InnoRules Installation and Operation Guide

v8

2025. 03. 31

InnoRules

목차 (Contents)

1. Quick Start

이 장에서는 InnoRules를 설치하고 구성을 하는 방법을 간략히 살펴본다. 설치하려는 시스템 구성은 다음과 같다.

- ❒ 룰 시스템은 개발과 운영의 두 개의 단위 룰 시스템으로 구성된다.
- ❒ 룰 빌더 서비스를 활성화한다.

1.1 설치 전 필요사항

InnoRules Quick Start를 수행하기 위해서는 아래의 사항들이 준비되어 있어야 한다.

- ❒ InnoRules용 OS 계정
  OS에 로그인하여 설치를 수행하고 InnoRules 서버를 실행하기 위한 계정이다.
- ❒ JDK 1.8 이상
- ❒ 최소 설치 공간 100MB
  InnoRules가 설치될 파일 시스템 상의 필요 공간으로 위의 OS 계정에 쓰기 권한이 부여되어 있어야 한다.
- ❒ InnoRules 설치 마법사: innorules-inst.jar
  InnoRules를 설치하는 실행 가능한 jar 파일이다.
- ❒ InnoRules 라이선스 파일
- ❒ 데이터베이스 서버 및 계정
  데이터베이스는 각 단위 룰 시스템의 룰 데이터를 저장하기 위해 사용된다. 각 단위 시스템마다 별도의 데이터베이스 계정이 필요하다. 본 예제에서는 개발, 운영 시스템을 구성할 것이므로 두 개의 데이터베이스 계정이 필요하다. 데이터베이스 계정은 다음의 권한을 가져야 한다.
  - ○ 데이터베이스에 접속
  - ○ 테이블 생성
  - ○ 데이터 생성, 삭제 및 조회
- ❒ JDBC 드라이버
  설정 마법사 및 InnoRules 서버가 위의 데이터베이스에 접근하기 위해 필요하다.
- ❒ InnoRules Web Builder
  설치 마법사에 포함되어 있으며, 룰 저작 클라이언트 도구로서 룰 빌더 서비스를 호출한다.

1.2 설치 마법사 실행

InnoRules 설치 마법사는 터미널 기반의 애플리케이션이다. 설치를 위해서는 터미널에 로그인을 해야 한다. 본 예제에서는 UNIX 터미널에서의 실행을 예로 들지만, Windows 환경에서도 실행 방법은 동일하다. 설치 마법사를 실행하기 전에 Figure 1-2-1과 같이 설치할 호스트에 설치 마법사 파일과 라이선스 파일을 위치시켜야 한다.

```
Innorules:~/install$ ls
innorules-inst.jar    innorules.lic
Innorules:~/install$
```

Figure 1 - 2 - 1

InnoRules 설치 마법사를 Figure 1-2-2와 같이 실행한다. PATH에 java의 경로가 등록되어 있지 않다면 java의 전체 경로를 입력해야 한다.

```
Innorules:~/install$ java -jar innorules-inst.jar
```

Figure 1 - 2 - 2

설치 마법사를 실행하면 Figure 1-2-3과 같이 소개 화면이 출력된다. 설치하고자 하는 InnoRules의 버전을 확인하고 사전 요구 사항을 확인한다. 본 예제에서는 InnoRules 서버를 이용하여 룰 빌더 서비스를 활성화할 것이기 때문에 시스템 관리자와 협의하여 HTTP 서비스 포트와 제어 포트를 할당 받아야 한다. 준비가 완료되었으면 엔터를 눌러 다음으로 진행한다.

```
*******************************************************************************

소개

*******************************************************************************

InnoRules 8.0.0를 설치하시게 된 것을 환영합니다. 설치 이전에 다음 사전 요구 사항을 확인하십시오.
        - JDK 1.8(또는 그 이상)
        - InnoRules 설치 디렉터리. 적어도 100MB 이상의 여유 공간이 필요합니다.
        - (선택 사항)HTTP 서비스 포트와 제어 포트가 할당되어 있어야 합니다.
[엔터를 누르십시오]
```

Figure 1 - 2 - 3

설치 마법사 및 InnoRules 서버가 사용할 JDK의 위치를 Figure 1-2-4와 같이 지정한다. 설치 마법사는 JDK의 위치를 추정할 수 있다면 그것을 기본값으로 보여준다. 기본값으로 보인 JDK를 그대로 사용하고자 한다면 엔터를 누르고 다른 JDK를 사용하고자 하다면 직접 경로를 입력 후 엔터를 누른다.

```
*******************************************************************************

JDK 홈 디렉터리

*******************************************************************************
```

```
JDK가 설치된 곳은 어디입니까? JDK 버전은 1.8 이상이어야 합니다.(기본값: /opt/java/java-8)

 >>
```

Figure 1 - 2 - 4

InnoRules를 설치할 디렉터리를 Figure 1-2-5와 같이 지정한다. 기본값으로 사용자 홈 디렉터리의 innorules-home이 지정된다. 변경하려면 설치를 원하는 디렉터리의 전체 경로를 입력한다. 단, 마법사를 실행하고 있는 시스템 계정으로 설치 디렉터리에 쓰기 작업을 할 수 있어야 한다.

```
********************************************************************************

설치 디렉터리

********************************************************************************

InnoRules를 어느 디렉터리에 설치하시겠습니까?(기본값: /home/innorules/innorules-home)

 >>
```

Figure 1 - 2 - 5

InnoRules에서 제공하는 서비스들에서 출력 될 로그 파일의 위치를 아래의 Figure 1-2-6과 같이 지정한다. 기본값은 InnoRules 서버의 기본 로그 디렉터리이다. 변경하려면 로그 파일이 생성될 디렉터리의 전체 경로를 입력한다. 단, InnoRules 서버가 로그 디렉터리에 쓰기 작업을 할 수 있어야 한다. InnoRules 서비스에 관련하여서는 3.8 InnoRules 서비스를 참조하라.

```
********************************************************************************

로그 디렉터리

********************************************************************************

어디에 로그 파일을 생성하시겠습니까? InnoRules 서비스들과 룰 런타임들이 하위 디렉터리에 로그를 기록할 것입니다.(기본값: /home/innorules/innorules-home/innorules-server/logs)

 >>
```

Figure 1 - 2 - 6

InnoRules 서버에서 사용할 서비스 포트와 제어 포트를 Figure 1-2-7과 같이 지정한다. 이 포트들은 InnoRules 서버가 사용하게 되므로, 서버를 실행하지 않을 경우 임의의 값을 입력해도 무방하다. 그러나, 빌더 서비스와 같은 InnoRules 서비스를 사용하려 한다면 시스템 관리자와 협의하여 서비스 포트

와 제어 포트를 부여받도록 한다.

```
********************************************************************************

InnoRules 서버 설정

********************************************************************************

InnoRules 서버는 관리 서비스, 룰 빌더 서비스 등과 같은 서비스를 제공할 수 있는 독립 서버입니다. 서버는 이 설치 노드에 실행될 수도 있고 아닐 수도 있습니다만, 지금 포트를 설정해 놓는 것이 좋습니다. 제어 포트는 InnoRules 서버를 중지시키기 위해 사용됩니다.

포트 번호를 입력하십시오.

[필수] HTTP 서비스 포트(기본값: 25802) >>
```

Figure 1 - 2 - 7

InnoRules 서버가 사용할 포트를 Figure 1-2-8와 같이 지정한다. InnoRules 서버는 기본적으로 두 개의 포트를 사용하는데, 하나는 HTTP 서비스를 위한 포트이며 나머지 하나는 서버를 제어하기 위한 포트이다. 서비스 포트와 제어 포트를 입력하고 엔터를 눌러 다음으로 진행한다.

```
...
포트 번호를 입력하십시오.

[필수] HTTP 서비스 포트(기본값: 25802) >>
[필수] 제어 포트(기본값: 25805) >>
```

Figure 1 - 2 - 8

InnoRules의 라이선스 파일을 Figure 1-2-9와 같이 지정한다. 설치 마법사는 작업 디렉터리에서 가용한 라이선스 파일들을 검색하여 보여준다. 목록에서 사용할 라이선스 파일을 선택할 수도 있고 설치가 완료된 이후에 수작업으로 라이선스 파일을 복사할 수도 있다.

```
********************************************************************************

라이선스 파일

********************************************************************************

현재 작업 디렉터리에서 다음의 라이선스 파일을 찾았습니다. 어떤 라이선스 파일을 설치하시겠습니까?
```

```
(*) 1. innorules.lic
        InnoRules(Expires: 2025-12-31)
    2. 설치 이후 수작업으로 복사



선택하십시오. >>
```

Figure 1 - 2 - 9

만약 설치 마법사의 작업 디렉터리에 라이선스 파일이 없는 경우 라이선스 파일 목록 대신에 Figure 1-2-10과 같은 메시지가 출력된다. 이 경우 설치를 하는 데에는 지장이 없지만, 설치 완료 후에 라이선스 파일을 수작업으로 복사해야만 InnoRules를 사용할 수 있다.

```
********************************************************************************

라이선스 파일

********************************************************************************

현재 작업 디렉터리에서 라이선스 파일을 찾지 못했습니다. 설치가 완료된 이후 수작업으로 라이선스 파일을 복사
해야 합니다.
[엔터를 누르십시오]
```

Figure 1 - 2 - 10

다음은 InnoRules 설치 마법사의 설치 전 최종 확인 단계이다. 지금까지 설정한 사항들의 요약과 설치 여부를 아래의 Figure 1-2-11과 같이 확인한다. Yes 또는 Y를 입력하고 엔터를 눌러 다음으로 진행한다.

```
********************************************************************************

요약 및 설치

JDK Home          : /opt/java/java-8
Install Directory    : /home/innorules/innorules-home
Service Port         : 25802
Admin Port           : 25805
License File         : /home/innorules/install/innorules.lic

이 설정으로 InnoRules를 설치합니다. 괜찮겠습니까? (Yes/No) >> Yes
```

Figure 1 - 2 - 11

위와 같은 설정으로 정상적으로 설치가 완료되었다면 아래의 Figure 1-2-12와 같이 설치 성공 메시지가 출력된다. 엔터를 눌러 설치를 종료한다.

```
[정보] Installation succeeded.[엔터를 누르십시오]
```

Figure 1 - 2 - 12

InnoRules 시스템 설치가 완료되면 설치 마법사에서 지정한 InnoRules 설치 디렉터리 하위에 아래의 Figure 1-2-13과 같이 폴더가 생성된 것을 확인 할 수 있다.

```
innorules:~/innorules-home$ ls
config  innorules-server  lib  license  manual-config  script
```

Figure 1 - 2 - 13

1.3 룰 시스템 구성 마법사 실행

설치가 완료되었으면 단위 룰 시스템을 구성해야 한다. 단위 룰 시스템 구성을 위해서는 룰이 저장될 데이터베이스에 접근할 수 있어야 한다. 데이터베이스 접근을 위해 사전에 준비한 JDBC 드라이버를 설치 디렉터리의 lib/jdbc-drivers에 복사를 한다. Figure 1-3-1은 오라클 데이터베이스의 JDBC 드라이버인 ojdbc8.jar를 lib/jdbc-drivers로 복사하는 예시이다.

```
innorules@:~/innorules-home/lib/jdbc-drivers$ ls
ojdbc8.jar
innorules@:~/innorules-home/lib/jdbc-drivers$
```

Figure 1 - 3 - 1

Figure 1-3-2와 같이 룰 시스템 구성 마법사를 실행한다. Windows에서는 system-wiz.bat을 실행한다.

```
innorules@:~/innorules-home/script$ ./system-wiz.sh
```

Figure 1 - 3 - 2

Figure 1-3-3은 룰 시스템 구성 마법사의 초기 화면이다.

```
*******************************************************************************

소개

*******************************************************************************

룰 시스템 구성 마법사에 오신 것을 환영합니다. 이 마법사는 룰 시스템을 구성하도록 도와줍니다.
```

```
[엔터를 누르십시오]
```

Figure 1 - 3 - 3

Figure 1-3-4과 같이 룰 시스템 마법사를 이용하여 단위 룰 시스템에 대한 설정을 하거나 룰 시스템에서 접근할 업무 데이터베이스에 대해 설정할 수 있다. 본 예제에서는 개발과 운영의 두 개의 단위 룰 시스템을 설정할 것이다.

```
*********************************************************************************************

설정 유형 선택

*********************************************************************************************

설정 유형을 선택하시오.
    1. 단위 룰 시스템 관리
        단위 룰 시스템 설정을 관리합니다.
    2. 업무 데이터베이스 관리
        업무 데이터베이스 설정을 관리합니다.


선택하십시오. >> 1
```

Figure 1 - 3 - 4

단위 룰 시스템 관리를 선택하면 등록된 단위 룰 시스템의 목록이 Figure 1-3-5와 같이 출력된다. 등록된 단위 룰 시스템이 없으므로 목록은 비어 있다. 'a'를 입력하여 추가를 선택한다.

```
*********************************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리

*********************************************************************************************

등록된 단위 룰 시스템의 목록입니다. 새로운 단위 룰 시스템을 추가하거나 기존의 것을 수정할 수 있습니다.

[항목들]
----------------------
    | ID | 이름 |
----------------------
```

```
[가능한 작업들]
a. 추가

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >> a
```

Figure 1 - 3 - 5

추가하고자 하는 단위 룰 시스템의 ID와 이름을 Figure 1-3-6와 같이 입력한다. ID는 10자 이내의 식별자로 영문자 대소문자, 숫자 또는 '_'로만 이루어져야 한다. 예제에서는 개발 단계의 의미로 'DEV'를 입력한다. 이름은 임의의 문자를 사용 가능하지만 60자 이내이어야 한다.

```
********************************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보

********************************************************************************************

단위 룰 시스템의 기본 정보를 입력하십시오.

[필수] ID >> DEV
[필수] 이름 >> Development
```

Figure 1 - 3 - 6

단위 룰 시스템의 기본 정보가 입력되었으면 이 단위 룰 시스템의 룰 데이터가 저장될 DB에 대한 설정을 해야 한다. Figure 1-3-7과 같이 룰 DB의 이름과 설명을 입력한다.

```
********************************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보 -> 룰 DB 기본 정보

********************************************************************************************

룰 DB의 기본 정보를 입력하십시오. 정보는 한 번 지정하면 변경할 수 없습니다.

[필수] 이름 >> RULEDB
[필수] 설명(기본값: ) >>
```

Figure 1 - 3 - 7

다음 단계는 데이터베이스의 유형을 선택하는 단계이다. 몇몇 데이터베이스들의 목록이 미리 마법사에 등록이 되어 있다. 사용하고자 하는 데이터베이스가 이 목록에 없는 경우는 기술지원을 요청하라.
본 예제에서는 Figure 1-3-8과 같이 Oracle을 선택하였다.

```
********************************************************************************
설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보 -> 룰 DB 서버 유형

********************************************************************************

룰 DB로 사용할 데이터베이스 서버의 유형이 무엇입니까?

    1. Oracle
        Oracle database (with type 4 driver)
    2. MSSQL
        Microsoft SQL server
    3. DB2
        IBM DB2 with DB2 UDB JDBC universal driver
    4. MySQL
        MySQL database server
    5. Postgre
        Postgre database server


선택하십시오. >> 1
```

Figure 1 - 3 - 8

데이터베이스 서버를 선택하면 Figure 1-3-9와 같이 서버에 접속하기 위한 부가 정보를 입력하는 화면이 나타난다. 입력해야 할 정보는 선택한 데이터베이스에 따라 다를 수 있다.

```
********************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보 -> 룰 DB 접속 정보

********************************************************************************

룰 DB RULEDB에 접속하기 위한 정보를 입력하십시오.

[필수] address >> your_database_ip
[필수] port(기본값: 1521) >> your_database_port
[필수] SID or service name(기본값: ORCL) >> your_database_sid
[필수] 사용자 계정 >> your_database_account
[필수] 비밀번호 >> your_database_password
```

Figure 1 - 3 - 9

모든 값을 입력하면 Figure 1-3-10과 같이 룰 DB 등록 확인 메시지가 출력된다. 모든 값들이 올바른지 확인을 하고 Yes 또는 Y를 입력한다.

```
이 정보를 이용하여 룰 DB를 등록하시겠습니까? 등록을 할 경우 이 접속 정보로 DB에 로그인하여 테이블을 생성하거나 데이터를 수정하게 될 수도 있습니다. (Yes/No) >> Yes
```

Figure 1 - 3 - 10

Figure 1-3-10에서 Yes를 입력하면 마법사는 Figure 1-3-9의 정보를 이용하여 데이터베이스 서버에 접속을 하여 일련의 테이블들을 생성한다. 만약 이미 테이블들이 생성이 되어 있는 경우 테이블 및 데이터의 정합성을 체크한다.

현재 단위 룰 시스템이 개발 시스템이고, 룰 DB를 최초로 등록한다면 Figure 1-3-11와 같이 기본 날짜 형식을 입력한다. 기본 날짜 형식은 날짜 형식을 사용하는 룰 문법에서 사용되며, 기본값은 yyyy-MM-dd이다. 이 형식은 Java의 SimpleDateFormat형식을 준수한다. 여기서는 엔터를 눌러 다음 단계로 진행한다.

```
********************************************************************************

룰 시스템에서 사용할 기본 날짜 형식을 입력하십시오. 기본 날짜 형식은 날짜 형식을 사용하는 룰 문법에서 사용됩니다. 이 형식은 Java의 SimpleDateFormat형식을 준수하여야 합니다.(기본값: yyyy-MM-dd)

 >>
```

Figure 1 - 3 - 11

정상적으로 룰 DB가 등록되었다면 룰 DB 관리 화면이 나타나고 단위 룰 시스템 DEV에 등록된 룰 DB의 목록이 Figure 1-3-12와 같이 보인다. 작업을 선택하지 않은 상태에서 엔터를 눌러 다음 단계로 진행한다.

```
********************************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보 -> 룰 DB 관리

********************************************************************************************

단위 룰 시스템 'DEV'에 등록된 룰 DB의 목록입니다. 새로운 룰 DB를 등록하거나 기존의 룰 DB 정보를 수정할 수 있습니다.

[항목들]
```

```
------------------------------------------------------------------
    | 이름     | JDBC URL                                  | 사용자 계정 | 설명 |
------------------------------------------------------------------
  1 | RULEDB | jdbc:oracle:thin:@ip:port:SID       | your account |
------------------------------------------------------------------

[가능한 작업들]
a. 추가     b. 정보변경     c. 위로     d. 아래로

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 1 - 3 - 12

개발 단위 룰 시스템의 등록이 완료되었으며 단위 룰 시스템 관리 화면에 Figure 1-3-13와 같이 DEV 시스템이 등록된 것이 보여진다.

```
*********************************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리

*********************************************************************************************

등록된 단위 룰 시스템의 목록입니다. 새로운 단위 룰 시스템을 추가하거나 기존의 것을 수정할 수 있습니다.

