Designer 사이에서 편리하게 작업 할 수 있다.

Meta 시스템 연동

Meta Viewer을 이용해서 Meta 시스템에서 정의된 논리명/물리명으로 검색 해서 Data-Model에 추가 할 수 있고, 논리명/물리명으로 입력된 정보를 바탕으로 Meta 시스템에서 정의한 정보를 검색해서 추가 정보를 자동으로 입력 할 수 있다.

Excel Export & Import

Data-Model을 Excel로 Export 해서 다양한 용도로 활용 할 수 있고 Excel로 정의한 Data-Model를 Import 할 수 있다.



**[그림 4‑４]** Data-Model Designer

### Mapping-Rule Designer

Mapping-Rule Designer는 Message 와 Message 사이의 관계를 정의하거나 변환에 관한 정보를 정의하고 관리 하는 사용자 도구 이다.

직관적인 UI

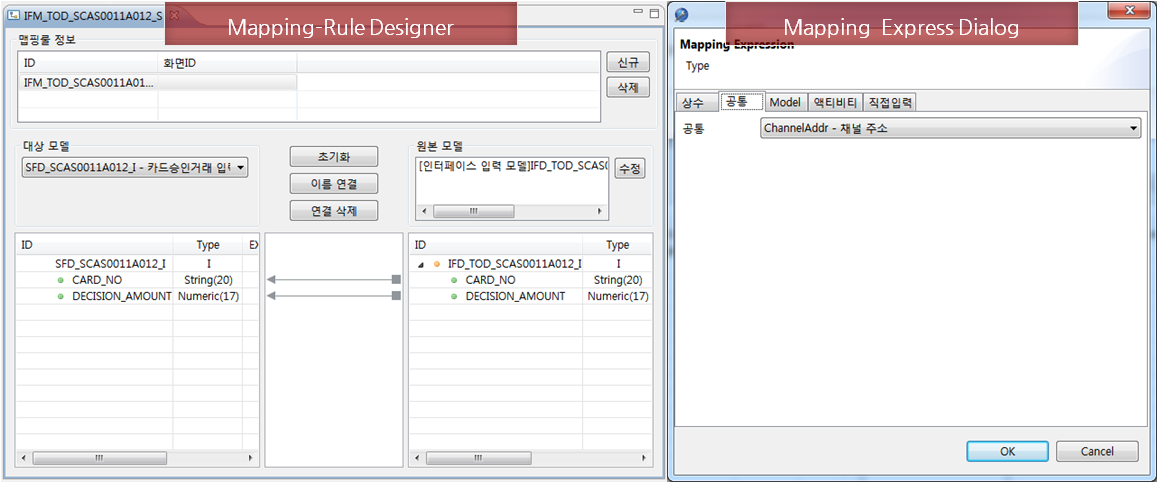
Data-Model 사이에 관계를 명확하게 이해 할 수 있도록 라인 형태로 표시한다. Data-Model 사이에 관계를 정의 하기 위해 소스 Data-Model 에서 대상 Data-Model로 Drag and Drop으로 쉽게 정의 할 수 있다. 또한 동일한 이름의 Data-Model의 경우 이름 연결 기능으로 빠르게 Data-Model 사이에 관계를 정의 할 수 있다.

다양한 소스로부터 맵핑

Data-Model로 관계를 정의 하기 위해 소스 Model 뿐만 아니라 상수, CArea, Data-Model 그리고 Mapping Activity를 사용 할 수 있다.

변환 지원

소스 Data-Model를 변환해서 하기 위해 Mapping Activity를 사용하여 변환한다. 편리한 Mapping Express Dialog를 제공해서 간단하게 Mapping Activity를 사용 할 수 있다. 다양한 Mapping Activity는 기본적으로 제공하고 추가적으로 Mapping Activity가 필요한 경우 Mapping Activity SDK 사용 해서 추가 할 수 있다.