[항목들]
----------------------------------------
    | ID         | 이름          |
----------------------------------------
  1 | DEV       | Development|
----------------------------------------

[가능한 작업들]
a. 추가     b. 수정 및 룰 DB 관리     c. 마지막 단위 룰 시스템 삭제

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 1 - 3 - 13

DEV 단위 룰 시스템을 등록한 것과 동일한 방법으로 운영 단위 룰 시스템을 추가한다. Figure 1-3-14은 PROD라는 ID로 단위 룰 시스템을 추가한 것으로 보여준다.

```
*********************************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리

*********************************************************************************************

등록된 단위 룰 시스템의 목록입니다. 새로운 단위 룰 시스템을 추가하거나 기존의 것을 수정할 수 있습니다.

[항목들]
-----------------------------------------
    | ID         | 이름          |
-----------------------------------------
  1 | DEV        | Development|
-----------------------------------------
  2 | PROD    | Production    |
-----------------------------------------

[가능한 작업들]
a. 추가     b. 수정 및 룰 DB 관리     c. 마지막 단위 룰 시스템 삭제

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 1 - 3 - 14

1.4 서비스 구성 마법사 실행

여기서는 서비스 구성 마법사를 실행하여 룰 빌더 서비스를 구성한다. 서비스 구성 마법사에 대한 자세한 사항은 2.4 서비스 구성 마법사를 참조하라.

Figure 1-4-1과 같이 서비스 구성 마법사를 실행한다. Windows에서는 service-wiz.bat을 실행한다.

```
innorules@:~/innorules-home/script$ ./service-wiz.sh
```

Figure 1 - 4 - 1

서비스 구성 마법사를 실행하면 Figure 1-4-2과 같은 초기 화면이 출력된다. 엔터를 눌러 다음으로 진행한다.

```
*********************************************************************************************
```

```
소개

********************************************************************************

InnoRules 서비스 구성 마법사에 오신 것을 환영합니다. 이 마법사는 당신의 호스트에 다양한 서비스들을 설정하
도록 도와줄 것입니다.
[엔터를 누르십시오]
```

Figure 1 - 4 - 2

InnoRules에서 제공하는 서비스를 구성할 수 있는 화면이 아래의 Figure 1-4-3과 같이 출력된다. 룰 빌더 서비스를 구성하기 위해 '2'를 입력하고 엔터를 눌러 다음으로 진행한다.

```
********************************************************************************

서비스 유형 선택

********************************************************************************

설정하고자 하는 서비스의 유형을 선택하십시오. 종료하려면 엔터를 누르십시오.
    1. 관리 서비스
        동작 중인 룰 런타임 시스템과 룰 시스템 컴포넌트들을 관리할 수 있습니다.
    2. 룰 빌더 서비스
        룰을 작성하고 관리할 수 있습니다.
    3. TCP 커넥터 서비스
        원격 어플리케이션이 룰 서비스를 호출할 수 있습니다.
    4. 룰 REST 서비스
        룰 REST 서비스를 설정하고 웹 환경에서 룰을 호출할 수 있습니다.

선택하십시오. >> 2
```

Figure 1 - 4 - 3

빌더 서비스 구성 화면이 Figure 1-4-4과 같이 출력된다. 빌더 서비스를 활성화하기 위해서는 'd'를 입력하고 엔터를 눌러 다음으로 진행한다.

```
********************************************************************************

서비스 유형 선택 -> 빌더 서비스

********************************************************************************
```

```
빌더 서비스들에 대한 작업을 선택하십시오.

[전역 설정]
    이관 전용        : 설정되지 않음
    평균 동시 사용자: 2
    최대 동시 사용자: 10
    암호화 방식    : SHA-256
    테이블 룰 조건식 체크    : 설정되지 않음
    클러스터 접속 타임 아웃(ms)   : 5,000
    클러스터 응답 타임 아웃(ms)    : 60,000
    클러스터 문자셋    : UTF-8

[항목들]
--------------------------------------------------------------------------
    | 이름 | 주소 | 서비스 포트 | 모니터 포트 | 로컬 |
--------------------------------------------------------------------------

[가능한 작업들]
c. 전역 설정 변경     d. 로컬 설치 노드에 빌더 서비스 활성화

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >> d
```

Figure 1 - 4 - 4

모니터 서버 접속 정보 등록 화면이 Figure 1-4-5와 같이 출력된다. 빌더 서버의 기능인 모니터 서버가 사용할 주소와 포트를 선택한다. 모니터 포트는 시스템 담당자 또는 네트워크 담당자에 협의하여 할당 받아야 한다. 기본 값은 25810이다.

```
*********************************************************************************************

서비스 유형 선택 -> 빌더 서비스 -> 모니터 서버 접속 정보

*********************************************************************************************

모니터 서버의 접속 정보를 입력하십시오.

[필수] 접속 주소
    1. 100.10.0.10
    2. 127.0.0.1
```

```
>> 1
[필수] 포트(기본값: 25810) >>
```

Figure 1 - 4 - 5

빌더 서비스가 구성이 정상적으로 완료되었다면 Figure 1-4-6과 같이 빌더 서비스 목록에 추가된 서버가 표시된다. 작업을 선택하지 않고 엔터를 눌러 서비스 마법사를 종료한다.

```
********************************************************************************

서비스 유형 선택 -> 빌더 서비스

********************************************************************************

빌더 서비스들에 대한 작업을 선택하십시오.

[전역 설정]
    이관 전용        : 설정되지 않음
    평균 동시 사용자: 2
    최대 동시 사용자: 10
    암호화 방식    : SHA-256
    테이블 룰 조건식 체크    : 설정되지 않음
    클러스터 접속 타임 아웃(ms)   : 5,000
    클러스터 응답 타임 아웃(ms)    : 60,000
    클러스터 문자셋    : UTF-8

[항목들]
--------------------------------------------------------------------------------
    | 이름                                   | 주소           | 서비스 포트 | 모니터 포트 | 로컬 |
--------------------------------------------------------------------------------
  1 | 906946d6-8a78-4877-9930-f9d160254d46 | 100.10.0.10 | 25802      | 25810   | V     |
--------------------------------------------------------------------------------

[가능한 작업들]
a. 위로     b. 아래로     c. 전역 설정 변경     d. 로컬 설치 노드의 빌더 서비스 비활성화     e. 빌더 서비스 접속 URL 확인
작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 1 - 4 - 6

1.5 InnoRules 서버 구동

InnoRules 설치가 정상적으로 완료되었다면 InnoRules 서버를 기동한다. InnoRules 서버에 대한 자세한 사항은 3.8.1 InnoRules 서버를 참조하라.

InnoRules 서버는 설치 디렉터리 하위의 script에서 Figure 1-5-1과 같이 다음 명령어로 실행시킨다. Windows의 경우 start-server.bat을 실행한다.

```
innorules@:~/innorules-home/script$ ./start-server.sh
```

Figure 1 - 5 - 1

InnoRules 서버가 정상적으로 구동이 되면 Figure 1-5-2와 같이 출력된다.

```
Using CATALINA_BASE:   /home/innorules/innorules-home/innorules-server/default
Using CATALINA_HOME:   /home/innorules/innorules-home/innorules-server/apache-tomcat-9
Using CATALINA_TMPDIR: /home/innorules/innorules-home/innorules-server/default/temp
Using JRE_HOME: /opt/java/java-8
Using CLASSPATH:       :/home/innorules/innorules-home/lib/jdbc-drivers/*:/home/innorules/innorules-home/innorules-server/apache-tomcat-9/bin/bootstrap.jar:/home/innorules/innorules-home/innorules-server/apache-tomcat-9/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.
```

Figure 1 - 5 - 2

InnoRules 서버의 프로세스는 Figure 1-5-3과 같이 console에서 아래 명령어로 확인할 수 있다.

```
innorules@siyang-ubuntu-pc:~$ ps -aef | grep catalina
innorules 8442 1 8 18:02 pts/1 00:00:04 /opt/java/java-8/bin/java -Djava.util.logging.config.file=/home/innorules/innorules-home/innorules-server/default/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -Dinnorules.home=/home/innorules/innorules-home -Dinnorules.log=/home/innorules/innorules-home/innorules-server/logs -classpath :/home/innorules/innorules-home/lib/jdbc-drivers/*:/home/innorules/innorules-home/innorules-server/apache-tomcat-9/bin/bootstrap.jar:/home/innorules/innorules-home/innorules-server/apache-tomcat-9/bin/tomcat-juli.jar -Dcatalina.base=/home/innorules/innorules-home/innorules-server/default -Dcatalina.home=/home/innorules/innorules-home/innorules-server/apache-tomcat-9 -Djava.io.tmpdir=/home/innorules/innorules-home/innorules-server/default/temp
org.apache.catalina.startup.Bootstrap start
innorules@:~/innorules-home/script$
```

Figure 1 - 5 - 3

netstat 명령어를 이용하여 Figure 1-5-4와 같이 모니터 서버의 포트가 사용되고 있음을 확인할 수 있다.

```
innorules@:~/innorules-home/script$ netstat -a | grep 25810
tcp6       0      0 100.10.0.10%4163:25810 [::]:*                  LISTEN
tcp6       0      0 100.10.0.10%4163:25810 100.10.0.10%4168:56791 ESTABLISHED
tcp6       0      0 100.10.0.10%4163:25810 100.10.0.10%4168:56792 ESTABLISHED
tcp6       0      0 100.10.0.10%4163:56792 100.10.0.10%4168:25810 ESTABLISHED
tcp6       0      0 100.10.0.10%4163:56791 100.10.0.10%4168:25810 ESTABLISHED
innorules@siyang-ubuntu-pc:~$
```

Figure 1 - 5 - 4

InnoRules 서버의 로그는 Figure 1-5-5와 같이 다음 경로의 로그 파일에서 확인이 가능하다.

```
~/innorules-home/innorules-server/logs/system/innorules-server.log
```

Figure 1 - 5 - 5

1.6 룰 빌더 접속

InnoRules Web Builder를 이용하여 InnoRules 서버(룰 빌더 서버)를 접속한다. InnoRules Web Builder는 "InnoRules Web Builder Installation Guide"를 참고하여 설치한다. InnoRules Web Builder가 설치되면 웹 브라우저로 Figure 1-6-1과 같이 InnoRules 서버(룰 빌더 서버)에 접속한다. (Default URL: https://IP:PORT/innorules/web/index.html)

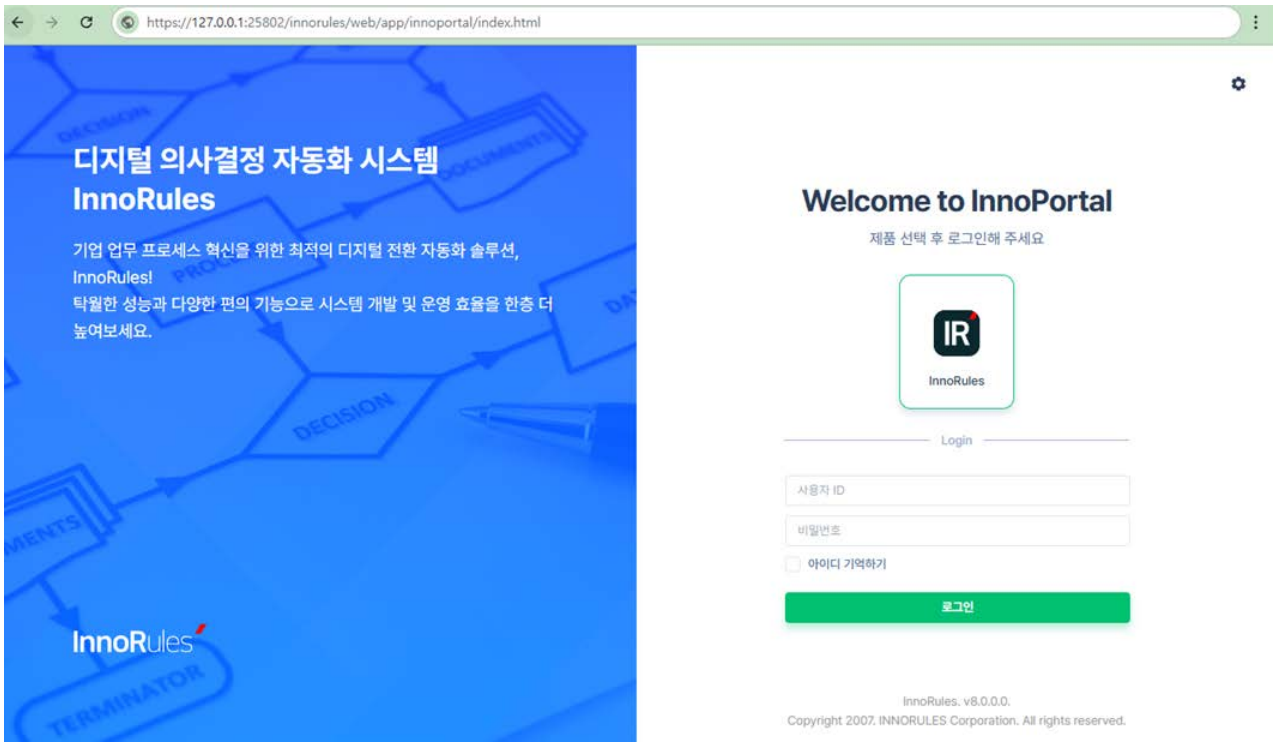

Figure 1 - 6 - 1

로그인 화면에서 Figure 1-6-2과 같이 'admin' 유저로 접속한다. 'admin' 유저의 초기 비밀번호는 'admin' 이다. InnoRules 서버가 최초로 기동 된 시점에는 'admin' 유저로만 접속이 가능하다.

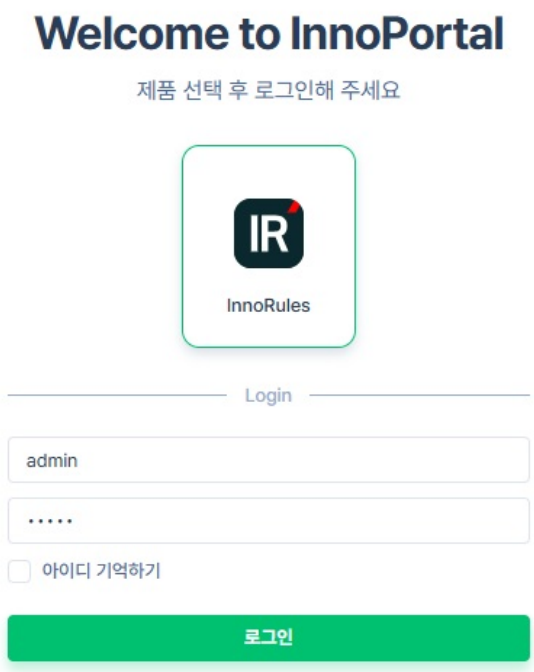

Figure 1 - 6 - 2

지금까지의 과정이 모두 정상적으로 진행되었다면 Figure 1-6-3과 같이 InnoRules 서버에 정상적으로 접속하여 메인 화면이 표시된다.

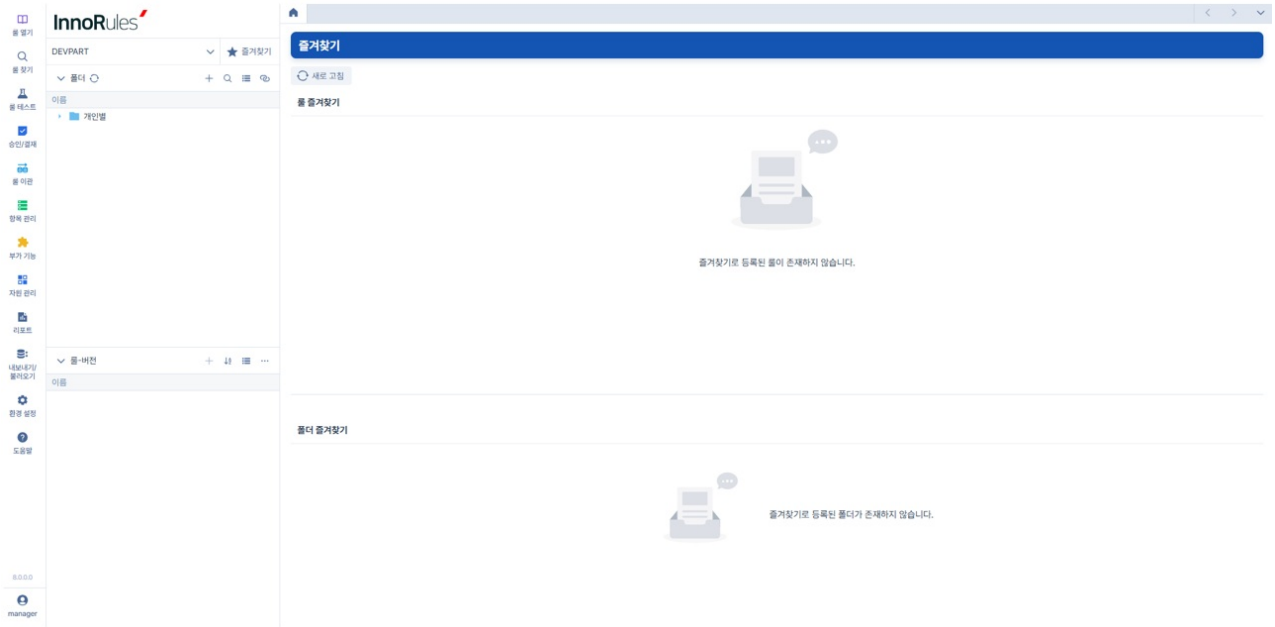

Figure 1 - 6 - 3

1.7 InnoRules 서버 중지

InnoRules 서버는 설치 디렉터리 하위의 script 디렉터리에서 Figure 1-7-1과 같이 다음 명령어로 중지 시킨다. Windows의 경우 stop-server.bat를 실행한다.

```
innorules@:~/innorules-home/script$ ./stop-server.sh
```

Figure 1 - 7 - 1

InnoRules 서버가 정상적으로 중지 되면 Figure 1-7-2와 같이 출력된다.

```
Using CATALINA_BASE:    /home/innorules/innorules-home/innorules-server/default
Using CATALINA_HOME:    /home/innorules/innorules-home/innorules-server/apache-tomcat-9
Using CATALINA_TMPDIR: /home/innorules/innorules-home/innorules-server/default/temp
Using JRE_HOME:         /opt/java/java-8
Using CLASSPATH:        :/home/innorules/innorules-home/lib/jdbc-drivers/*:/home/innorules/innorules-home/innorules-server/apache-tomcat-9/bin/bootstrap.jar:/home/innorules/innorules-home/innorules-server/apache-tomcat-9/bin/tomcat-juli.jar
```

Figure 1 - 7 - 2

2. 마법사

InnoRules는 제품을 설치하고 구성을 설정을 할 수 있도록 다음의 마법사들이 제공된다.

- ❒ 설치 마법사(Installation Wizard)
- ❒ 설정 클러스터 마법사(Configuration Cluster Wizard)
- ❒ 룰 시스템 구성 마법사(Rule System Configuration Wizard)
- ❒ 서비스 구성 마법사(Service Configuration Wizard)
- ❒ 룰 애플리케이션 구성 마법사(Rule Application Configuration Wizard)
- ❒ 애드온 마법사(Addon Wizard)

설치 마법사는 InnoRules를 설치하기 위해 사용되며, 설치가 완료된 이후 나머지 마법사들을 필요에 따라 실행한다. 이 장에서는 각 마법사의 기능과 사용 방법에 대해 설명하고, 가상의 시나리오를 가정하여 어떤 순으로 마법사들을 실행하는지를 알아본다.

2.1 설치 마법사

설치 마법사는 InnoRules를 대상 호스트에 설치하는 실행 가능한 jar 파일이다. 한 번의 설치 작업으로 설치된 일련의 디렉터리와 파일들을 설치 노드(Installation Node)라고 한다. 동일한 호스트라도 경로를 달리하여 여러 노드를 설치할 수 있다.

하나의 설치 노드에는 단위 룰 시스템 설정, 각종 InnoRules 서비스 설정, 룰 애플리케이션 설정들이 작성될 수 있다. 이 설정들은 서로 연관성 및 의존성을 갖고 있다. 따라서, 완전히 독립된 두 개의 룰 시스템을 하나의 설치 노드에 구성할 수는 없다. 독립적인 룰 시스템을 구성하려면 InnoRules를 추가적으로 설치하고 그 노드에 설정을 해야 한다.

각각의 설치 노드는 독립된 설치 디렉터리에 위치한다. 따라서, 설치 디렉터리를 지정할 때 이미 다른 설치 노드가 사용하는 디렉터리 또는 그 하위 디렉터리를 지정해서는 아니 된다. 또한, 각각의 설치 노드에는 고유한 식별자가 부여되므로 디렉터리 복사 등 비정상적인 방법으로 설치를 해서도 아니 된다.

설치 마법사는 Figure 2-1-1과 같이 다음 명령을 이용하여 실행한다.

```
Java –jar innorules-inst.jar
```

Figure 2 - 1 - 1

2.1.1 소개

Figure 2-1-2는 설치 마법사의 소개 단계이다. 이 단계에서는 InnoRules를 설치하는 데에 필요한 시스템 요구사항을 보여준다.

```
********************************************************************************
```

```
소개

********************************************************************************

InnoRules 8.0.0를 설치하시게 된 것을 환영합니다. 설치 이전에 다음 사전 요구 사항을 확인하십시오.
- JDK 1.8(또는 그 이상)
- InnoRules 설치 디렉터리. 적어도 100MB 이상의 여유 공간이 필요합니다.
- (선택 사항)서비스 포트와 제어 포트가 할당되어 있어야 합니다.
[엔터를 누르십시오]
```

Figure 2 - 1 - 2

2.1.2 JDK 홈 디렉터리

Figure 2-1-3은 JDK 홈 디렉터리를 선택하는 단계를 보여준다. InnoRules는 자바 기반 애플리케이션으로 JDK 1.8 이상을 필요로 한다. 이 단계에서 각종 마법사 및 InnoRules 서버가 사용할 JDK의 경로를 지정한다.

```
********************************************************************************

JDK 홈 디렉터리

********************************************************************************

JDK가 설치된 곳은 어디입니까? JDK 버전은 1.8 이상이어야 합니다.(기본값: /opt/java/java-8)

>>
```

Figure 2 - 1 - 3

설치 마법사는 설치 마법사를 실행한 자바의 경로를 이용하여 JDK가 설치된 디렉터리를 추정하여 기본값으로 보여준다. 만약 JDK의 위치를 추정할 수 없는 경우 기본값은 보이지 않는다.

만약 지정한 디렉터리가 올바르지 않은 JDK 디렉터리이거나 설치된 JDK가 손상된 경우, 적절한 실행 권한이 없는 경우 에러 메시지가 출력된다. 또한, 지정한 JDK가 1.8 이상이 아닌 경우에도 에러 메시지가 출력된다.

다음은 에러 메시지의 예이다.

- ❒ 경로가 올바르지 않을 경우

```
[에러] The path is not a directory.[엔터를 누르십시오]
or
[에러] There is no bin directory.[엔터를 누르십시오]
```

Figure 2 - 1 - 4

❒ JDK 버전이 낮은 경우

```
[에러] Java version is lower than 1.8.[엔터를 누르십시오]
```

Figure 2 - 1 - 5

2.1.3 설치 디렉터리

Figure 2-1-6은 설치 디렉터리를 지정하는 단계이다. 설치 디렉터리는 다음의 조건을 만족해야 한다.

- ❒ 이미 존재하는 디렉터리인 경우 비어 있는 디렉터리이어야 하며 마법사를 실행 한 OS 계정이 하위 디렉터리와 파일을 생성할 권한이 있어야 한다.
- ❒ 디렉터리가 아직 존재하지 않는 경우 마법사를 실행 한 OS 계정이 디렉터리를 생성할 권한이 있어야 한다.
- ❒ 디렉터리에는 최소 100MB의 여유 공간이 있어야 한다. 이 여유 공간은 최초 설치에 필요한 공간이다. 설치 후에도 InnoRules 서버의 로그, 서드 파티 라이브러리, JDBC 드라이버의 추가로 용량이 증가할 수도 있다. 여유 공간을 산정할 때에는 이 공간도 고려해야 한다.

```
********************************************************************************

설치 디렉터리

********************************************************************************

InnoRules를 어느 디렉터리에 설치하시겠습니까?(기본값: /home/user/innorules-home)

 >>
```

Figure 2 - 1 - 6

2.1.4 로그 디렉터리

Figure 2-1-7은 로그 디렉터리를 지정하는 단계이다. InnoRules 서버와 룰 애플리케이션들은 지정된 디렉터리의 하위에 디렉터리를 생성하고 로그를 기록한다.

```
********************************************************************************

로그 디렉터리
```

```
*******************************************************************************

어디에 로그 파일을 생성하시겠습니까? InnoRules 서비스와 다양한 룰 애플리케이션이 하위 디렉터리에 로그를 기록할 것입니다.(기본값: /home/user/innorules-home/innorules-server/logs)

>>
```

Figure 2 - 1 - 7

기본값은 설치 디렉터리의 innorules-server/logs 디렉터리이다. 2.1.3 설치 디렉터리에서 설명한 바와 같이 설치에 필요한 최소 공간에는 로그 파일이 고려되지 않았으므로 기본 로그 디렉터리를 사용하려는 경우 필요 공간에 로그의 용량을 고려하여야 한다.

InnoRules 서버뿐만 아니라 룰 애플리케이션도 이 디렉터리에 로그를 기록하므로 적절하게 권한을 부여하여야 한다. 만약 시스템에서 별도로 관리하는 로그 디렉터리가 있다면 공간이나 권한 관리 측면에서 그 디렉터리를 이용하는 것이 권장된다.

2.1.5 InnoRules 서버 설정

Figure 2-1-8은 InnoRules 서버가 사용할 포트를 지정하는 단계이다.

```
*******************************************************************************

InnoRules 서버 설정

*******************************************************************************

InnoRules 서버는 관리 서비스, 룰 빌더 서버스 등과 같은 서비스를 제공할 수 있는 독립 서버입니다. 서버는 이 설치 노드에 실행될 수도 있고 아닐 수도 있습니다만, 지금 포트를 설정해 놓는 것이 좋습니다. 제어 포트는 InnoRules 서버를 중지시키기 위해 사용됩니다.

포트 번호를 입력하십시오.

[필수] HTTP 서비스 포트(기본값: 25802) >>
[필수] 제어 포트(기본값: 25805) >>
```

Figure 2 - 1 - 8

InnoRules 서버는 Apache Tomcat 서버와 InnoRules 웹 애플리케이션으로 구성된다. 서비스 포트는 Apache Tomcat의 HTTP 포트이며 제어 포트는 Apache Tomcat 서버 종료에 사용되는 포트이다. 서비스 구성 마법사의 설정에 따라 InnoRules 웹 애플리케이션은 다양한 서비스를 제공하게 되는데 룰 빌더 서비스, 룰 웹 서비스 등이 HTTP 포트를 이용하여 서비스 된다.

모든 설치 노드에서 InnoRules 서버를 실행할 필요는 없다. InnoRules 서버를 실행하지 않는다면 임의의 포트를 지정하면 된다.

InnoRules 서버가 설치되면 각 서버에는 고유한 UUID가 부여가 된다. 이 UUID는 설정 클러스터 내에서 서버를 식별하는 데에 사용된다. 설정 클러스터에 대한 내용은 3.2 설정 클러스터를 참고하라.

2.1.6 라이선스 파일

Figure 2-1-9는 라이선스 파일을 선택하는 단계이다. 라이선스 파일은 제품을 구매할 때 설치 파일과 함께 제공된다. InnoRules를 설치할 때 라이선스 파일을 설치 마법사의 실행 경로에 위치시키면 아래와 같이 목록에 라이선스가 표시된다. 실행 경로에 두 개 이상의 라이선스 파일이 있다면 그 중 하나를 선택할 수 있다.
설치할 때 라이선스 파일을 지정하지 않고 설치가 완료된 다음 설치 디렉터리의 license 디렉터리에 수작업으로 복사를 해도 된다.

```
********************************************************************************

라이선스 파일

********************************************************************************

현재 작업 디렉터리에서 다음의 라이선스 파일을 찾았습니다. 어떤 라이선스 파일을 설치하시겠습니까?
(*) 1. innorules.lic
            InnoRules(Expires: 2025-12-31)
     2. 설치 이후 수작업으로 복사



선택하십시오. >>
```

Figure 2 - 1 - 9

2.1.7 요약 및 설치

Figure 2-1-10은 입력한 정보를 확인하여 설치를 진행하는 단계이다.

```
********************************************************************************

요약 및 설치

JDK Home              : /opt/java/java-8
Install Directory : /home/user/innorules-home
```

```
Service Port        : 25802
Admin Port          : 25805
License File        : /home/innorules/innorules.lic

이 설정으로 InnoRules를 설치합니다. 괜찮겠습니까? (Yes/No) >> y
[정보] Installation succeeded.[엔터를 누르십시오]
```

Figure 2 - 1 - 10

입력한 정보를 확인 후 설치를 완료하게 된다.

2.2 설정 클러스터 마법사

설치 노드의 구성 설정은 설치 디렉터리의 config/rule-systems.xml에 저장이 된다. 많은 경우 서로 다른 호스트의 설치 노드들이 설정을 공유해야 할 경우가 있다. 여러 설치 노드들이 설정을 공유하도록 함으로써 룰 시스템 설정에 변경이 일어날 때 각 설치 노드들에 일일이 변경 사항을 반영하지 않도록 할 수 있다. 이는 시스템 구성을 용이하게 하며 작업자 실수를 줄일 수 있게 해 준다. 설정 클러스터에 대한 내용은 3.2 설정 클러스터를 참고하라.

설정 클러스터링은 일련의 설치 노드들을 하나의 묶음으로 관리하여 그것들의 설정을 동기화 해 주는 기능이다. 설정 클러스터에 속한 설치 노드들 중 하나에서 설정을 변경하면 나머지 설치 노드에도 동일하게 변경된다. 하나의 설치 노드는 단 하나의 설정 클러스터에만 포함될 수 있다. 설정 클러스터 마법사는 설정 클러스터링을 구성해 주는 마법사이다.

설정 클러스터를 꼭 구성해야 하는 것은 아니다. 만약 룰 시스템이 하나의 설치 노드로만 이루어졌다면 설정 클러스터 구성을 하지 않아도 된다.

마법사는 script 디렉터리에서 Figure 2-2-1과 같이 다음의 명령으로 실행한다.

```
UNIX 계열에서:
./cluster-wiz.sh
Windows에서:
cluster-wiz.bat
```

Figure 2 - 2 - 1

2.2.1 소개

Figure 2-2-2는 설정 클러스터 마법사의 소개 단계이다.

```
********************************************************************************
소개
********************************************************************************

InnoRules 설정 클러스터 마법사에 오신 것을 환영합니다. 이 마법사를 이용하여 설정 클러스터를 관리할 수 있습니다.
[엔터를 누르십시오]
```

Figure 2 - 2 - 2

2.2.2 설치 노드들(최초)

Figure 2-2-3은 설정 클러스터를 구성하는 설치 노드들의 목록을 보여준다. 여기서 설정 클러스터는 마

법사가 실행되고 있는 설치 노드가 속한 클러스터를 의미한다.

```
*******************************************************************************
설치 노드들
*******************************************************************************

작업을 선택하십시오.

[항목들]
-----------------------------------
    | 태그 | 가용 | 로컬 | 동기화 |
-----------------------------------

[가능한 작업들]
a. 노드 추가      c. 동기화      e. 새로 고침

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 2 - 3

설정 클러스터가 아직 구성되어 있지 않은 경우 노드 추가, 동기화, 새로 고침 메뉴만 보인다.

2.2.3 원격 노드 설정을 위한 프로토콜

Figure 2-2-3에서 노드 추가를 선택하면 Figure 2-2-4와 같이 원격 설치 노드의 프로토콜을 선택하게 된다. 설정 클러스터링은 원격 설치 노드의 설정 파일을 수정해야 하기 때문에 이 파일을 원격 접근할 수 있는 방법을 기술해 줘야 한다. 원격 파일 접근을 위한 프로토콜로 현재 SFTP와 FTP가 지원된다.

```
*******************************************************************************
설정 노드들 -> 원격 설치 노드의 프로토콜
*******************************************************************************

프로토콜을 선택하십시오.
(*) 1. SFTP
        SFTP를 이용하여 원격 설정에 접근합니다. 호스트의 주소, 포트, ID, 비밀
        번호와 설치된 절대 경로가 필요합니다.
    2. FTP
        FTP를 이용하여 원격 설정에 접근합니다. 호스트의 주소, 포트, ID, 비밀번
        호와 설치된 절대 경로가 필요합니다.
선택하십시오. >>
```

Figure 2 - 2 - 4

2.2.4 프로토콜 상세

프로토콜 상세는 Figure 2-2-4에서 선택한 프로토콜에 따라 추가적으로 필요한 정보를 입력하는 화면이다.

SFTP를 선택한 경우 Figure 2-2-5의 화면이 나타난다.

```
********************************************************************************
설정 노드들 -> 프로토콜 상세
********************************************************************************

SFTP 프로토콜과 관련된 정보를 입력하십시오.
    id: User ID
    *password: Password
    host: Remote Host Name or Address
    port: SSH Port
    remote-dir: Remote InnoRules Home Directory

id >>
```

Figure 2 - 2 - 5

Figure 2-2-5의 각 항목 의미는 다음과 같다.

- ❒ id : SFTP ID
- ❒ password : SFTP Password
- ❒ host : 원격 호스트의 IP
- ❒ port : 원격 호스트의 SFTP Port
- ❒ remote-dir : 원격 호스트의 설치 경로

Figure 2-2-4에서 FTP를 선택한 경우 Figure 2-2-6의 화면이 나타난다.

```
********************************************************************************
설정 노드들 -> 프로토콜 상세
********************************************************************************

FTP 프로토콜과 관련된 정보를 입력하십시오.
    id: User ID
    *password: Password
    host: Remote Host Name or Address
```

```
    port: FTP Port
    remote-dir: Remote InnoRules Home Directory


id >>
```

Figure 2 - 2 - 6

Figure 2-2-6의 각 항목 의미는 다음과 같다.

- ❒ id : FTP ID
- ❒ password : FTP Password
- ❒ host : 원격 호스트의 IP
- ❒ port : 원격 호스트의 FTP Port
- ❒ remote-dir : 원격 호스트의 설치 경로

2.2.5 로컬 설치 노드의 등록

현재 마법사를 실행하고 있는 설치 노드를 **로컬 설치 노드(Local Installation Node)**라고 한다. Figure 2-2-3에서 설치 노드를 추가할 때에는 로컬 설치 노드를 가장 먼저 추가해야 한다. 그렇지 않은 경우 Figure 2-2-7과 같은 에러가 발생한다.

```
[에러] 설정 노드를 등록하는 도중 에러 발생: Cannot update the configuration. Not all clusters are synchronized or there is no local config.
...
 [엔터를 누르십시오]
```

Figure 2 - 2 - 7

클러스터에 로컬 설치 노드를 추가할 때 IP 주소로 루프백 주소(localhost, 127.0.0.1)를 사용해서는 아니 된다. IP 주소는 이후에 추가될 설치 노드들에서 이 설치 노드의 설정을 원격 접근할 때에도 사용되기 때문에 다른 설치 노드가 인식 및 접근 가능한 IP 주소를 사용해야 한다. 만약 루프백 주소를 입력하는 경우 Figure 2-2-8과 같은 에러가 발생한다.

```
[에러] 설정 노드를 등록하는 도중 에러 발생: Loopback address is not allowed.
java.lang.Exception: Loopback address is not allowed.
 [엔터를 누르십시오]
```

Figure 2 - 2 - 8

2.2.6 원격 설치 노드의 등록

설정 클러스터에 로컬 설치 노드를 등록하였다면 원격의 설치 노드를 클러스터에 추가할 수 있다. 원격 설치 노드를 클러스터에 추가하는 방법은 로컬 설치 노드를 추가하는 방법과 동일하다.

원격의 설치 노드가 신규로 설치되어 config 디렉터리가 비어 있을 경우, 클러스터에 추가될 때 로컬 설치 노드의 설정이 원격에 복사가 된다. 만약 원격 설치 노드에서 설치 이후에 설정 작업을 했다면(그래서 rule-systems.xml이 존재한다면) 설치 노드를 추가할 때 Figure 2-2-9와 같은 경고 메시지를 보게 될 것이다.

```
원격 설정 노드에서 이전의 설정 파일이 발견되었으나 로컬의 설정과 상이합니다. 로컬의 설정을 새로운 원격 노드에 복사하시겠습니까? (Yes/No) >>
```

Figure 2 - 2 - 9

Yes를 선택하면 원격 설정은 로컬의 설정으로 덮어써지고 원격 설치 노드는 클러스터에 추가된다. No를 선택하면 원격의 설정은 변경되지 않으며 클러스터에도 추가되지 않는다.

2.2.7 설치 노드들

Figure 2-2-10은 로컬 설치 노드와 또 다른 설치 노드를 클러스터에 추가하였을 때의 설치 노드들 단계를 보여준다.

정상적으로 로컬 설정이 추가되면 다음과 같은 노드 리스트가 나타난다.

```
********************************************************************************
설정 노드들
********************************************************************************

작업을 선택하십시오.

[항목들]
--------------------------------------------------------------------------------
    | 태그                                                          | 가용 | 로컬 | 동기화 |
--------------------------------------------------------------------------------
  1 | SFTP/user@192.168.0.100:22//home/user/innorules-home          | V    | V    | V      |
--------------------------------------------------------------------------------
  2 | SFTP/user@192.168.0.101:22//home/user/innorules-home-remote   | V    |      | V      |
--------------------------------------------------------------------------------

[가능한 작업들]
a. 노드 추가     b. 노드 제거     c. 동기화     d. 문제점 상세 보기     e. 새로 고침

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 2 - 10

목록의 각 필드의 의미는 다음과 같다.

- 태그
  원격 설치 노드에 접근하는 방법을 표현하는 문자열이다. 문자열의 맨 앞에는 프로토콜이 위치하며 그 이후에는 프로토콜에 의존적인 설정들이 따른다.
- 가용
  현재 목록의 설치 노드에 접근 가능한지를 보여준다. 설치 노드에 원격 접근할 수 없을 때는 설정 클러스터 마법사를 제외한 나머지 마법사를 이용하여 설정을 변경할 수 없다. 만약, 관리상의 목적으로 원격 호스트를 중지시킨 상태에서 룰 시스템의 설정을 변경하려면 b. 노드 제거 메뉴를 이용하여 가용하지 않은 설치 노드를 클러스터에서 제거한 이후에 설정을 변경한다. 이후 원격 호스트가 다시 가동되면 다시 클러스터에 원격 설치 노드를 추가한다.
- 로컬
  목록의 설치 노드가 로컬 설치 노드인가를 나타낸다. 로컬 설치 노드의 가용 필드와 동기화 필드는 항상 체크되어 있다.
- 동기화
  로컬 설치 노드의 설정 기준으로 목록의 설치 노드의 설정의 동일 여부를 나타낸다. 동기화되어 있지 않은 경우 설치 노드 추가를 포함하여 어떤 설정도 변경할 수 없다. 만약 동기화가 되어 있지 않은 상태에서 설치 노드를 추가하려고 한다면 Figure 2-2-7의 에러가 발생한다.