**[그림 4‑５]** Mapping-Rule Designer

### Operation Designer

Operation Designer는 iGate 내부에서 처리 되는 일련의 프로세스를 정의하고 개발하는 도구이다.

직관적인 UI

Operation Designer는 Operation를 순서도 형식으로 표시하고, 주요 Component들을 icon으로 표현 하고 처리 Flow를 라인으로 포시 해서 전체적인 처리 Flow를 한눈에 알아보기 쉽게 구성하고 각각의 Component들을 Drag And Drop 으로 배치 할 수 있고 다양한 논리 Flow를 지원해서 쉽게 빠르게 Operation를 작성할 수 있다.

다양한 처리 Flow 지원

Operation Designer는 조건 처리, 반복 처리 그리고 Switch Case 처리를 지원 하고 직관적으로 표시 해서 다양한처리 흐름을 구성 할 수 있다.

|  |  |
| --- | --- |
| **종류** | **역할** |
| **IF** | **조건에** 따라 분기 한다. |
| SWITCH | 분기 블록을 처리 한다. |
| CASE | 분기 블록의 조건을 수행하고 참일 경우 처리 한다 |
| DEFAULT | 분기 블록의 Case 조건이 해당사항이 없을 경우 처리 한다. |
| WHILE | 반복 블록을 시작 한다. |
| BREAK | 반복 블록을 빠져 나간다. |
| CONTINUE | 반복 블록의 처음으로 돌아간다. |

분기 처리 지원

Operation Designer는 동시에 진행 되는 Component 처리를 하기 위해 Flow을 지원해서 다양하게 분기 및 병합을 할 수 있다.