작업 메뉴는 다음과 같다.

- 노드 추가
  새로운 설치 노드를 설정 클러스터에 추가한다.
- 노드 제거
  등록된 설치 노드를 클러스터로부터 제거한다. 로컬 설치 노드는 가장 나중에 제거되어야 한다. 그렇지 않은 경우 에러가 발생한다.
- 동기화
  만약 어떠한 이유로 원격 설치 노드의 설정이 로컬 설치 노드의 설정과 상이한 경우 c. 동기화 메뉴를 이용하여 동기화 시킬 수 있다. 동기화를 하면 로컬의 설정을 원격 설치 노드에 복사를 하기 때문에 어느 설치 노드에서 동기화 메뉴를 실행하느냐에 따라 결과가 달라질 수 있다. 따라서, 동기화가 되어 있지 않을 때에는 그 원인을 확인하고 설정이 최신인 설치 노드에서 동기화 메뉴를 실행해야 한다.
- 문제점 상세 보기
  원격 설치 노드가 가용하지 않은 경우 상세한 메시지를 볼 수 있다.
- 새로 고침
  원격 설치 노드의 가용 상태 및 동기화 상태를 갱신한다.

2.2.8 UUID 파일

설정 클러스터 마법사를 포함한 모든 설정 마법사는 설정 클러스터의 노드들 중 어떤 노드가 로컬 설치 노드인지 확인하기 위해서 마법사가 시작될 때 config 디렉터리에 UUID 파일을 생성한다. UUID 파일은 크기는 0이며 다음과 같은 형식의 파일 이름을 갖는다.

550e8400-e29b-41d4-a716-446655440000

마법사가 정상 종료되면 UUID 파일은 지워지지만 비정상적으로 종료되는 경우 지워지지 않을 수도 있다. 만약 로컬 호스트에서 마법사가 실행 중이지 않음에도 불구하고 config 디렉터리에 UUID 파일이 존재한다면 지워도 무방하다.

2.2.9 락 파일

원격의 설정 파일을 독점적으로 접근하기 위해서 설정 클러스터 마법사를 포함한 모든 설정 마법사는 시작할 때 원격의 config 디렉터리에 lock 파일을 생성하여 락을 획득한다. lock 파일에는 어떤 설치 노드에서 락을 획득했는가에 대한 정보가 기록된다. 만약 마법사가 시작될 때 노드들 중 락을 획득하지 못하는 노드가 있는 경우 Figure 2-2-11의 경고가 발생한다.

```
원격 설정 노드에서 락 파일이 발견되었습니다. 다른 설정이 진행 중이거나 이전의 설정 작업이 비정상적으로 종료되었을 수 있습니다. 락 파일을 제거하고 새로 만드시겠습니까? (Yes/No) >>
```

Figure 2 - 2 - 11

lock 파일은 마법사가 정상 종료되면 삭제된다. 그러나, 어떤 이유로 마법사가 비정상 종료되면 lock 파일이 지워지지 않고 남아 있을 수 있다. Figure 2-2-11의 에러는 다른 마법사가 이미 실행되고 있거나 이전에 실행된 마법사가 비정상 종료가 되었음을 나타낸다. 따라서, 마법사를 이용한 설정 작업이 이루어지지 않고 있음에도 불구하고 Figure 2-2-11의 에러가 발생하면 Yes를 선택하여 기존의 lock 파일을 제거하고 새로운 lock 파일을 생성하여 락을 획득하게 할 수가 있다. 그러나, 그 이전에 마법사가 실행 중인지부터 명확하게 확인을 해 봐야 한다.

2.3 룰 시스템 구성 마법사

룰 시스템 구성 마법사는 룰 시스템을 이루는 단위 룰 시스템들을 구성 및 관리하며 룰 시스템이 접근할 업무 데이터베이스의 접속 정보를 관리하도록 도와주는 마법사이다.

룰 시스템 구성 마법사는 script 디렉터리에서 Figure 2-3-1과 같이 다음의 명령으로 실행한다.

```
UNIX 계열에서:
./system-wiz.sh
Windows에서:
system-wiz.bat
```

Figure 2 - 3 - 1

2.3.1 소개

Figure 2-3-2는 룰 시스템 구성 마법사의 소개 단계이다.

```
********************************************************************************

소개

********************************************************************************

룰 시스템 구성 마법사에 오신 것을 환영합니다. 이 마법사는 룰 시스템을 구성하도록 도와줍니다.
[엔터를 누르십시오]
```

Figure 2 - 3 - 2

2.3.2 설정 대상 선택

시스템 설정 마법사를 이용하여 단위 룰 시스템을 관리하거나 업무 데이터베이스를 관리할 수 있다. 단위 룰 시스템은 독립적으로 서비스될 수 있는 룰들과 이들을 저장하기 위한 DB들, 이들을 실행하기 위한 애플리케이션 설정 등으로 구성된다. 단위 룰 시스템에 대한 자세한 사항은 3.3.1 단위 룰 시스템을 참조하라.
업무 데이터베이스는 룰이 실행되는 동안 참조할 수 있는 룰 시스템 외부의 데이터베이스이다. 업무 데이터베이스 관리에서는 업무 데이터베이스의 목록과 그에 대한 기본 접속 정보를 관리한다. 동일한 목적의 업무 데이터베이스라도 단위 룰 시스템에 따라, 그리고 애플리케이션에 따라 접속 정보를 서로 다르게 관리할 수 있다. 업무 데이터베이스에 대한 자세한 사항은 3.4 업무 데이터베이스를 참조하라.

```
********************************************************************************

```

```
설정 대상 선택

********************************************************************************

설정 대상을 선택하시오.
    1. 단위 룰 시스템 관리
        단위 룰 시스템 설정을 관리합니다.
    2. 업무 데이터베이스 관리
        업무 데이터베이스 설정을 관리합니다.



선택하십시오. >>
```

Figure 2 - 3 - 3

Figure 2-3-3에서 단위 룰 시스템 관리를 선택한 경우 2.3.3 단위 룰 시스템 관리 화면으로 이동하며, 업무 데이터 베이스 관리를 선택하면 2.3.4 업무 데이터베이스 목록으로 이동한다.

2.3.3 단위 룰 시스템 관리

Figure 2-3-4는 등록된 단위 룰 시스템들을 보여준다. 각 단위 룰 시스템에는 고유한 ID와 이름이 부여되어 있다.

```
********************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리

********************************************************************************

등록된 단위 룰 시스템의 목록입니다. 새로운 단위 룰 시스템을 추가하거나 기존의 것을 수정할 수 있습니다.

[항목들]
----------------------------
    | ID         | 이름             |
----------------------------
  1 | DEV        | Development |
----------------------------
  2 | TEST       | Test             |
----------------------------
```

```
[가능한 작업들]
a. 추가      b. 수정 및 룰 DB 관리      c. 마지막 단위 룰 시스템 삭제
작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 3 - 4

다음은 각 작업 메뉴의 기능이다.

- ❒ 추가
  단위 룰 시스템을 추가하고 기본 룰 DB를 등록한다.
- ❒ 수정 및 룰 DB 관리
  단위 룰 시스템의 정보를 수정하고, 단위 룰 시스템에 등록된 룰 DB들을 관리한다
- ❒ 마지막 단위 룰 시스템 삭제
  목록의 가장 아래에 있는 단위 룰 시스템을 제거한다. 단위 룰 시스템에 하나도 등록되어 있지 않은 경우 이 메뉴는 보이지 않는다.

등록된 단위 룰 시스템의 순서는 전체 룰 시스템에서 중요한 의미를 갖는다. 위의 단위 시스템에서 작성된 룰들 중 일부는 바로 아래의 단위 시스템으로 이관할 수 있다. 이에 대한 자세한 사항은 3.3.2 이관을 참조하라.

2.3.3.1 단위 룰 시스템 기본 정보

단위 룰 시스템을 추가하기 위해서는 Figure 2-3-5와 같이 기본 정보를 입력해야 한다.

```
*******************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보

*******************************************************************************

단위 룰 시스템의 기본 정보를 입력하십시오.

[필수] ID >> PROD
[필수] 이름 >> Production
```

Figure 2 - 3 - 5

단위 룰 시스템의 기본 정보에는 ID와 이름이 있다.

- ❒ ID
  1자에서 10자로 이루어질 수 있으며 알파벳 대소문자, 숫자, 언더스코어('_')로만 이루어져야 한다. 전체 룰 시스템 내에서 단위 룰 시스템의 ID는 고유해야 한다. 마법사가 단위 룰 시스템 ID의 고유성을 체크해 주기는 하지만, 설정 클러스터가 룰 시스템의 일부만을 관리할 경우 룰 시스템 내에서의 고유성을 체크하지는 못한다.

❒ 이름
단위 룰 시스템의 이름은 단위 룰 시스템의 목적을 사람이 이해할 수 있도록 도와주는 목적을 한다. 60자 이내로 이루어져야 하며 문자의 제한은 없다. 이름 역시 전체 룰 시스템 내에서 고유해야 한다.

단위 룰 시스템을 추가하면 Figure 2-3-4의 가장 마지막에 추가된다. 일단 추가가 되면 그 순서를 변경할 수는 없다. 따라서, 단위 룰 시스템을 추가할 때에는 전체 룰 시스템을 어떤 단위 룰 시스템들로 구성할 것인지 심사숙고해야 한다.
만약 이미 존재하는 두 단위 룰 시스템 사이에 다른 단위 시스템을 추가하려고 한다면, 뒤쪽의 단위 룰 시스템을 제거한 후 삽입하려는 단위 룰 시스템을 추가한 다음 제거했던 단위 룰 시스템을 다시 추가해야 한다.

시스템 ID와 이름을 입력한 후 엔터를 누르면 2.3.3.5 룰 DB 기본 정보로 이동한다.

2.3.3.2 단위 룰 시스템 기본 정보 수정

Figure 2-3-6은 단위 룰 시스템의 기본 정보를 수정하는 단계를 보여준다.

```
********************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보

********************************************************************************

단위 룰 시스템의 기본 정보를 입력하십시오.

[필수] 이름(기본값: Development) >>
```

Figure 2 - 3 - 6

단위 룰 시스템의 이름만 수정이 가능하다. ID는 수정이 불가능하다. 기존의 단위 룰 시스템 이름이 기본값으로 보인다. 수정 또는 수정 없이 엔터를 누르면 2.3.3.4 룰 DB 관리로 이동한다.

2.3.3.3 마지막 단위 룰 시스템 삭제

마지막에 등록된 단위 룰 시스템을 Figure 2-3-7과 같이 삭제할 수 있다.

```
********************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리

```

```
********************************************************************************

등록된 단위 룰 시스템의 목록입니다. 새로운 단위 룰 시스템을 추가하거나 기존의 것을 수정할 수 있습니다.

[항목들]
----------------------------
    | ID        | 이름             |
----------------------------
  1 | DEV       | Development |
----------------------------
  2 | TEST      | Test             |
----------------------------

[가능한 작업들]
a. 추가     b. 수정 및 룰 DB 관리     c. 마지막 단위 룰 시스템 삭제

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >> c

단위 룰 시스템 TEST을(를) 삭제하시겠습니까? 삭제하면 룰 DB의 내용은 보존되지만 관련된 설정(룰 애플리케이션, TCP 커넥터 서버 등)은 복원할 수 없습니다.  (Yes/No) >>
```

Figure 2 - 3 - 7

목록의 단위 룰 시스템 중 가장 마지막의 단위 룰 시스템만 삭제가 가능하다. 단위 룰 시스템이 삭제되더라도 룰 DB의 데이터는 삭제되지 않고 유지된다. 그러나, 단위 룰 시스템과 관련된 다른 설정들은 모두 삭제된다. 삭제되는 설정에는 다음과 같은 것들이 있다.

- ❒ 룰 애플리케이션 설정
- ❒ TCP 커넥터 및 REST 서비스 설정
- ❒ 업무 데이터베이스 기본 접속 설정

2.3.3.4 룰 DB 관리

룰 DB 관리는 단위 룰 시스템에 등록된 룰 DB들의 목록을 Figure 2-3-8과 같이 보여주는 단계이다.

```
********************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보 -> 룰 DB 관리

********************************************************************************
```

```
단위 룰 시스템 'DEV'에 등록된 룰 DB의 목록입니다. 새로운 룰 DB를 등록하거나 기존의 룰 DB 정보를 수정할 수 있습니다.

[항목들]
-------------------------------------------------------------------------------------
    | 이름        | JDBC URL                                          | 사용자 계정       | 설명 |
-------------------------------------------------------------------------------------
  1 | RULEDB1 | jdbc:oracle:thin:@onlinedb:1521:ORCL      | innorules       |       |
-------------------------------------------------------------------------------------
  2 | RULEDB2 | jdbc:oracle:thin:@batchdb:1521:ORCL       | innorules       |       |
-------------------------------------------------------------------------------------

[가능한 작업들]
a. 추가     b. 정보변경     c. 제거     d. 위로     e. 아래로

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 3 - 8

룰 DB는 룰 데이터가 저장되는 영속적인 저장 공간이다. 룰 DB에 대한 자세한 사항은 3.3.1.1 룰 저장소을 참조하라.

하나의 단위 시스템에는 여러 개의 룰 DB를 등록할 수 있다. 룰 DB의 등록 순서는 의미가 있다. 룰 빌더 서버가 룰 데이터를 저장할 때에는 등록된 모든 룰 DB에 동일하게 데이터를 저장한다. 그러나, 룰 데이터를 조회할 때는 가장 우선 순위가 높은 룰 DB로부터 데이터를 읽어 들인다. Figure 2-3-8에서는 RULEDB1이 이에 해당된다. 룰 DB 목록에서 가장 위에 있는 룰 DB를 마스터 룰 DB라고 한다.

룰 빌더 서버는 모든 룰 DB에 동일한 데이터를 저장하기 때문에 룰 데이터의 동기화를 보장한다. 따라서, 사용자는 룰 DB 데이터를 임의로 변경해서는 아니 된다. 이럴 경우 예기치 않은 문제가 발생할 수 있다.

다음은 각 작업 메뉴의 기능이다.

- ❒ 추가
  룰 DB를 추가한다
- ❒ 정보변경
  룰 DB의 접속 정보를 변경한다. 접속 정보 변경 화면은 룰 DB 추가 화면과 동일하다.
- ❒ 제거
  룰 DB를 제거한다. 단위 시스템에는 최소한 1개의 룰 DB가 등록되어 있어야 하므로, 등록된 룰 DB가 1개일 때에는 이 메뉴가 나타나지 않는다. 룰 DB를 제거하더라도 데이터가 삭제되지는 않는다.
- ❒ 위로
  목록에서 바로 위의 룰 DB와 우선 순위를 바꾼다.

❒ 아래로
목록에서 바로 아래의 룰 DB와 우선 순위를 바꾼다.

2.3.3.5 룰 DB 기본 정보

추가할 룰 DB의 기본 정보를 Figure 2-3-9와 같이 입력하는 단계이다.

```
*********************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보 -> 룰 DB 기본 정보

*********************************************************************************

룰 DB의 기본 정보를 입력하십시오. 정보는 한 번 지정하면 변경할 수 없습니다.

[필수] 이름 >>
[필수] 설명(기본값: ) >>
```

Figure 2 - 3 - 9

룰 DB의 이름은 1자부터 20자 이내의 영문자 대소문자, 숫자 또는 언더스코어('_')로 이루어져야 한다. 룰 DB 이름은 단위 룰 시스템 안에서 고유해야 한다. 룰 DB의 설명에는 룰 DB에 대한 간략한 설명을 입력한다. 이 설명은 Figure 2-3-8의 목록에 나타난다.

룰 DB가 목록에 일단 추가되고 나면 이름 또는 설명을 변경할 수 없다. 만약 변경하고 싶다면 룰 DB를 제거했다가 다시 등록해야 한다.

이름과 설명을 입력하고 엔터를 입력하면 2.3.3.6 룰 DB 서버 유형 선택으로 이동한다.

2.3.3.6 룰 DB 서버 유형 선택

Figure 2-3-10은 룰 DB로 사용한 데이터베이스 서버를 선택하는 단계이다.

```
*********************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보 -> 룰 DB 서버 유형

*********************************************************************************

룰 DB로 사용할 데이터베이스 서버의 유형이 무엇입니까?
```

```
    1. Oracle
        Oracle database (with type 4 driver)
    2. MSSQL
        Microsoft SQL server
    3. DB2
        IBM DB2 with DB2 UDB JDBC universal driver
    4. MySQL
        MySQL database server
    5. Postgre
        Postgre database server

선택하십시오. >>
```

Figure 2 - 3 - 10

룰 DB로 사용할 데이터베이스 서버를 목록에서 선택해야 한다. InnoRules는 다양한 데이터베이스 서버에 접속할 수 있는 방법과 룰 테이블들을 생성하기 위한 DDL를 준비하고 있다. 만약, 사용하고 있는 데이터베이스 서버가 목록에 없다면 기술지원을 요청하도록 한다.

비록 InnoRules는 여러 데이터베이스에 대한 접속 방법과 DDL을 제공하기는 하지만 JDBC 드라이버는 별도로 제공을 하지 않는다. JDBC는 데이터베이스 서버 벤더가 제공하는 최신의 안정적인 JDBC 드라이버를 사용하여야 하며, 설치 마법사를 실행하기 전에 이 드라이버를 lib/jdbc-drivers에 미리 복사를 해 놓아야 한다. 각 데이터베이스를 선택했을 때 필요한 JDBC 드라이버 클래스는 다음과 같다.

- Oracle
  oracle.jdbc.driver.OracleDriver
- MSSQL
  com.microsoft.sqlserver.jdbc.SQLServerDriver
- DB2
  com.ibm.db2.jcc.DB2Driver
- MySQL
  com.mysql.jdbc.Driver
- Postgre
  org.postgresql.Driver

만약 여기에 기술된 JDBC 드라이버 이외의 드라이버를 사용해야 한다면 기술지원을 요청한다.

데이터베이스 서버를 선택하였다면 DB에 접속 정보를 입력하는 2.3.3.7 룰 DB 접속 정보로 이동한다.

2.3.3.7 룰 DB 접속 정보

Figure 2-3-11은 추가할 룰 DB의 접속 정보를 입력하는 단계이다.

```
*********************************************************************************************

설정 유형 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보 -> 룰 DB 접속 정보

*********************************************************************************************

룰 DB RULEDB3에 접속하기 위한 정보를 입력하십시오.

[필수] address >>
[필수] port(기본값: 1521) >>
[필수] SID or service name(기본값: ORCL) >>
[필수] 사용자 계정 >>
[필수] 비밀번호 >>

이 정보를 이용하여 룰 DB를 등록하시겠습니까? 등록을 할 경우 이 접속 정보로 DB에 로그인하여 테이블을 생성하거나 데이터를 수정하게 될 수도 있습니다. (Yes/No) >>
```

Figure 2 - 3 - 11

데이터베이스 유형마다 사용할 JDBC URL의 형식이 미리 지정되어 있으며, 이에 따라 필요한 정보를 사용자에게 질의하게 된다. 데이터베이스 유형에 따른 JDBC URL 형식은 lib/jdbc-drivers/database.xml에 정의되어 있다. 만약, 정의되어 있는 URL 형식이 사용하고자 하는 URL의 형식과 다른 경우 기술지원을 요청한다.

추가된 룰 DB는 기존에 추가되어 있는 단위 룰 시스템 및 룰 DB들과 정합성을 유지해야 한다. 정합성 유지를 위해 마법사는 룰 DB에 테이블을 생성하거나 데이터를 변경할 수도 있다. 어떤 작업을 할 것인가는 경우에 따라 다르다.

- ❒ 현재의 단위 시스템이 가장 앞 단계의 단위 시스템인 경우, 그리고 등록하려는 룰 DB가 그 단위 시스템의 첫 번째 룰 DB인 경우: 테이블이 존재하지 않는다면 테이블을 생성하고 초기 데이터를 입력한다. 테이블이 존재한다면 테이블을 검증 후 룰 데이터를 그대로 사용한다.
- ❒ 현재의 단위 시스템이 가장 앞 단계의 단위 시스템이 아닌 경우, 그리고 등록하려는 룰 DB가 그 단위 시스템의 첫 번째 룰 DB인 경우: 테이블이 존재하지 않는다면 테이블을 생성하고 초기 데이터를 입력한다. 테이블이 존재한다면 테이블 검증 및 이전 단위 룰 시스템의 데이터에 위배되지 않는지 플래그 검사 등을 실시한다. 이상이 없는 경우 등록한다.
- ❒ 등록하려는 룰 DB가 그 단위 시스템의 첫 번째 룰 DB가 아닌 경우: 테이블이 존재하지 않는다면 테이블을 생성하고 첫 번째 룰 DB의 데이터를 복사한다. 테이블이 존재한다면 테이블을 검증 후 추가하려는 DB와 첫 번째 룰 DB의 데이터가 동일한 경우만 DB 추가를 허용한다.

추가가 완료되면 2.3.3.8 기본 날짜 형식로 이동한다.

2.3.3.8 기본 날짜 형식

날짜 형식을 사용하는 룰 문법에서 사용하는 기본 날짜 형식을 입력하는 단계이다. 이 단계는 개발계 룰 시스템의 첫 번째 룰 데이터베이스를 추가하는 경우에만 출력이 되며, 기본 날짜 형식을 지정하면 이후 추가되는 모든 룰 시스템은 동일한 날짜 형식을 사용하게 된다. 이 날짜 형식은 Java의 SimpleDateFormat의 형태를 준수한다. 기본 날짜 형식은 설정 후, 변경이 불가능 하다.

```
********************************************************************************

룰 시스템에서 사용할 기본 날짜 형식을 입력하십시오. 기본 날짜 형식은 날짜 형식을 사용하는 룰 문법에서 사용
됩니다. 이 형식은 Java의 SimpleDateFormat형식을 준수하여야 합니다.(기본값: yyyy-MM-dd)

 >>
```

Figure 2 - 3 - 12

기본값이 아닌 다른 값을 입력하는 경우 아래와 같이 재확인 메시지가 출력된다.

```
날짜 형식은 입력 후 변경이 불가능하며, 이후에 등록되는 룰 시스템에 동일한 날짜 형식으로 입력됩
니다. 이 날짜 형식을 등록하시겠습니까?: yyyyMMdd (Yes/No) >>
```

Figure 2 - 3 - 13

2.3.3.9 단위 룰 시스템 기본 정보의 처리

다른 시스템에서 등록된 룰 DB를 사용하여 신규 단위 룰 시스템이 추가된 경우(2.3.3.7 룰 DB 접속 정보 단계에서 테이블을 생성하지 않고, 룰 데이터를 그대로 사용하게 설정한 경우) 룰 시스템 구성 마법사는 해당 룰 DB에 존재하는 단위 룰 시스템 기본 정보를 어떻게 처리할 것인지 질의한다. 이 단계는 2.3.3.1 단위 룰 시스템 기본 정보 단계에서 사용자가 입력한 단위 룰 시스템의 기본 정보와 지정된 룰 DB상의 룰 시스템 기본 정보(ID, 이름, 시스템 레벨 등)가 상이할 경우 수행된다. 즉, 아래와 같은 경우이다.

- ❒ 사용자가 입력한 단위 룰 시스템의 ID, 이름이 룰 DB에 저장된 값과 다른 경우
- ❒ 현재 선택한 단위 시스템의 단계(시스템 레벨)와 룰 DB상의 룰 시스템 단계(시스템 레벨)가 다른 경우

출력 화면은 아래와 같다. 출력 메시지에는 현재 지정된 룰 DB상의 단위 룰 시스템 정보를 포함한다.

```
********************************************************************************

등록하려는 룰 DB에 다른 단위 시스템에서 사용했던 시스템 정보가 존재합니다. 시스템 정보를 어떻게 처리하시
겠습니까?
```

```
        시스템 ID: TEST, 시스템 이름: test, 시스템 레벨: 2, 기본 날짜 형식: yyyy-MM-dd
    1. 룰 DB의 시스템 정보 사용
        룰 DB상에 존재하는 시스템 정보를 사용하여 단위 룰 시스템 추가
    2. 입력된 시스템 정보 사용
        사용자가 입력한 시스템 정보를 사용하여 룰 DB의 시스템 정보를 갱신하고
        단위 룰 시스템 추가


선택하십시오. >>
```

Figure 2 - 3 - 14

사용자는 룰 DB에 존재하는 시스템 정보를 이용하여 시스템을 추가하거나 입력한 시스템 정보로 룰 DB를 갱신하고 시스템을 추가하여 진행할 수 있다. 만약 현재 추가하고 있는 단위 시스템이 가장 앞 단계의 단위 시스템이 아니고, 룰 DB상의 룰 시스템 기본 정보가 현재의 단위 시스템 보다 앞 단계의 단위 시스템인 경우, 룰 시스템 구성 마법사는 위의 1번 선택지를 표시하지 않고 입력된 시스템 정보를 사용하여 단위 룰 시스템을 추가하도록 한다.

사용자가 입력된 시스템 정보 사용을 선택한 경우 아래와 같이 확인 메시지가 출력이 된다.

```
시스템 정보를 업데이트 하시겠습니까? 시스템 정보를 제외한 다른 룰 데이터는 변경되지 않습니다. (Yes/No) >>
```

Figure 2 - 3 - 15

만약 사용자가 입력한 시스템 정보를 룰 DB에 갱신하는 경우, 해당 룰 DB를 참조하는 다른 설치 노드가 존재한다면, 해당 설치 노드가 가지는 시스템 정보와 룰 DB의 시스템 정보가 상이하여, 해당 설치 노드에서 시스템 초기화 오류가 발생할 수도 있다. 따라서, 시스템 기본 정보를 어떻게 처리할지 선택하기 전에 해당 룰 DB가 다른 설치 노드에서 사용되고 있는지를 확인하고 진행하여야 한다.

2.3.4 업무 데이터베이스 목록

룰 시스템에서 접근할 업무 데이터베이스들의 목록을 Figure 2-3-16와 같이 관리하는 단계이다.

```
********************************************************************************

설정 유형 선택 -> 업무 데이터베이스 목록

********************************************************************************

업무 데이터베이스를 관리합니다. 어떤 작업을 하시겠습니까?

[항목들]
```

```
--------------------------------------------------------------------------------
   | ID | 이름  | 설명                          | 접속정보 미등록 |
--------------------------------------------------------------------------------
  1 | 1  | APPDB | application database | V                |
--------------------------------------------------------------------------------
  2 | 2  |       |                              |                  |
--------------------------------------------------------------------------------
 [가능한 작업들]
a. 수정      b. 삭제

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 3 - 16

목록의 각 필드의 의미는 다음과 같다.

- ❒ ID
  업무 데이터베이스의 ID이다. 업무 데이터베이스의 ID는 일련번호로 부여되며 임의로 부여할 수 없다.
- ❒ 이름
  업무 데이터베이스의 이름이다. 이름이 비어 있는 것은 그 ID로는 업무 데이터베이스가 등록되어 있지 않음을 의미한다. 이름이 비어 있는 슬롯은 항상 하나 이상 존재한다.
- ❒ 설명
  업무 데이터베이스의 설명이다.
- ❒ 접속정보 미등록
  업무 데이터베이스에 접속하는 방법은 단위 룰 시스템마다 다르게 설정할 수 있다. 업무 데이터베이스 접속 방법이 설정되지 않은 단위 룰 시스템이 하나라도 있는 경우 체크 표시가 된다.

다음은 각 작업 메뉴의 기능이다.

- ❒ 수정
  업무 데이터베이스를 추가하거나 수정한다. 수정 작업을 선택하고 이름이 비어 있지 않은 슬롯의 ID를 선택하면 그 업무 데이터베이스를 수정할 수 있으며, 이름이 비어 있는 슬롯의 ID를 선택하면 업무 데이터베이스를 추가할 수 있다.
- ❒ 삭제
  업무 데이터베이스에 대한 접속 설정을 제거한다. 단, 이 업무 데이터베이스에 접근하는 룰이 있는 경우 제거되지 않는다.

2.3.4.1 업무 데이터베이스 기본 정보

업무 데이터베이스의 기본 정보를 설정하는 단계이다

```
********************************************************************************
```

```
설정 유형 선택 -> 업무 데이터베이스 목록 -> 업무 데이터베이스 기본 정보

*******************************************************************************************

업무 데이터베이스의 기본 정보를 입력하십시오.

[필수] 이름 >>
[필수] 설명 >>
```

Figure 2 - 3 - 17

업무 데이터베이스의 기본 정보로는 이름과 설명이 있다. 이름은 60자 이내이어야 하며 다른 업무 데이터베이스의 이름과 중복이 되어서는 아니 된다.

기본 정보를 입력하면 2.3.4.2 업무 데이터베이스 기본 접속 정보 단계로 넘어간다

2.3.4.2 업무 데이터베이스 기본 접속 정보

이 단계에서는 각 단위 시스템 별 업무 데이터베이스 접속 정보를 조회하고 Figure 2-3-18과 같이 관리할 수 있다.

```
*******************************************************************************************

설정 유형 선택 -> 업무 데이터베이스 목록 -> 업무 데이터베이스 기본 접속 정보

*******************************************************************************************

단위 룰 시스템별 업무 데이터베이스 기본 접속 정보입니다. 수정할 시스템을 선택하십시오.

[항목들]
-------------------------------------------------------------------------------------
    | 단위 룰 시스템 | URL                                                | 계정      | 사용 룰 수 |
-------------------------------------------------------------------------------------
1 | DEV                | jdbc:oracle:thin:@bizdev:1521:ORCL | APPDBU   | 10           |
-------------------------------------------------------------------------------------
2 | TEST               | jdbc:oracle:thin:@biztest:1521:ORCL | APPDBU   | 8            |
-------------------------------------------------------------------------------------
 [가능한 작업들]
```

```
a. 수정      b. 제거

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 3 - 18

업무 데이터베이스 접속 정보는 시스템 별로 상이할 수 있다. 이 단계에서는 업무 데이터베이스에 대한 각 시스템 별 기본 접속 정보를 관리한다. 기본 접속 정보는 다양한 InnoRules 서비스들에서 사용한다.

목록의 각 필드의 의미는 다음과 같다.

- ❒ 단위 룰 시스템
  어느 단위 룰 시스템의 접속 정보인가를 나타낸다.
- ❒ URL
  업무 데이터베이스에 접속하기 위한 JDBC URL이다.
- ❒ 계정
  업무 데이터베이스에 접속하는 데에 사용될 데이터베이스 사용자 계정이다.
- ❒ 사용 룰 수
  이 단위 룰 시스템에서 이 업무 데이터베이스를 사용하는 룰이 몇 개나 되는가를 나타낸다.

다음은 각 작업 메뉴의 기능이다.

- ❒ 수정
  단위 룰 시스템에 대한 기본 접속 정보를 수정한다. 기존에 등록된 정보가 없다면 설정한다.
- ❒ 제거
  단위 룰 시스템에 대한 기본 접속 정보를 제거한다. 단위 룰 시스템에 이 접속 정보를 사용하는 룰이 있더라도 접속 정보는 제거 가능하다. 그러나, 기본 접속 정보를 사용하는 InnoRules 서비스 등에서는 에러가 발생할 수 있다.

2.3.4.3 업무 데이터베이스 서버 유형 선택

Figure 2-3-19는 업무 데이터베이스 서버의 유형을 선택하는 단계이다.

```
********************************************************************************

설정 유형 선택 -> 업무 데이터베이스 목록 -> 업무 데이터베이스 기본 접속 정보 -> 데이터베이스 서버 선택

********************************************************************************

단위 룰 시스템 DEV에서 업무 데이터베이스 1의 데이터베이스 서버 유형을 선택하십시오.
    1. Oracle
```

```
        Oracle database (with type 4 driver)
    2. MSSQL
        Microsoft SQL server
    3. DB2
        IBM DB2 with DB2 UDB JDBC universal driver
    4. MySQL
        MySQL database server
    5. Postgre
        Postgre database server


선택하십시오. >>
```

Figure 2 - 3 - 19

사용하고자 하는 데이터베이스 서버의 유형이 목록에 없다면 기술지원을 요청한다. 데이터베이스 서버를 선택하면 2.3.4.4 업무 데이터베이스 접속 정보 단계로 넘어간다.

2.3.4.4 업무 데이터베이스 접속 정보

Figure 2-3-20는 업무 데이터베이스의 접속 정보를 설정하는 단계이다.

```
********************************************************************************

설정 유형 선택 -> 업무 데이터베이스 목록 -> 업무 데이터베이스 기본 접속 정보 -> 업무 데이터베이스 접속 정보

********************************************************************************

단위 룰 시스템 DEVPART에서 업무 데이터베이스 1에 접속하기 위한 정보를 입력하십시오.

[필수] address >>
[필수] port(기본값: 1521) >>
[필수] SID or service name(기본값: ORCL) >>
[필수] 사용자 계정 >>
[필수] 비밀번호 >>

연결을 테스트하시겠습니까? (Yes/No) >>
```

Figure 2 - 3 - 20

데이터베이스에 따라 사용할 JDBC URL의 형식이 미리 지정되어 있으며, 이에 따라 필요한 정보를 사용자에게 질의하게 된다. 데이터베이스 서버에 따른 JDBC URL 형식은 lib/jdbc-drivers/database.xml에 정의되어 있다. 만약, 사용하고자 하는 URL의 형식이 이와 같지 않은 경우 기술지원을 요청한다.

입력이 완료되면 연결 테스트를 하고 테스트가 성공하면 설정이 변경되고 2.3.4.2 업무 데이터베이스 기본 접속 정보 단계로 돌아간다.

2.4 서비스 구성 마법사

InnoRules 서버는 룰 시스템 관리에 필요한 다양한 서비스를 제공한다. 서비스 구성 마법사는 마법사가 실행되는 로컬 설치 노드에 서비스들을 구성하도록 도와준다. 서비스 구성 마법사는 로컬 설치 노드의 서비스만을 구성하기 때문에 어떤 설치 노드에 서비스를 구성하려면 그 설치 노드에 있는 서비스 구성 마법사를 실행해야 한다.

서비스 구성 마법사를 이용하면 다음의 InnoRules 서비스를 구성할 수 있다.

- ❒ 관리 서비스, 관리 서비스에 대해서는 3.8.2 관리 서비스를 참고하라.
- ❒ 룰 빌더 서비스, 룰 빌더 서비스에 대해서는 3.8.3 룰 빌더 서비스를 참고하라.
- ❒ TCP 커넥터 서비스, TCP 커넥터 서비스에 대해서는 3.8.4 TCP 커넥터 서비스를 참고하라.
- ❒ 룰 REST 서비스, 룰 REST 서비스에 대해서는 3.8.5 룰 REST 서비스를 참고하라.

서비스 구성 마법사는 script 디렉터리에서 Figure 2-4-1와 같이 다음의 명령으로 실행한다.

```
UNIX 계열에서:
./service-wiz.sh
Windows에서:
service-wiz.bat
```

Figure 2 - 4 - 1

2.4.1 소개

Figure 2-4-2은 서비스 구성 마법사의 소개 단계를 보여준다.

```
********************************************************************************

소개

********************************************************************************

InnoRules 서비스 구성 마법사에 오신 것을 환영합니다. 이 마법사는 당신의 호스트에 다양한 서비스들을 설정하도록 도와줄 것입니다.
[엔터를 누르십시오]
```

Figure 2 - 4 - 2

2.4.2 서비스 유형 선택

Figure 2-4-3는 관리하고자 하는 서비스를 유형을 선택하는 단계이다. 하나의 설치 노드에 서로 다른 두 개 이상의 서비스를 활성화할 수도 있다.

```
********************************************************************************

서비스 유형 선택

********************************************************************************

설정하고자 하는 서비스의 유형을 선택하십시오. 종료하려면 엔터를 누르십시오.
    1. 관리 서비스
        동작 중인 룰 애플리케이션과 룰 시스템 컴포넌트들을 관리할 수 있습니다.
    2. 룰 빌더 서비스
        룰을 작성하고 관리할 수 있습니다.
    3. TCP 커넥터 서비스
        원격 어플리케이션이 룰 서비스를 호출할 수 있습니다.
    4. 룰 REST 서비스
        룰 REST 서비스를 설정하고 웹 환경에서 룰을 호출할 수 있습니다.

선택하십시오. >>
```

Figure 2 - 4 - 3

선택한 서비스에 따라 각각의 서비스 화면으로 이동한다.

2.4.3 관리 서비스

관리 서비스는 설정 클러스터에 속한 룰 애플리케이션의 상태를 모니터링하고 관리하는 서비스이다.

2.4.3.1 관리 서비스 목록

Figure 2-4-4은 설정 클러스터에 활성화된 관리 서비스의 목록을 보여주는 단계이다.

```
********************************************************************************

서비스 유형 선택 -> 관리 서비스

********************************************************************************

관리 서비스들에 대한 작업을 선택하십시오

[항목들]
```

```
------------------------------------------------------------------------
| 이름                                          | 바인딩 주소      | 포트    | 로컬 |
------------------------------------------------------------------------
1 | d205593b-d43c-47c1-9304-39b8b6b649f0 | 10.10.10.1   | 25811 |       |
------------------------------------------------------------------------

[가능한 작업들]
a. 위로 b. 아래로 c. 로컬 설치 노드의 관리 서비스 활성화

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 4 - 4

하나의 설정 클러스터에 관리 서비스를 여러 개 활성화할 수도 있다. 활성화된 서비스들은 Figure 2-4-4의 목록에 표시된다. 관리 서비스들에는 우선 순위가 부여된다. 목록에서 위쪽에 위치한 서비스가 우선 순위가 높다. 설정 클러스터의 InnoRules 서버와 룰 애플리케이션들은 먼저 우선 순위가 높은 관리 서비스에 접속을 시도한다. 관리 서비스의 우선 순위에 대한 자세한 사항은 3.8.2 관리 서비스를 참조하라.

목록의 각 필드의 의미는 다음과 같다.

- ❒ 이름
  관리 서버가 활성화된 InnoRules 서버의 이름으로 설치될 때 부여된 UUID이다.
- ❒ 바인딩 주소와 포트
  이 관리 서버에 접속하기 위한 IP 주소와 포트이다.
- ❒ 로컬
  로컬 설치 노드의 InnoRules인가 여부이다. 체크가 된 경우 로컬 설치 노드에의 활성화된 관리 서비스임을 의미한다.

다음은 각 작업 메뉴의 기능이다.

- ❒ 위로
  선택된 관리 서비스의 우선 순위를 한 단계 높인다.
- ❒ 아래로
  선택된 관리 서비스의 우선 순위를 한 단계 낮춘다.
- ❒ 로컬 설치 노드의 관리 서비스 활성화
  로컬 설치 노드에 관리 서비스를 활성화한다. 이 메뉴는 로컬 설치 노드에 관리 서비스가 비활성화 되어 있을 때만 나타난다.
- ❒ 로컬 설치 노드의 관리 서비스 비활성화
  로컬 설치 노드에 관리 서비스를 비활성화 한다. 이 메뉴는 로컬 설치 노드에 관리 서비스가 활성화되어 있을 때만 나타난다.

2.4.3.2 관리 서비스 접속 정보

Figure 2-4-5는 관리 서비스의 접속 정보를 설정하는 단계이다.