|  |  |
| --- | --- |
| **종류** | **역할** |
| FLOW | 처리 분기하고 병합한다. |

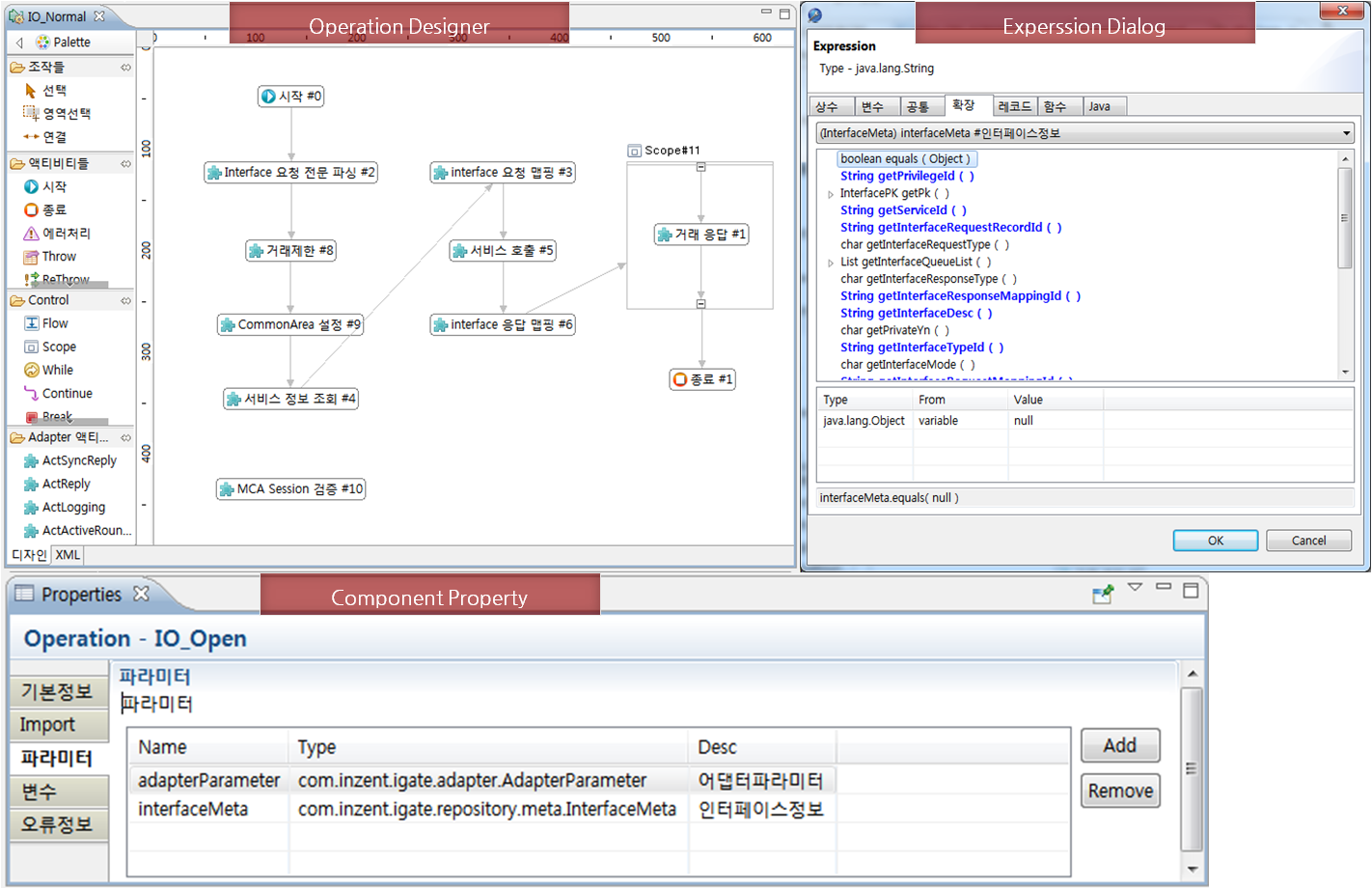
에러 처리 지원

Operation Designer는 Error Handler를 지원 해서 Operation내에 오류가 발생 했을 경우 오류 처리에 대해 적절하게 대처 할 수 있고 단위 블록에 대해서 Error Handler를 지원 할 수 있는 Scope 블록을 지원 한다. 또한 Throw 와 Rethrow를 통해 에러를 발생 시키거나 발생한 에러를 상위로 던질 수 있다.

|  |  |
| --- | --- |
| **종류** | **역할** |