```
*******************************************************************************

서비스 유형 선택 -> 관리 서비스 -> 관리 서비스 접속 정보

*******************************************************************************

관리 서비스의 접속 정보를 입력하십시오.

[필수] 접속 주소
    1. 10.10.10.1
    2. 10.10.10.2
    3. 192.168.0.1
    4. 127.0.0.1
>>
[필수] 포트(기본값: 25811) >>
```

Figure 2 - 4 - 5

관리 서비스를 활성화하기 위해서는 클라이언트의 접속을 대기할 IP 주소와 포트를 지정해야 한다. 관리 서비스가 사용할 주소를 선택하되 클라이언트들이 식별하고 접속 가능한 IP 주소를 지정해야 한다. 포트의 기본값은 25811이며 시스템 관리자와 협의하여 할당을 받는 것이 좋다.

정보를 모두 정상적으로 입력하면 관리 서비스가 활성화되고 다시 Figure 2-4-6과 같이 관리 서비스 목록으로 돌아간다. 로컬 관리 서비스가 가장 아래에 추가된 것에 주목하라.

```
*******************************************************************************

서비스 유형 선택 -> 관리 서비스

*******************************************************************************

관리 서비스들에 대한 작업을 선택하십시오

[항목들]
-------------------------------------------------------------------------
| 이름                                    | 바인딩 주소      | 포트   | 로컬 |
```

```
------------------------------------------------------------------------
1 | d205593b-d43c-47c1-9304-39b8b6b649f0 | 10.10.10.1   | 25811 |         |
------------------------------------------------------------------------
2 | 911e97bf-3a84-4a76-bf80-4ecd221cdc68 | 10.10.10.2   | 25811 |    V    |
------------------------------------------------------------------------

[가능한 작업들]
a. 위로 b. 아래로 c. 로컬 설치 노드의 관리 서비스 비활성화

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 4 - 6

2.4.4 룰 빌더 서비스(Rule Builder Service)

룰 빌더 서비스는 룰 요소들을 저작 및 관리하는 서비스이다.

2.4.4.1 룰 빌더 서비스 목록

Figure 2-4-7은 룰 빌더 서비스 목록을 보여주는 단계이다.

```
*******************************************************************************

서비스 유형 선택 -> 빌더 서비스

*******************************************************************************

빌더 서비스들에 대한 작업을 선택하십시오.

[전역 설정]
    이관 전용        : 설정되지 않음
    평균 동시 사용자: 2
    최대 동시 사용자: 10
    암호화 방식    : SHA-256
    테이블 룰 조건식 체크    : 설정되지 않음
    클러스터 접속 타임 아웃(ms)   : 5,000
    클러스터 응답 타임 아웃(ms)    : 60,000
    클러스터 문자셋    : UTF-8
```

```
[항목들]
-------------------------------------------------------------------------------
|                  이름                  |      주소       |서비스 포트| 모니터 포트 | 로컬 |
-------------------------------------------------------------------------------
1 | d205593b-d43c-47c1-9304-39b8b6b649f0 | 10.10.10.1   | 25802 | 25810 |
-------------------------------------------------------------------------------

[가능한 작업들]
a. 위로      b. 아래로      c. 전역 설정 변경      d. 로컬 설치 노드의 빌더 서비스 비활성화      e. 빌더 서비스 접속 URL 확인
작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 4 - 7

전역 설정은 이 설정 클러스터에 활성화되는 모든 빌더 서비스에 적용되는 설정이다. 전역 설정 인자들에 대한 설명은 2.4.4.2 빌더 서비스 전역 설정을 참조하라.

하나의 설정 클러스터에 룰 빌더 서비스를 여러 개 활성화할 수도 있다. 활성화된 서비스들은 Figure 2-4-7의 목록에 표시된다. 이 룰 빌더 서버들에는 우선 순위가 부여되고 동작 중인 룰 빌더 서버들 중 가장 우선 순위가 높은 것만이 서비스를 제공한다. 이것을 룰 빌더 서버 클러스터링이라고 한다. 룰 빌더 서버 클러스터링에 대해서는 3.8.3.2 빌더 서버 클러스터링을 참조하라.

목록의 각 필드의 의미는 다음과 같다.

- ❒ 이름
  룰 빌더 서비스가 활성화된 InnoRules 서버의 이름으로 설치될 때 부여된 UUID이다.
- ❒ 주소
  이 룰 빌더 서비스에 접속하기 위한 IP 주소이다.
- ❒ 서비스 포트
  룰 빌더 서비스를 제공하는 포트이다. 룰 빌더 서비스는 HTTP를 이용하여 제공된다.
- ❒ 모니터 포트
  룰 변경에 대한 동기화를 하기 위한 모니터 포트이다.
- ❒ 로컬
  로컬 설치 노드의 InnoRules인가 여부이다. 체크가 된 경우 로컬 설치 노드에의 활성화된 관리 서비스임을 의미한다.

다음은 각 작업 메뉴의 기능이다.

- ❒ 위로
  선택된 룰 빌더 서비스의 우선 순위를 한 단계 높인다.
- ❒ 아래로
  선택된 룰 빌더 서비스의 우선 순위를 한 단계 낮춘다.
- ❒ 전역 설정 변경
  룰 빌더 서버의 전역 설정을 변경한다.

- ❒ 로컬 설치 노드의 룰 빌더 서비스 활성화
  로컬 설치 노드에 룰 빌더 서비스를 활성화한다. 이 메뉴는 로컬 설치 노드에 룰 빌더 서비스가 비활성화 되어 있을 때만 나타난다.
- ❒ 로컬 설치 노드의 룰 빌더 서비스 비활성화
  로컬 설치 노드에 룰 빌더 서비스를 비활성화 한다. 이 메뉴는 로컬 설치 노드에 룰 빌더 서비스가 활성화되어 있을 때만 나타난다.
- ❒ 빌더 서비스 접속 URL 확인
  선택된 룰 빌더 서비스의 접속 URL을 확인한다. 이 메뉴는 등록된 룰 빌더 서비스가 하나 이상 존재할 때만 나타난다.

2.4.4.2 빌더 서비스 전역 설정

Figure 2-4-8은 빌더 서비스의 전역 설정을 하는 단계를 보여준다.

```
********************************************************************************

서비스 유형 선택 -> 빌더 서비스 -> 빌더 서비스 전역 설정

********************************************************************************

빌더 서비스의 전역 설정 인자들의 값을 입력하십시오.

[필수] 이 클러스터의 빌더 서버를 이관 전용으로 설정하시겠습니까?
(*) 1. 이관 전용으로 설정하지 않음
    2. 이관 전용으로 설정
>>
[필수] 평균 동시 사용자(기본값: 2) >>
[필수] 최대 동시 사용자(기본값: 10) >>
[필수] 비밀번호 암호화 방식(기본값: SHA-256) >>
[필수] 테이블 룰 조건식 체크
(*) 1. 테이블 룰 조건식을 체크하지 않음
    2. 테이블 룰 조건식을 체크함
>>
[필수] 클러스터 접속 타임 아웃(ms)(기본값: 5000) >>
[필수] 클러스터 응답 타임 아웃(ms)(기본값: 60000) >>
[필수] 클러스터 문자셋(기본값: UTF-8) >>
```

Figure 2 - 4 - 8

빌더 서비스의 전역 설정 인자들은 다음과 같다.

- ❒ 이관 전용
  이관 전용으로 설정된 경우 이 설정 클러스터의 빌더 서비스들은 룰 저장 기능이 비활성화 되며 이관만 가능하다. 이관 전용에 대해서는 3.8.3.3 다단계 룰 시스템과 이관 전용 모드를 참조하라.
- ❒ 평균 동시 사용자와 최대 동시 사용자
  동시에 빌더 서비스를 이용하는 사용자의 수이다. 이는 동시에 실행되는 룰 빌더의 수가 아니라 동시에 호출되는 빌더 서비스의 수이다. InnoRules 서버는 이에 기반하여 로컬 룰 서버의 개수 등 성능/자원 파라미터를 조정한다.
- ❒ 비밀번호 암호화 방식
  사용자 비밀번호를 룰 DB에 저장할 때 사용되는 암호화 방식이다. 지원되는 암호화 알고리즘은 DES와 SHA256이 있으며 기본값은 SHA-256이다. 비밀번호 암호화 방식의 변경은 DES에서 SHA-256으로 변경만 가능하며 역방향은 불가능하므로 주의하라.
- ❒ 테이블 룰 조건식 체크
  테이블 룰에서 사용되는 조건식들의 정합성을 체크한다. 정합성 체크의 자세한 사양은 별첨된 "InnoRules 테이블 룰 조건식 체크" 문서를 참조하라.
- ❒ 클러스터 접속 타임 아웃
  빌더 클러스터 내의 빌더 서버 간의 접속 타임 아웃이다. 단위는 밀리 세컨드(millisecond) 이며 기본값은 5,000 이다. 빌더 서버 클러스터링에 대해서는 3.8.3.2 빌더 서버 클러스터링을 참고하라.
- ❒ 클러스터 응답 타임 아웃
  빌더 클러스터 내의 빌더 서버 간의 응답 타임 아웃이다. 단위는 밀리 세컨드(millisecond) 이며 기본값은 60,000 이다. 응답 타임 아웃의 값이 너무 작은 경우 정상적인 빌더 서버 처리 과정에서 타임 아웃으로 인한 오류가 발생할 수 있으므로 주의 해야 한다. 빌더 서버 클러스터링에 대해서는 3.8.3.2 빌더 서버 클러스터링을 참고하라.
- ❒ 클러스터 접속 문자셋
  빌더 클러스터 내의 빌더 서버 간의 전송 문자셋이다. 기본값은 UTF-8 이다. 빌더 서버 클러스터링에 대해서는 3.8.3.2 빌더 서버 클러스터링을 참고하라.

2.4.4.3 모니터 서버 접속 정보

모니터 서버의 접속 정보를 입력하는 단계이다.

```
********************************************************************************

서비스 유형 선택 -> 빌더 서비스 -> 모니터 서버 접속 정보

********************************************************************************

모니터 서버의 접속 정보를 선택하십시오.
```

```
[필수] 접속 주소
    1. 10.10.10.1
    2. 10.10.10.2
    3. 192.168.0.1
    4. 127.0.0.1
>>
[필수] 포트(기본값: 25810) >>
```

Figure 2 - 4 - 9

모니터 서버는 룰 빌더 서버의 일부 기능으로 룰이 변경될 때 룰 서버들이 변경 내역을 동기화할 수 있도록 해 주는 서버이다. 모니터 서버에 대해서는 3.8.3.1 모니터 서버와 모니터 세션을 참조하라.

같은 설정 클러스터의 룰 서버들이 이 모니터 서버에 접속해야 하므로 룰 서버들이 접속할 수 있는 주소를 선택한다.

모니터 서버 포트의 기본값은 25810이나 시스템 관리자와 협의하여 설정하도록 한다. 같은 설정 클러스터의 룰 서버들이 이 모니터 서버에 접속할 수 있도록 필요하다면 방화벽 설정을 해야 한다.

룰 빌더 서비스는 InnoRules 서버의 HTTP 통신 기능을 이용하므로 별도로 주소나 포트를 설정할 필요가 없다.

모니터 서버의 접속 정보를 설정하면 룰 빌더 서비스가 추가되고 2.4.4.1 룰 빌더 서비스 목록으로 이동한다.

2.4.4.4 룰 빌더 서비스 접속 URL 확인

룰 빌더 서비스를 사용하기 위한 접속 정보를 확인하는 단계이다.

```
빌더 서비스 'd205593b-d43c-47c1-9304-39b8b6b649f0'를 사용하기 위해서는 다음의 정보를 빌더 클라이언트에 입력하십시오.
단, 다음 정보의 변경(커스터마이징 등)이 있는 경우에 실제 접속 정보는 관리자에게 확인이 필요합니다.
        호스트    : 10.10.10.1
        URL       : /innorules/services/builder
        포트      : 25802
[엔터를 누르십시오]
```

정상적으로 활성화된 룰 빌더 서비스의 경우, 사용자는 접속 정보로 룰 빌더 서비스를 사용할 수 있다.

다만, 접속 정보에 영향을 주는 요소(커스터마이징, Web Server를 통한 접속 등)가 있다면, 실제 접속 정보는 해당 관리자에게 확인이 필요하다.

엔터를 누르면 2.4.4.1 룰 빌더 서비스 목록으로 이동한다.

2.4.5 TCP 커넥터 서비스

TCP 커넥터 서비스는 TCP/IP와 InnoRules 자체의 프로토콜을 이용하여 원격으로 룰 서비스를 호출할 수 있게 해 주는 서비스이다. TCP 커넥터 서비스에 대한 자세한 사항은 3.8.4 TCP 커넥터 서비스를 참조하라.

2.4.5.1 단위 룰 시스템 선택

TCP 커넥터 서비스를 활성화하기 위해서는 룰 서비스를 하고자 하는 단위 룰 시스템을 먼저 선택해야 한다. Figure 2-4-10은 단위 룰 시스템을 선택하는 단계를 보여준다.

```
********************************************************************************

서비스 유형 선택 -> 단위 룰 시스템 선택

********************************************************************************

TCP 커넥터 서비스를 이용하여 룰 서비스를 하고자 하는 단위 룰 시스템을 선택하십시오.
    1. DEV
        Development
    2. PROD
        Production


선택하십시오. >> 2
```

Figure 2 - 4 - 10

룰 빌더 서비스가 이관 전용으로 설정되어 있는 경우 가장 앞 단계의 단위 룰 시스템은 Figure 2-4-10의 목록에 보이지 않으며 선택할 수도 없다. 만약 선택할 수 있는 단위 룰 시스템이 없는 경우 다음의 경고 메시지가 출력된다.

```
사용할 수 있는 단위 룰 시스템이 없습니다. 단위 룰 시스템을 먼저 등록하십시오.
```

Figure 2 - 4 - 11

TCP 커넥터 서비스를 활성화할 단위 룰 시스템을 선택하면 2.4.5.2 TCP 커넥터 서버 그룹 목록으로 이동한다.

2.4.5.2 TCP 커넥터 서버 그룹 목록

Figure 2-4-12는 설정 클러스터에 활성화된 PROD 단위 룰 시스템의 TCP 커넥터 서버 그룹의 목록을 보여준다.

```
*******************************************************************************

서비스 유형 선택 -> TCP 커넥터 서버 그룹 목록

*******************************************************************************

단위 룰 시스템 'PROD'에 등록된 TCP 커넥터 서버 그룹 목록입니다.

[항목들]
-------------------------------------------------------------------------------
    | 이름    | 모드            | 서버 개수 | 로컬 우선 | 평균 동시 처리량 | 최대 동시 처리량 | 로컬 |
-------------------------------------------------------------------------------
  1 | online | load-balance | 1     | N           | 100                | 100                |       |
-------------------------------------------------------------------------------

[가능한 작업들]
a. 그룹 관리      b. 로컬 설치 노드에 TCP 커넥터 서비스 활성화

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 4 - 12

TCP 커넥터 서버들은 부하분산(Load Balancing) 또는 장애극복(Fail Over) 목적으로 그룹화된다. 이 그룹을 TCP 커넥터 서버 그룹이라고 한다. 이에 대한 자세한 사항은 3.8.4.2 TCP 커넥터 서버 그룹을 참조하라.

항목 목록의 각 필드의 의미는 다음과 같다.

- ❒ 이름
  TCP 커넥터 서버 그룹의 이름이다.
- ❒ 모드
  TCP 커넥터 서버 그룹의 고가용 정책이다. load-balance는 이 그룹이 부하분산을 목적으로 하고 있음을 의미하며 fail-over는 장애극복을 목적으로 하고 있음을 의미한다.
- ❒ 서버 개수
  몇 개의 서버가 이 그룹에 포함되어 있는가를 의미한다.
- ❒ 로컬 우선
  로컬 우선 옵션의 활성화 여부이다.

- ❒ 평균 동시 처리량 및 최대 동시 처리량
  TCP 서버 그룹이 처리할 룰 서비스 처리량의 예측 값이다.
- ❒ 로컬
  로컬 설치 노드의 TCP 커넥터 서버가 이 그룹에 속하는가를 나타낸다.

작업 메뉴는 다음과 같다.

- ❒ 그룹 관리
  선택한 TCP 서버 그룹의 설정을 변경한다.
- ❒ 로컬 설치 노드에 TCP 커넥터 서비스 활성화
  로컬 설치 노드에 TCP 커넥터 서비스를 활성화한다. 로컬 설치 노드에 TCP 커넥터 서비스가 비활성화 되어 있을 때만 나타난다.
- ❒ 로컬 설치 노드에 TCP 커넥터 서비스 비활성화
  로컬 설치 노드의 TCP 커넥터 서비스를 비활성화 한다. 로컬 설치 노드에 TCP 커넥터 서비스가 활성화되어 있을 때만 나타난다.

2.4.5.3 TCP 커넥터 서비스 접속 정보

TCP 커넥터 서비스 활성화 메뉴를 선택하면 Figure 2-4-13와 같은 TCP 커넥터 서비스 접속 정보 설정 단계가 나타난다.

```
*******************************************************************************

서비스 유형 선택 -> TCP 커넥터 서버 그룹 목록 -> TCP 커넥터 서비스 접속 정보

*******************************************************************************

TCP 커넥터 서비스의 접속 정보를 선택하십시오.

[필수] 접속 주소
    1. 10.10.10.1
    2. 10.10.10.2
    3. 192.168.0.1
    4. 127.0.0.1
>>
[필수] 포트 >>
[필수] 사용할 룰 DB를 선택하십시오.
    1. RULEDB1
    2. RULEDB2
>>
```

Figure 2 - 4 - 13

접속 주소와 포트는 원격의 룰 애플리케이션들이 이 TCP 커넥터 서버에 접속하는 데에 사용될 주소와 포트이다. 주소의 목록에는 이 호스트의 네트워크 주소들이 나열된다. 이 중 원격 애플리케이션이 접속 가능한 주소 중 하나를 선택하도록 한다. 포트는 시스템 관리자와 협의하여 부여하며 이 때 방화벽 설정도 같이 확인하도록 한다.

사용할 룰 DB에는 이 단위 시스템(이 예에서는 PROD)에 등록된 모든 룰 DB들이 나열되며 이 중 하나를 선택한다. TCP 커넥터 서버는 여기서 지정된 룰 DB로부터 룰 데이터를 조회한다. 데이터베이스의 부하 정도, 시스템의 요건 등을 고려하여 룰 DB를 선택한다.

접속 정보를 입력하면 2.4.5.4 TCP 서버 그룹 선택으로 이동한다.

2.4.5.4 TCP 서버 그룹 선택

Figure 2-4-14은 활성화하려는 TCP 커넥터 서버가 어떤 그룹에 속할 것인지를 결정하는 단계이다.

```
********************************************************************************

서비스 유형 선택 -> TCP 커넥터 서버 그룹 목록 -> TCP 커넥터 서버 그룹 선택

********************************************************************************

TCP 커넥터 서버를 포함시킬 그룹을 선택하십시오.
    1. online
    2. (새로운 그룹 생성)


선택하십시오. >>
```

Figure 2 - 4 - 14

목록에는 이 단위 시스템에 이미 생성된 그룹들이 보여지며 이 그룹들 중 하나를 선택할 수 있다. 또는 새로운 그룹을 생성할 수도 있다. 기존의 그룹을 선택하게 되면 TCP 커넥터 서버가 활성화 되고 2.4.5.2 TCP 커넥터 서버 그룹 목록으로 돌아가며, 새로운 그룹 생성을 선택하면 2.4.5.5 TCP 커넥터 서버 그룹 설정으로 이동한다.

2.4.5.5 TCP 커넥터 서버 그룹 설정

Figure 2-4-15은 TCP 커넥터 서버 그룹을 설정하는 단계를 보여준다.

```
********************************************************************************
```

```
서비스 유형 선택 -> TCP 커넥터 서버 그룹 목록 -> TCP 커넥터 서버 그룹 선택 -> TCP 커넥터 서버 그룹 설정

********************************************************************************

시스템 'PROD'에 추가할 TCP 커넥터 서버 그룹을 설정하십시오.

[필수] 그룹 이름 >> batch
[필수] 고가용 모드를 선택하십시오.
    1. 장애극복(FAIL OVER)
    2. 부하분산(LOAD BALANCE)
>> 2
[필수] 로컬 우선를 적용하시겠습니까?
    1. 예
    2. 아니오
>> 1
[필수] 평균 동시 처리량 >> 5
[필수] 최대 동시 처리량 >> 5
```

Figure 2 - 4 - 15

그룹의 이름은 이 단위 룰 시스템 내에서 그룹을 식별하는 데에 사용된다. 이 이름은 단위 룰 시스템 내에서 고유하여야 한다. 그룹 이름은 2.4.5.4 TCP 서버 그룹 선택에서 그룹 추가를 선택하여 들어왔을 때에만 입력 가능하다.

고가용 모드는 장애극복 또는 부하분산 중 하나를 선택할 수 있다. 고가용 모드에 대해서는 3.8.4.2 TCP 커넥터 서버 그룹을 참조하라.

로컬 우선 모드는 이 그룹을 사용하는 룰 애플리케이션들이 서버들의 우선 순위에 관계없이 애플리케이션과 동일한 호스트에서 동작하는 서버를 우선적으로 사용하도록 하는 옵션이다. 이 옵션에 대한 자세한 사항은 3.8.4.2 TCP 커넥터 서버 그룹을 참조하라.

평균 동시 처리량과 최대 동시 처리량은 이 그룹이 처리하게 될 예상 처리량이다. 예상 처리량과 고가용 모드에 따라 TCP 커넥터 서버가 점유할 자원의 양이 자동적으로 결정된다.

모든 정보가 입력되면 2.4.5.6 TCP 커넥터 서버 목록으로 이동한다.

2.4.5.6 TCP 커넥터 서버 목록

Figure 2-4-16은 TCP 커넥터 서버 그룹의 설정과 그룹에 속한 서버의 목록을 보여주는 단계이다.

```
********************************************************************************

서비스 유형 선택 -> TCP 커넥터 서버 그룹 목록 -> TCP 커넥터 서버 목록

********************************************************************************

단위 룰 시스템 'PROD'의 TCP 커넥터 서버 그룹 'online'의 설정과 서버의 목록은 다음과 같습니다.
모드: load-balance
로컬 우선: N
평균 동시 처리량: 100
최대 동시 처리량: 100



[항목들]
--------------------------------------------------------------------------
    | UUID                                   | 접속 주소  | 포트  | 로컬 |
--------------------------------------------------------------------------
  1 | 72631759-9cb6-4c33-a2ad-ba9f4d06a2cc | 10.10.10.2 | 25800 |      |
--------------------------------------------------------------------------
  2 | 4451bb87-facf-4768-87ee-fa075333311e | 10.10.10.1 | 25811 | V    |
--------------------------------------------------------------------------

[가능한 작업들]
a. 그룹 설정 변경     b. 위로     c. 아래로

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 4 - 16

Figure 2-4-16은 단위 룰 시스템 PROD에 등록된 online이라는 서버 그룹을 보여주고 있다. 화면의 상단에서는 그룹의 고가용 설정 및 동시 처리량을 보여준다.

목록에는 이 그룹에 속한 TCP 커넥터 서버들을 보여준다. 목록의 각 필드의 의미는 다음과 같다.

- ❒ UUID
  TCP 커넥터 서버가 활성화된 InnoRules 서버의 이름으로 설치될 때 부여된 UUID이다.
- ❒ 접속 주소와 포트
  이 TCP 커넥터 서버에 접속하기 위한 IP 주소와 포트이다.
- ❒ 로컬
  로컬 설치 노드의 InnoRules인가 여부이다. 체크가 된 경우 로컬 설치 노드에의 활성화된 관리 서비스임을 의미한다.

작업 메뉴는 다음과 같다.

- ❒ 그룹 설정 변경
  TCP 커넥터 서버 그룹의 설정을 변경한다.
- ❒ 위로/아래로
  선택된 TCP 커넥터 서버의 순서를 위 또는 아래로 이동한다. 순서는 고가용 모드에서 중요하게 사용된다.

2.4.6 룰 REST 서비스

룰 REST 서비스는 REST를 이용하여 룰 서비스를 호출할 수 있도록 해 주는 서비스이다. 룰 REST 서비스에 대한 자세한 사항은 3.8.5 룰 REST 서비스를 참조하라.

2.4.6.1 룰 REST 서비스

Figure 2-4-22는 등록된 단위 룰 시스템의 목록과 각 단위 시스템에 룰 REST 서비스가 활성화되어 있는가를 보여주는 단계이다.

```
********************************************************************************

서비스 유형 선택 -> 룰 REST 서비스

********************************************************************************

룰 REST 서비스들에 대한 작업을 선택하십시오.

[항목들]
-----------------------------------------------------------
    | 시스템 ID | 시스템 이름 | 별칭 | 활성화 여부 | 로컬 |
-----------------------------------------------------------
  1 | DEV       | 개발계      | N/A  |             |      |
-----------------------------------------------------------
  2 | TEST      | 테스트계    | N/A  |             |      |
-----------------------------------------------------------

[가능한 작업들]
a. 룰 REST 서비스 URL 조회     b. 별칭 변경     c. 로컬 설치 노드에 룰 REST 서비스 활성화

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 4 - 17

목록의 각 필드의 의미는 다음과 같다.

- ❒ 시스템 ID와 시스템 이름
  등록된 단위 룰 시스템들의 ID와 이름이다. 각 단위 룰 시스템 별로 룰 REST 서비스를 활성화 할지 지정할 수 있다.
- ❒ 별칭
  룰 REST 서비스를 이용하여 룰을 호출할 때 어떤 단위 시스템의 룰을 호출할 것인지를 지정하기 위해 별칭이 URL에 삽입된다.
- ❒ 활성화 여부
  로컬 설치 노드를 포함하여 설정 클러스터 내에 이 단위 룰 시스템을 위한 룰 REST 서비스가 활성화되어 있는가를 나타낸다.
- ❒ 로컬
  로컬 설치 노드에 이 단위 룰 시스템을 위한 룰 REST 서비스가 활성화되어 있는가를 나타낸다.

작업 메뉴는 다음과 같다.

- ❒ 룰 REST 서비스 URL 조회
  단위 룰 시스템의 룰 REST 서비스를 호출할 수 있는 URL들의 목록을 보여준다. 룰 REST 서비스를 호출하려는 애플리케이션에 이 URL을 제공할 수 있다.
- ❒ 별칭 변경
  별칭을 변경한다.
- ❒ 로컬 설치 노드에 룰 REST 서비스 활성화
  로컬 설치 노드의 InnoRules 서버에 룰 REST 서비스를 활성화 또는 비활성화 한다.

2.4.6.2 룰 REST 서비스 시스템 별칭 변경

하나의 InnoRules 서버가 여러 단위 룰 시스템의 룰을 REST 서비스할 수 있기 때문에 URL에 단위 룰 시스템을 지정할 수 있어야 한다. 단위 룰 시스템의 ID를 사용할 경우 URL에 시스템의 ID가 노출되므로 이를 피할 수 있도록 단위 룰 시스템마다 별칭을 부여할 수 있다. 이 별칭을 룰 REST 서비스 시스템 별칭이라고 한다. 룰 REST 서비스 URL은 다음의 형식을 갖는다.

http://[IP]:[PORT]/innorules/services/rest/[Alias]/[RULECODE]/[DATE]

여기서, IP와 PORT는 InnoRules 서버의 주소와 서비스 포트이며 Alias는 룰 REST 서비스 시스템 별칭이다. RULECODE 은 호출 대상 룰 코드이다. DATE는 룰 실행 기준 일자이며 생략 가능하다.
별칭은 10자 이내의 영문자, 숫자 또는 언더스코어로 이루어져야 한다.

Figure 2-4-23은 별칭을 지정하는 단계를 보여준다.

```
********************************************************************************

서비스 유형 선택 -> 룰 REST 서비스 -> 룰 REST 서비스 시스템 별칭 변경

********************************************************************************
```

```
단위 룰 시스템 'DEV'에 대한 별칭을 변경합니다.(기본값: DEV)
 >>
```

Figure 2 - 4 - 18

기본값으로는 기존에 지정된 별칭이 설정된다. 별칭은 다른 단위 시스템의 별칭과 중복될 수 없다. 설정을 완료하면 2.4.6.1 룰 REST 서비스로 이동한다.

2.4.6.3 룰 REST 서비스 활성화

하나의 InnoRules 서버가 여러 단위 룰 시스템에 대해서 룰 REST 서비스를 할 수 있다. Figure 2-4-24 은 REST 서비스를 활성화할 단위 룰 시스템을 선택하는 단계를 보여준다.

```
*******************************************************************************

서비스 유형 선택 -> 룰 REST 서비스 -> 룰 REST 서비스 활성화

*******************************************************************************

룰 REST 서비스를 활성화할 시스템을 선택하십시오. 여기서 선택되지 않는 단위 룰 시스템에 대해서는 룰 REST
서비스가 비활성화 됩니다.
아무것도 선택하지 않을 경우 로컬 설치 노드의 모든 룰 REST 서비스가 비활성화 됩니다.
(*) 1. DEV
        개발계
    2. TEST
        테스트계

다중 선택하십시오. 선택을 끝내려면 '0'을 입력하십시오. >>
```

Figure 2 - 4 - 19

서비스를 활성화할 단위 시스템의 번호를 한 번에 한 개씩 입력한다. 선택이 완료되면 0을 입력한다. REST 서비스를 비활성화하려면 단위 시스템의 번호를 입력하지 말고 바로 0을 입력한다. 만약 활성화된 단위 시스템을 변경하기 않으려면 아무 것도 입력하지 말고 엔터를 누른다. 현재 활성화된 단위 룰 시스템은 번호 앞에 (*) 표시가 되어 있다.

활성화할 단위 시스템이 있는 경우 2.4.6.4 룰 REST 서비스 설정으로 이동하며, 그렇지 않은 경우 2.4.6.1 룰 REST 서비스로 이동하게 된다.

2.4.6.4 룰 REST 서비스 설정

Figure 2-4-25는 룰 REST 서비스 URL에 보일 IP 주소를 선택하는 단계를 보여준다. 이 IP는 2.4.6.1 룰 REST 서비스의 룰 REST 서비스 URL을 생성하는 데에 사용된다.

```
*******************************************************************************

서비스 유형 선택 -> 룰 REST 서비스 -> 룰 REST 서비스 활성화 -> 룰 REST 서비스 설정

*******************************************************************************

룰 REST 서비스의 정보를 입력하십시오.

[필수] 룰 REST 서비스에 접속할 주소를 선택하십시오.
    1. 10.10.10.1
    2. 10.10.10.2
    3. 192.168.0.1
    4. 127.0.0.1
>>
```

Figure 2 - 4 - 20

룰 REST 서비스는 InnoRules의 HTTP 서비스를 이용하여 서비스되므로 InnoRules 서버가 바인딩한 모든 IP 주소들을 이용하여 접근이 가능하다. 대개 InnoRules 서버는 Figure 2-4-25의 리스트의 모든 IP 주소를 바인딩하여 실행될 것이므로 IP 주소 선택이 웹 서비스의 동작에 영향을 미치지는 않는다. 그러나, 선택된 주소가 URL에 반영이 되고 이 URL을 애플리케이션 개발자에게 전달해야 하므로 REST 서비스를 호출하려는 애플리케이션에서 접근이 가능한 IP 주소를 선택하도록 한다.
IP 주소를 선택하면 2.4.6.5 시스템 별칭 설정으로 이동한다.

2.4.6.5 시스템 별칭 설정

Figure 2-4-26은 2.4.6.3 룰 REST 서비스 활성화에서 선택한 단위 시스템에 대한 룰 REST 서비스 시스템 별칭을 설정하는 단계이다.

```
*******************************************************************************

서비스 유형 선택 -> 룰 REST 서비스 -> 룰 REST 서비스 활성화 -> 룰 REST 서비스 시스템 별칭 설정

*******************************************************************************
```

```
선택된 룰 REST 서비스 시스템의 별칭을 설정하십시오.

[필수] DEV(기본값: DEV) >>
[필수] TEST(기본값: TEST) >>
```

Figure 2 - 4 - 21

별칭은 각 단위 룰 시스템 별로 설정한다. 별칭을 설정하는 방법은 2.4.6.2 룰 REST 서비스 시스템 별칭 변경과 동일하다. 별칭 설정이 완료되면 2.4.6.1 룰 REST 서비스로 이동한다.

2.5 룰 애플리케이션 구성 마법사

룰 애플리케이션은 룰 서비스를 호출하는 애플리케이션이다. 애플리케이션이 룰 서버로 룰 서비스를 요청하게 된다. 룰 서버는 TCP 커넥터 서버와 같이 원격지에 있을 수도 있고 애플리케이션 내부에 위치할 수도 있다. 어느 경우든 룰 서비스를 호출하기 위해서는 애플리케이션에 룰 서비스를 호출할 수 있는 자원들이 만들어져야 한다. 이 자원들에는 TCP 커넥터 서버에 연결할 수 있는 커넥션 풀 또는 로컬 룰 서버 및 룰 DB 커넥션 풀 등이 포함될 수 있다.

룰 애플리케이션 구성 마법사는 애플리케이션이 어떤 방식으로 룰 서비스를 호출할지, 어떤 자원을 애플리케이션 내부에 구성할지를 설정하게 해 주는 마법사이다.

룰 애플리케이션 마법사는 룰 애플리케이션을 구성하는 마법사이다. script 디렉터리에서 Figure-2-5-1 과같이 다음의 명령으로 실행한다.

```
UNIX 계열에서:
./runtime-wiz.sh
Windows에서:
runtime-wiz.bat
```

Figure 2 - 5 - 1

2.5.1 소개

Figure 2-5-2는 룰 애플리케이션 구성 마법사의 소개 단계이다.

```
*******************************************************************************

소개

*******************************************************************************

InnoRules 룰 애플리케이션 설정 마법사에 오신 것을 환영합니다. 이 마법사를 이용하여 룰 애플리케이션 설정을
```

```
관리할 수 있습니다.
[엔터를 누르십시오]
```

Figure 2 - 5 - 2

2.5.2 룰 시스템 선택

애플리케이션이 어떤 단위 시스템의 룰 서비스를 호출할 것인가를 선택하는 단계이다. 하나의 애플리케이션은 하나의 단위 시스템의 룰들만 호출할 수 있다.

```
*******************************************************************************

룰 시스템 선택

*******************************************************************************

룰 시스템을 선택하십시오.
    1. DEV
        Development
    2. PROD
        Production

선택하십시오. >> 2
```

Figure 2 - 5 - 3

2.5.3 룰 애플리케이션 설정

Figure 2-5-4는 선택된 단위 룰 시스템에 등록된 룰 애플리케이션 설정들의 목록을 보여준다.

```
*******************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정

*******************************************************************************

단위 룰 시스템 PROD의 룰 애플리케이션을 설정합니다. 작업을 선택하십시오.

[항목들]
-------------------------------------------------------------------------------
    | 설정 이름  | 룰 서비스 유형 | DB      | 로그 수준 | 설명                    |
-------------------------------------------------------------------------------
```

```
  1 | PROD-Online | local              | RULEDB | INFO        | Rule Service for online App |
--------------------------------------------------------------------------------


[가능한 작업들]
a. 추가      b. 삭제      c. 수정      d. 상세정보


작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 5 - 4

하나의 룰 애플리케이션 설정을 여러 애플리케이션이 사용할 수 있다. 같은 설정을 이용하는 애플리케이션들은 같은 방식으로 룰 서비스를 호출하게 된다.

목록의 각 필드의 의미는 다음과 같다.

- ❒ 설정 이름
  룰 애플리케이션 설정의 이름이다. 애플리케이션이 어떤 설정을 이용할 것인지 지정하는 데에 사용된다. 설정의 이름은 단위 룰 시스템에 서 고유해야 한다.
- ❒ 룰 서비스 유형
  로컬 룰 서버를 이용할 것인가 원격 룰 서버를 이용할 것인가를 나타낸다.
- ❒ DB
  룰 서비스 유형이 로컬 룰 서버인 경우 서버가 사용하는 룰 DB의 이름이다.
- ❒ 로그 수준
  룰 애플리케이션의 로그 수준이다.
- ❒ 설명
  이 설정에 대한 간략한 설명이다.

작업 메뉴는 다음과 같다.

- ❒ 추가
  룰 애플리케이션 설정을 추가한다.
- ❒ 삭제
  룰 애플리케이션 설정을 삭제한다. 만약 이 설정을 사용하는 애플리케이션이 있다면 에러가 발생할 것이므로 이 설정이 사용되고 있는지 면밀히 확인을 해야 한다.
- ❒ 수정
  룰 애플리케이션 설정을 변경한다.
- ❒ 상세정보
  룰 애플리케이션 설정을 요약하여 보여준다.

2.5.4 룰 애플리케이션 설정의 기본 정보

Figure 2-5-5는 룰 애플리케이션 설정의 기본 정보를 입력 받는 단계이다.

```
********************************************************************************
```

```
룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 애플리케이션 설정의 기본 정보

********************************************************************************

룰 애플리케이션 설정의 기본 정보를 입력하십시오.