| **Error Handler** | 에러 발생시 처리 한다. |
| **Throw** | 새로운 에러를 생성하고 발생한다. |
| **Rethrow** | 발생한 에러를 상위로 전달한다. |
| **Scope** | 에러 발생시 에러처리를 수행 할 수 있는 블록이다. |

편리한 Java Express 사용

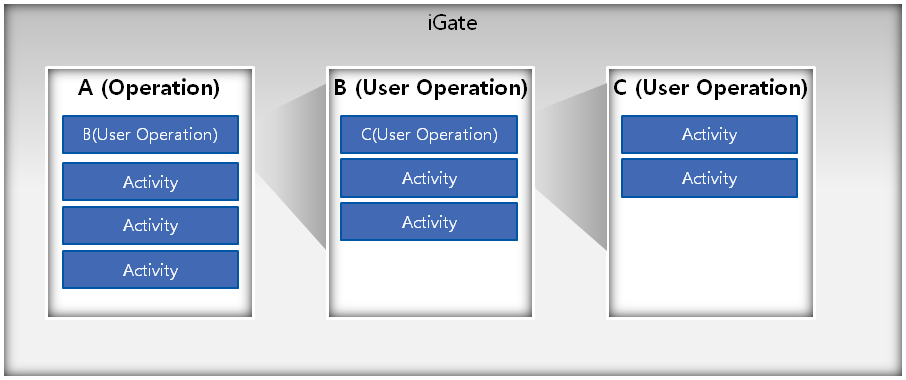
Operation Designer는 다양한 Java Express를 편리 하게 지원하기 위해 Express Dialog 지원 한다. Express Dialog는 상수, 변수, CArea, 확장, Data-Model, 함수, Java-Express 편리 하게 사용 할 수 있게 한다.



**[그림 4‑６]** Operation Designer

Operation

Operation은 User Operation, Activity로 구성되어 있고 각각의 Component 사이에 흐름제어, 반복 처리, 분기 및 병합, 오류 처리등과 같은 제어 요소로 이루어져 있다. User Operation과 Activity는 JAVA method 호출과 같이 parameter로 인터페이스를 정의 하고 있다. User Operation은 공통 기능을 수행 하기위해 단위 기능을 조합한 형태이고 User Operation은 다른 Operation에서 사용 할 수 있다. Activity는 단위 기능을 수행 하는 Component 이다.



**[그림 4‑７]** Operation Component

Operation의 종류 및 역할

|  |  |
| --- | --- |
| **종류** | **역할** |
| **Adapter Operation** | Adapter 처리 과정 중 Adapter Event에 따라 수행 하는 Operation |
| **Interface Operation** | Transaction 처리시 수행 되는 절차를 정의한 Operation |
| **SAF Operation** | SAF 처리시 수행 되는 절차를 정의한 Operation |
| **Expire Operation** | JMS 처리시 수행 되는 절차를 정의한 Operation |
| **User Operation** | 단위 기능을 처리 하기 위해 작성된 Component Operation  User Operation은 다른 Operation 에서 사용될 수 있다. |

## Tester

### TestCase

Test 지원

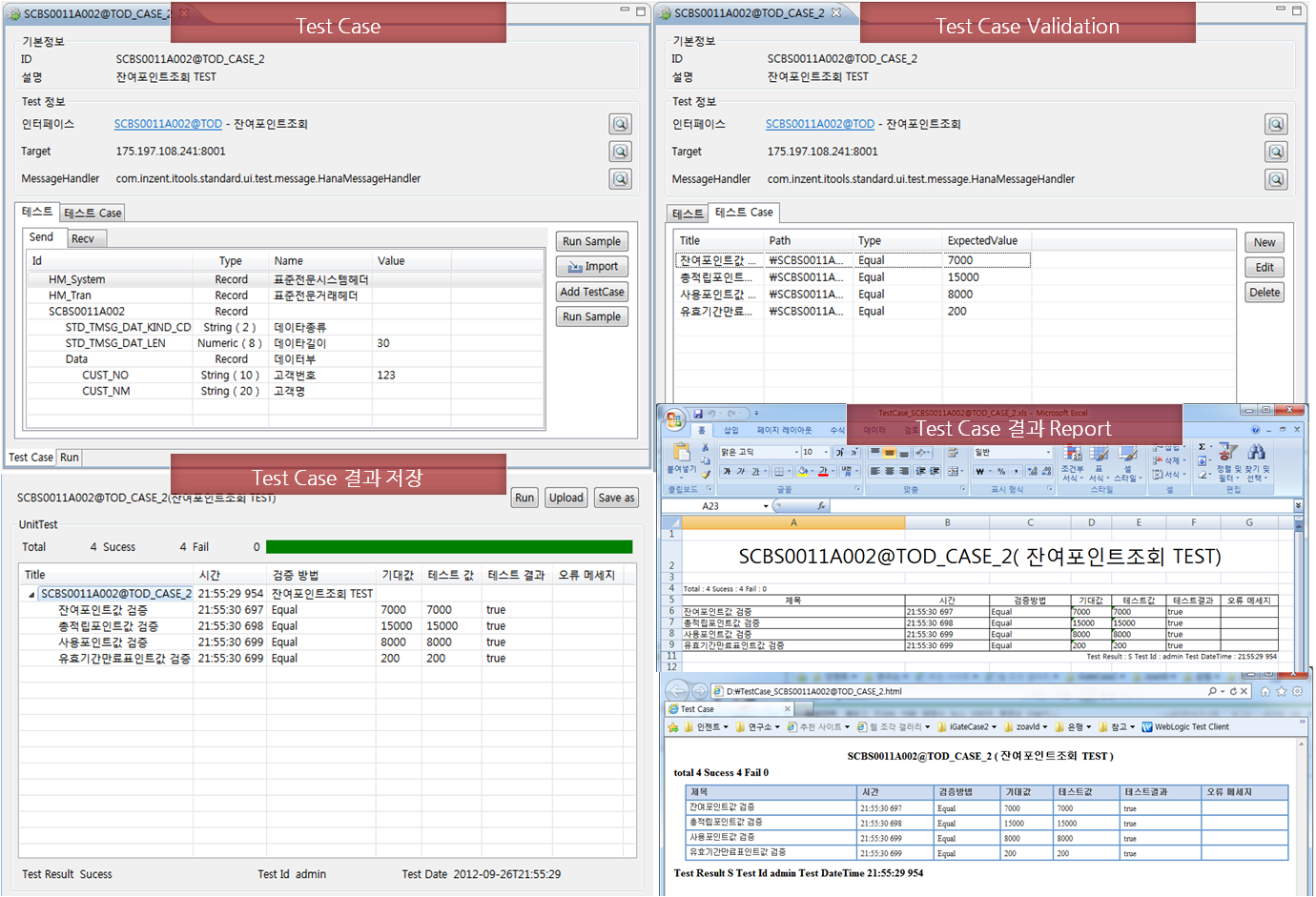
TestCase는 Interface을 손쉽게 Test 할 수 있도록 Request Model 구조로 표현되고 입력 값을 변경 할 수 있다. 또한 Response Message가 Model 구조로 표현되어 결과 확인을 쉽게 할 수 있다.

Test 결과 Validation

TestCase는 Test결과를 검증 하기 위한 여러 가지 검증 옵션들이 있고 이를 활용하여 Test 결과를 정확하게 식별 할 수 있습니다. 추출하고 결과 값을 여러 가지 Interface을 손쉽게 Test 할 수 있도록 Request Model 구조로 표현되고 입력 값을 변경 할 수 있다. 또한 Response Message가 Model 구조로 표현되어 결과 확인을 쉽게 할 수 있다.

Test 결과 저장 및 Report

TestCase는 사용자가 Test한 결과를 저장 할 수 있으면 저장된 결과를 서로 공유하여 Interface Test의 완성도를 올릴 수 있습니다. 또한 Test 결과를 Report 형태로 저장 하여 Test 결과서로 사용 할 수 있습니다.



**[그림 4‑８]** Test Case

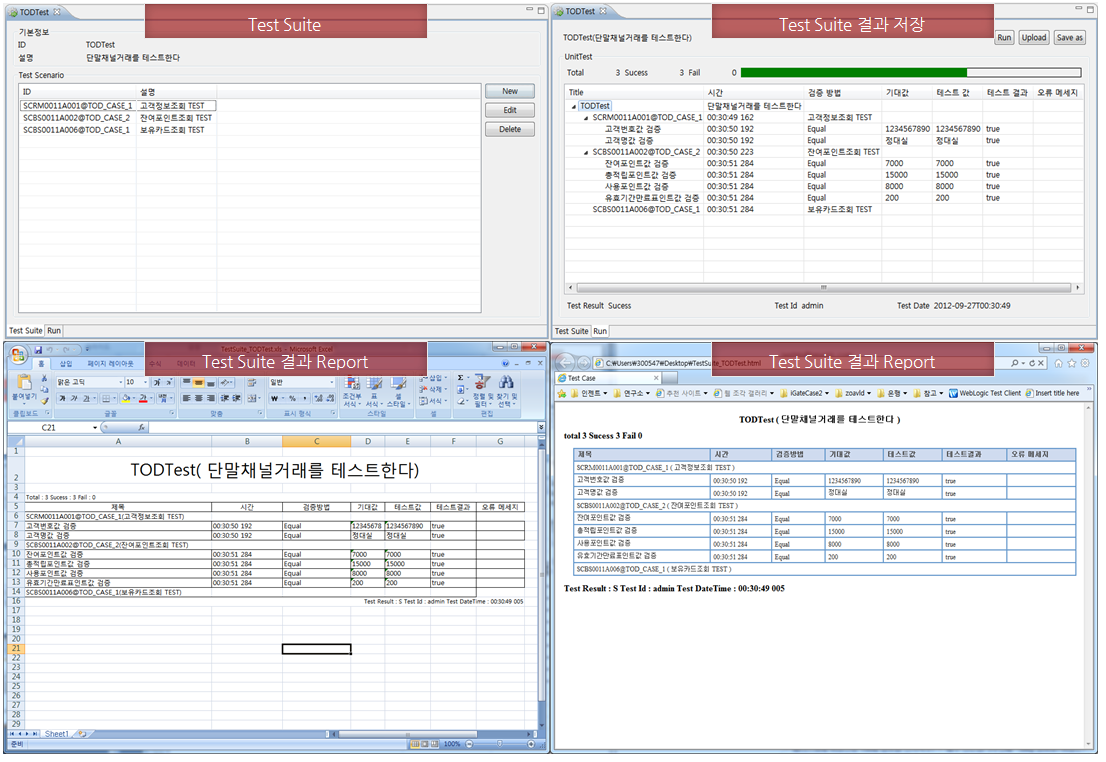
### TestSuit

Test Case 조합

TestSuite는 사용자들이 미리 만들어 놓은 여러 TestCase를 조합하여 각각 한번에 여러 TestCase를 Test 할 수 있다.

Test 결과 저장 및 Report

TestSuite는 TestCase에서 만들어 놓은 Test Validation로 TestCase 결과를 검증 할 수 있고 사용자가 Test한 결과를 저장 할 수 있으면 저장된 결과를 서로 공유하여 TestSuit의 Test 완성도를 올릴 수 있다. 또한 Test 결과를 Report 형태로 저장 하여 Test 결과서로 사용 할 수 있습니다.



**[그림 4‑９]** Test Suite

## 통합 상황판

통합 상황판

iGate에서 전체 상황을 한눈에 볼 수 있고 과거와 현재 상태를 명확히 알 수 있도록 차트도 표현 되어 있다. 주요 표시 내용 인스턴스 정보, 인스턴스별 메모리 정보, 인스턴스별 CPU 사용량, 시스템별 총응답 건수(누적),시스템별 응답 건수(시간), 실시간 거래 처리량/응답시간등 이 있습니다.

인스턴스 상황판

iGate 개별 인스턴스에 상황을 한눈에 볼 수 있고 과거와 현재 상태를 명확히 알 수 있도록 차트도 표현 되어 있다. 주요 표시 내용은 어댑터 정보,시스템별 응답 시간,시스템별 응답 건수,시스템별 처리 시간,시스템별 처리 건수,실시간 거래 처리량/응답시간등 이 있습니다.

장애 Alert

통합 상황판에서는 iGate에 오류가 발생 했을 때 오류정보를 보여주고 빠른 대응을 할 수 있도록 하고 각각 설정된 경고 상황이 발생 했을 때에도 발생한 경고 내용을 표시 한다.



**[그림 4‑１０]** 통합상황판

**용어 해설**

|  |  |
| --- | --- |
| **용어** | **해설** |
| **Activity** | iGate 내에 탑재되는 최소 단위의 업무 처리 로직.  Java로 작성되며, Normal/Service/Codec/Transform Activity 등으로 분류된다. |
| **DataModel** | iGate에 의해서 처리되는 여러 형태의 데이터 구조. |
| **Operation** | iGate 내에 탑재되는 여러 업무 처리 로직들이 작성되는 프로세스 룰.  GUI 기반의 Operation Designer에 의해서 작성된다. |
| **SOA** | Service-Oriented Architecture는 변화에 적시에 적절히 대응 할수 있는 IT 시스템을 구축할 수 있도록 하는 개념. 이를 위해 기업 내부 프로세스, 어플리케이션 들을 각각 ‘서비스’라는 기본적인 기능 단위로 나누고 이들 ‘서비스’를 연결하여 원하는 기능을 제공하도록 구성하고 비즈니스 환경이 변화 되었을 때 이 변화를 반영하여 서비스의 연결 구성을 쉽고 빠르게 변화시켜 기능을 제공하도록 하는 것 |
| **EBS** | Enterprise Service Bus로 서비스들을 컴포넌트화 된 논리적 집합으로 묶는 핵심 중간 도구이며, 비즈니스 프로세스 환경에 맞게 설계 및 전개할 수 있는 아키텍처 패턴, 기본적인 메시징, EAI 중개 기술을 사용하거나 J2EE시스템의 서비스 통합 버스와 같은 플랫폼 내의 구성요소를 사용하는 등 다양한 방법으로 구현가능 |
| **SAF** | Stored and Forward로 통신시스템에서 하나의 노드로 들어오는 패킷이나 메시지를 한 번 축척하고 나서 목적 부호로 상대국을 식별하여 그 정보를 전송하는 것 |
| **EDI** | Electronic Data Interchange로 전자문서교환방식이며 기업간 업무 처리에 있어서 종래의 종이로 된 문서나 서류를 교환하는 대신 컴퓨터로 처리 할 수 있는 구조화 되고 표준화된 양식  Interactive EDI, Internet EDI, XML EDI, Batch EDI, Open EDI가 있음 |
| **MCA** | Multi Channel Architecture로 은행의 각 영업점과 인터넷 뱅킹 콜센터 등에서 운영되는 다양한 채널을 통합 지원하는 시스템으로 단일화된 채널 지원을 통해 효과적인 고객 서비스를 제공하기 위한 목적으로 구축되는 것 |
| **Batch Program** | 일괄프로그램으로 모아서 한꺼번에 처리하는 프로그램 |
| **RMI** | Remote Method Invocation으로 분산시스템을 자바에서 지원해 주는 패키지로 네트워크로 연결된 다른 컴퓨터에 존재하는 메서드를 다른 컴퓨터에서 마치 내 컴퓨터에 있는 듯이 호출해서 사용하게끔 해주는 것이 java.rmi 패키지 |
| **Endian** | 단어를 형성하는 2진 바이트에서 저장하는 바이트의 순서를 나타내는 방법으로 big-endian, little-endian이 있으며 big-endian은 최상위 비트(MSB)부터 부호화되어 저장되며, little-endian은 최하위 비트(LSB)부터 부호화 되어 저장된다. 예를 들어 숫자 12가 2진수로 1100인데 big-endian은 1100, little-endian은 0011로 저장됨 |
| **Channel** | 은행 업무를 접하기 위한 고객의 접점의 매체, 업무시스템의 증가 만큼 채널도 다양화 된다. |

**색인**

Activity, 18

AnyFrame, 17

BANCS, 17

Composer, 17

DevOn-C, 17

DevOn-Java, 17

iGate@Link, 16

Mapper, 17

Operation, 18

Operation Designer, 38

Parser, 17

ProFrame, 16

RMI, 16

Service Designer, 35

Service Operation Designer, 40

SNA, 16

SYSTEMiER, 17

TCP/IP, 16

TestSuit, 42

Tmax, 16

Tuxedo, 16



Technology delivers the happiness

서울시 영등포구 대림3동 779-10 콤텍빌딩2층  
**www.inzent.com** | Tel: 02) 787-3600 | Fax: 02)787-3699