[필수] 룰 애플리케이션 설정 이름 >> online-webapp
[선택] 룰 애플리케이션 설정에 대한 설명 >> Rule applicaton configuration for an online web application.
[필수] 로그 수준
    1. ERROR
    2. WARN
(*) 3. INFO
    4. DEBUG
    5. TRACE
>> 4
[선택] 로그 태그를 지정하십시오. 룰 애플리케이션의 시스템 프로퍼티를 사용하여 태그를 지정하기 위해서
${system-property}를 이용할 수 있습니다.  >> ${container.name}
```

Figure 2 - 5 - 5

입력을 해야 할 기본 정보는 다음과 같다.

- ❒ 룰 애플리케이션 설정 이름
- ❒ 룰 애플리케이션 설정에 대한 설명
- ❒ 로그 수준

  룰 애플리케이션 전체의 로그 수준을 의미하는 것은 아니며, 룰 서비스 호출과 관계된 부분의 로그를 의미한다. 각 로그 수준별로 기록되는 로그는 다음과 같다.

  - ○ ERROR

    룰 시스템과 관련된 중대한 장애
  - ○ WARN

    룰 서비스 호출이 에러를 반환한 경우. 입력 값의 오류 등 비즈니스 로직 레벨의 오류로 인한 룰 서비스 호출 실패
  - ○ INFO

    룰 시스템의 주요한 작업 로그. 룰 서버 초기화, 종료 등
  - ○ DEBUG

    룰 서비스 호출, 업무 데이터베이스 쿼리 등에 소요된 시간 등
  - ○ TRACE

    룰 시스템 레벨에서 실행 경로 추적을 위한 로그

  ERROR로 갈수록 로그의 수준이 높고 TRACE로 갈수로 로그의 수준이 낮다. 로그의 수준이 낮을수록 기록되는 로그의 양이 많아지고 실행 성능은 저하된다.

❒ 로그 태그

단위 룰 시스템에 룰 애플리케이션 설정이 여러 개가 될 수 있기 때문에 각각의 로그 파일을 분리해야 할 필요가 있다. 로그 태그를 지정하면 기본 로그 디렉터리의 하위에 디렉터리를 생성하고(디렉터리의 이름은 로그 태그와 같다) 그 하위에 룰 애플리케이션의 로그를 기록한다. 로그 태그에 ${시스템 프로퍼티} 형식의 문자열을 포함시킬 수 있다. 그럴 경우 이 문자열을 지정된 시스템 프로퍼티의 값으로 변환하여 로그 태그로 사용한다. 이 방법은 하나의 룰 애플리케이션 설정을 여러 룰 애플리케이션이 이용할 때 로그 디렉터리를 분리하기 위해 사용할 수 있다. Figure 2-5-5의 예에서는 룰 애플리케이션의 container.name 시스템 프로퍼티의 값이 로그 태그로 사용될 것이다.

룰 애플리케이션 설정의 기본 정보를 입력하면 2.5.5 룰 서비스 유형으로 이동한다.

2.5.5 룰 서비스 유형

Figure 2-5-6은 룰 애플리케이션이 어떤 유형의 룰 서버를 사용할 것인가를 지정하는 단계이다.

```
*******************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형

*******************************************************************************

룰 서비스유형을 선택하십시오.
    1. 로컬 룰 서비스
        룰 애플리케이션의 JVM 내에 룰 서버를 실행합니다. 자바 메소드 호출을 이
        용하여 룰 서비스가 호출됩니다.
    2. 원격 룰 서비스
        원격의 TCP 커넥터 서버에 룰 서비스를 요청합니다. 서비스 호출에는 TCP/I
        P 통신이 사용됩니다.



선택하십시오. >> 1
```

Figure 2 - 5 - 6

로컬 룰 서비스를 선택할 경우 룰 애플리케이션 안에 로컬 룰 서버가 실행되고 이 서버로 서비스를 호출하게 된다. 같은 JVM 내에 룰 서버가 위치하기 때문에 자바 메소드 호출을 이용하여 서비스를 호출하게 되고 따라서 성능이 빠르다. 그러나, 로컬 룰 서버가 애플리케이션 JVM의 자원을 사용하므로 자원 사용량이 늘어나게 된다. 로컬 룰 서비스를 선택하면 2.5.6 룰 DB 접속 설정으로 이동한다.

원격 룰 서비스를 선택할 경우 TCP 커넥터 서버로 룰 서비스 호출을 하게 되고 애플리케이션은 TCP

커넥터 서버로의 TCP/IP 연결만을 관리한다. 기동 시간이 짧고 사용하는 자원의 양도 적은 장점이 있는 반면에 TCP/IP 통신으로 인한 속도 저하가 발생할 수도 있다. 원격 룰 서비스를 선택하면 2.5.13 원격 룰 서비스 유형 설정으로 이동한다.

2.5.6 룰 DB 접속 설정

Figure 2-5-7은 로컬 룰 서버가 사용할 룰 DB 커넥션 풀을 지정하는 단계를 보여준다.

```
*******************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형 -> 룰 DB 접속 설정

*******************************************************************************

로컬 룰 서버가 룰 DB에 대한 연결을 얻어오기 위해 어떤 방식을 사용할지 지정하십시오.
1. 데이터소스 참조 방식
        로컬 룰 서버가 미리 만들어져 있는 데이터소스로부터 룰 DB 연결을 가져오도록 합니다.
2. 전용 커넥션 풀 방식
        로컬 룰 서버가 전용으로 사용할 커넥션 풀 데이터소스를 로컬 생성합니다.



선택하십시오. >>
```

Figure 2 - 5 - 7

많은 경우에서 정책적으로 DB 커넥션 풀과 같은 자원은 애플리케이션 프레임워크가 관리하고 애플리케이션은 이를 참조만 할 수 있다. 대표적인 예로 웹 애플리케이션 환경에서 WAS가 커넥션 풀을 관리하고 웹 애플리케이션이 이를 참조하는 것이 있다.
데이터소스 참조 방식은 로컬 룰 서버가 JNDI를 이용하여 프레임워크가 제공하는 데이터소스를 찾아 룰 DB로 이용하도록 하는 방식이다. 이 방식을 이용하기 위해서는 프레임워크 담당자와 협의하여 미리 프레임워크에 룰 DB에 대한 데이터소스를 생성해 놓아야 한다. 이 방식을 선택하면 2.5.7 룰 DB에 대한 데이터소스(데이터소스 참조 방식)로 이동한다.

시스템 구성 마법사에서 단위 룰 시스템의 룰 DB에 대한 접속 방법을 기술하였다. 전용 커넥션 풀 방식은 이 접속 정보를 이용하여 커넥션 풀을 생성하고 로컬 룰 서버가 이를 이용하는 방식이다. 이 방식을 선택하면 2.5.8 룰 DB에 대한 데이터소스(전용 커넥션 풀 방식)로 이동한다.

2.5.7 룰 DB에 대한 데이터소스(데이터소스 참조 방식)

Figure 2-5-8은 데이터소스 참조 이름을 선택하는 단계이다.

```
*******************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형 -> 룰 DB에 대한 데이터소스

*******************************************************************************

룰 서버가 참조할 데이터소스의 이름을 입력하십시오.

 >> RULEDB
```

Figure 2 - 5 - 8

프레임워크에 등록된 룰 DB에 대한 데이터소스 이름을 입력한다. 데이터소스의 이름이 유효한지 마법사는 확인할 수가 없다. 잘못된 이름을 입력하는 경우 룰 애플리케이션이 초기화될 때 에러가 발생한다. 따라서, 이름을 입력할 때 주의를 기울여야 한다.

웹 애플리케이션의 경우 web.xml의 resource-ref에 등록된 자원을 찾기 위해서는 자원 참조 이름 앞에 java:comp/env/를 붙여줘야 한다. 이에 대해서는 서블릿 명세를 참조하라.

데이터소스 이름이 입력 완료되면 2.5.9 업무 데이터베이스 목록으로 이동한다.

2.5.8 룰 DB에 대한 데이터소스(전용 커넥션 풀 방식)

Figure 2-5-9는 룰 DB에 대한 커넥션 풀 생성을 위한 접속 정보를 선택하는 단계를 보여준다.

```
*******************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형 -> 룰 DB에 대한 데이터소스

*******************************************************************************

룰 서버가 어떤 룰 DB에 대한 커넥션 풀을 생성하도록 하시겠습니까?
    1. RULEDB1
        First Production Rule DB
    2. RULEDB2
        Second Production Rule DB


선택하십시오. >> 2
```

Figure 2 - 5 - 9

Figure 2-5-9의 단위 룰 시스템에는 두 개의 룰 DB, RULEDB1과 RULEDB2가 등록되어 있다. 룰 서버는 선택된 DB의 접속 정보를 이용하여 룰 애플리케이션 안에 룰 DB에 대한 커넥션 풀을 만들고 이를 이용하여 룰 데이터를 접근하게 된다.

룰 DB를 선택하면 2.5.9 업무 데이터베이스 목록으로 이동한다.

2.5.9 업무 데이터베이스 목록

시스템 구성 마법사에서 룰 시스템이 접근하는 업무 데이터베이스에 대해 정의하고 접속 정보도 지정을 하였다. 룰 애플리케이션들은 다양한 이유로 하여 이 접속 정보를 이용할 수 없는 경우가 있다. 또한, 애플리케이션이 호출하는 룰 서비스들이 모든 업무 데이터베이스를 접근하는 것도 아니다.

이 단계에서는 룰 애플리케이션이 어떤 업무 데이터베이스를 접근할 것인지, 그리고 어떤 방식으로 접근할 것인지에 대해서 지정한다.

```
*******************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형 -> 업무 데이터베이스 목록

*******************************************************************************

접속 방식을 설정할 업무 데이터베이스를 선택하십시오.

[항목들]
-----------------------------------------------------------------
    | ID | 이름       | 접속 방식       | 설명                          |
-----------------------------------------------------------------
  1 | 1  | CUSTOMER | 설정되지 않음    | Customer Database           |
-----------------------------------------------------------------
  2 | 2  | PFDB       | 설정되지 않음    | Product Factory Database |
-----------------------------------------------------------------

[가능한 작업들]
a. 편집

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 5 - 10

Figure 2-5-10은 업무 데이터베이스의 목록을 보여준다. 업무 데이터베이스들은 시스템 구성 마법사에서 등록했던 것들이다. 목록의 각 필드의 의미는 다음과 같다.

- ❒ ID, 이름, 설명
  시스템 구성 마법사에서 등록했던 업무 데이터베이스의 ID, 이름과 설명이다.
- ❒ 접속 방식
  어떤 방식으로 업무 데이터베이스에 접근할 것인가에 대한 방법이다. 접속 방식의 종류는 2.5.10 업무 데이터베이스 접속 방식 선택에 나와 있으며 각 방식에 대한 보다 자세한 내용은 3.6.8 업무 DB 접속 방법 설정을 참조하라. 최초에는 설정되지 않음으로 표시되어 있다.

편집을 선택하고 편집하고자 하는 업무 데이터베이스의 ID를 입력하면 2.5.10 업무 데이터베이스 접속 방식 선택으로 이동한다.
작업을 선택하지 않고 엔터를 누르면 2.5.14 룰 서비스 동시 사용자 수 설정으로 이동한다.

2.5.10 업무 데이터베이스 접속 방식 선택

Figure 2-5-11은 업무데이터베이스에 대한 접속 방식을 선택하는 단계를 보여준다.

```
********************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형 -> 업무 데이터베이스 목록 -> 업무 데이터베이
스 접속 방식 선택

********************************************************************************

룰 서버가 어떤 방식으로 업무 데이터베이스에 접속할지 선택하십시오.
(*) 1. 설정되지 않음
        이 설정을 사용하는 룰 애플리케이션은 이 업무 데이터베이스를 사용하지
        않습니다.
    2. 전용 커넥션 풀(기본접속정보)
        시스템 마법사에서 등록한 기본 접속 정보를 사용하여 업무 데이터베이스에
         대한 전용 커넥션 풀을 생성합니다.
    3. 전용 커넥션 풀(접속정보 지정)
        데이터베이스 접속 정보를 지정하여 업무 데이터베이스에 대한 전용 커넥션
         풀을 생성합니다.
    4. 데이터소스 참조
        기존에 생성된 데이터소스를 이용하여 업무 데이터베이스를 접근합니다.


선택하십시오. >>
```

Figure 2 - 5 - 11

다음은 각 접속 방식에 대한 설명이다.

- ❒ 설정되지 않음
  업무 데이터베이스의 접속 방식을 설정하지 않는다. 룰 애플리케이션이 호출할 룰 서비스들이 이 업무 데이터베이스를 접근하지 않을 경우에 선택한다. 설정되지 않았음에도 불구하고 이 업무 데이터베이스를 접근하는 룰 서비스를 호출한다면 룰 서비스 호출 에러가 발생할 것이다. 이 방식을 지정하면 2.5.9 업무 데이터베이스 목록으로 돌아간다.
- ❒ 전용 커넥션 풀(기본접속정보)
  룰 애플리케이션 안에 커넥션 풀을 생성하여 이를 이용한다. 시스템 구성 마법사에서 등록하였던, 이 단위 룰 시스템에서의 이 업무 데이터베이스의 기본 접속 정보를 접속 정보로 이용한다.
  이 방식을 지정하면 추가적인 정보는 묻지 않고 2.5.9 업무 데이터베이스 목록으로 돌아간다.
- ❒ 전용 커넥션 풀(접속정보 지정)
  룰 애플리케이션 안에 커넥션 풀을 생성하여 이를 이용한다. 접속 정보는 별도로 지정한다. 업무 데이터베이스가 미러링 되어 있고 룰 애플리케이션이 미러링 되어 있는 데이터베이스를 접근해야 할 경우에 이 방식을 이용할 수 있다. 이 방식을 지정하면 2.5.11 업무 데이터베이스 접속 정보(전용 커넥션 풀 접속 정보 지정)로 이동한다.
- ❒ 데이터소스 참조
  프레임워크에서 관리되는 데이터소스를 이용하여 업무 데이터베이스에 접근할 경우 이 방식을 이용한다. 이 방식을 지정하면 2.5.12 업무 데이터베이스 접속 정보(데이터소스 참조 방식)로 이동한다.

2.5.11 업무 데이터베이스 접속 정보(전용 커넥션 풀 접속 정보 지정)

Figure 2-5-12는 전용 커넥션 풀 (접속 정보 지정) 방식을 선택했을 때 접속 정보를 지정하는 단계를 보여준다.

```
********************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형 -> 업무 데이터베이스 목록 -> 업무 데이터베이스 접속 방식 선택 -> 업무 데이터베이스 접속 정보

********************************************************************************

업무 데이터베이스의 접속 정보를 입력하십시오.

[필수] JDBC 드라이버 클래스 >> oracle.jdbc.driver.OracleDriver
[필수] JDBC URL >> jdbc:oracle:thin:@batdb:1521:orcl
[필수] 데이터베이스 계정 >> customer
[필수] 계정 비밀번호 >>
```

Figure 2 - 5 - 12

여기서 사용되는 JDBC 드라이버의 라이브러리는 InnoRules 설치 디렉터리가 아닌 룰 애플리케이션의 클래스 경로에 위치해야 한다.

접속 정보 지정이 완료되면 2.5.9 업무 데이터베이스 목록으로 돌아간다.

2.5.12 업무 데이터베이스 접속 정보(데이터소스 참조 방식)

Figure 2-5-13은 데이터소스 참조 방식을 선택했을 때 데이터소스의 이름을 지정하는 단계를 보여준다.

```
*******************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형 -> 업무 데이터베이스 목록 -> 업무 데이터베이스 접속 방식 선택 -> 업무 데이터베이스 접속 정보

*******************************************************************************

업무 데이터베이스 연결을 제공할 데이터소스의 정보를 입력하십시오.

[필수] 데이터소스 이름 >> DATASOURCE
[선택] 이니셜 팩토리 클래스 >>
[선택] 프로바이더 URL >>
```

Figure 2 - 5 - 13

룰 애플리케이션이 웹 애플리케이션인 경우, resource-ref에 등록된 자원을 사용하려 한다면 데이터소스 이름 앞에 java:comp/env/를 붙여줘야 한다. 또한, 애플리케이션 프레임워크의 네이밍 시스템이 특정한 이니셜 팩토리, 프로바이더 URL을 필요로 한다면 지정할 수 있다.

접속 정보 지정이 완료되면 2.5.9 업무 데이터베이스 목록으로 돌아간다.

2.5.13 원격 룰 서비스 유형 설정

원격 룰 서비스를 선택하면 룰 애플리케이션이 사용할 TCP 커넥터 서버 그룹을 선택하게 된다.

```
*******************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형 -> TCP 커넥터 서버 그룹 선택

*******************************************************************************

원격 룰 서비스 호출에 사용할 TCP 커넥터 서버 그룹을 선택하십시오.
(*) 1. online
```

```
선택하십시오. >>
```

Figure 2 - 5 - 14

룰 애플리케이션은 TCP 커넥터 서버 그룹에 지정된 정책에 따라 그 그룹에 속한 TCP 커넥터 서버들에 룰 서비스를 요청하게 된다.

2.5.14 룰 서비스 동시 사용자 수 설정

Figure 2-5-15는 애플리케이션이 동시에 호출할 룰 서비스의 수를 지정하는 단계이다.

```
*******************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 서비스의 유형 -> 동시 사용자 수 설정

*******************************************************************************

룰 서비스의 동시 사용자 수를 설정하십시오.

[필수] 최대 동시 사용자 수 >>
[필수] 평균 동시 사용자 수 >>
```

Figure 2 - 5 - 15

이 사용자 수에 기반하여 룰 서비스를 호출하기 위한 자원이 할당한다. 이러한 자원에는 데이터베이스 커넥션(로컬 룰 서비스), TCP/IP 세션(원격 룰 서비스) 등이 있다.

2.5.15 룰 애플리케이션 설정 요약 및 구성

룰 애플리케이션 정보를 모두 입력하게 되면 아래와 같은 Figure 2-5-16과 같이 요약 정보를 확인한 후 설정 구성을 하게 된다. 설정을 구성하지 않으면 작업했던 내용은 모두 사라지게 된다.

```
[룰 애플리케이션 설정 요약]
* 설정의 이름: DEVPART
* 설명: dev
* 로그
    레벨: TRACE
    태그: test/
* 동시 사용자
```

```
    최대: 10
    평균: 10
* 룰 서비스의 유형: 로컬 룰 서비스
    룰 DB 접속 방식: 전용 커넥션 풀 방식(RULESDB_DEV)
    업무 데이터베이스 목록
        메타DB: 전용 커넥션 풀(기본접속정보)
        업무DB: 전용 커넥션 풀(기본접속정보)
* 모니터 서버 접속 정보: Use default information
* 룰 호출 로그 정보: Disabled
********************************************************************************

이 설정으로 룰 애플리케이션을 구성합니다. 괜찮겠습니까?
    1. Yes
(*) 2. No

선택하십시오. >>
```

Figure 2 - 5 - 16

2.6 애드온 마법사

InnoRules는 고유의 기능 이외에 사용자가 기능을 추가할 수 있는 몇 가지 인터페이스를 제공한다. 이를 애드온이라고 하며, 애드온에는 다음과 같은 것들이 있다.

- ❒ 사용자 정의 함수
- ❒ 콜백 서비스

사용자 정의 함수와 콜백 서비스에 대한 자세한 사항은 Rule Application Programming Guide와 InnoRules User's Guide of Rule Builder를 참조하라.

애드온 마법사는 위의 애드온들을 룰 시스템에 편리하게 등록할 수 있도록 도와준다. 애드온 마법사는 Figure 2-6-1과 같이 다음의 명령으로 실행한다.

```
UNIX 계열에서:
./addon-wiz.sh
Windows에서:
addon-wiz.bat
```

Figure 2 - 6 - 1

애드온을 설정하기 위해서는 애드온 마법사를 실행하기 전에 사용자가 작성한 애드온 라이브러리들을 jar 파일의 형태로 설치 디렉터리의 lib/addons에 위치시켜야 한다. 애드온 마법사는 이 디렉터리의 라이브러리들을 읽어 들여 가용한 애드온의 목록을 추출하게 된다.

2.6.1 소개

Figure 2-6-2는 애드온 마법사의 소개 단계를 보여준다.

```
*******************************************************************************

소개

*******************************************************************************

애드온 마법사에 오신 것을 환영합니다. 이 마법사를 이용하여 룰 시스템에 다양한 추가 기능을 설치할 수 있습니다.
[엔터를 누르십시오]
```

Figure 2 - 6 - 2

2.6.2 애드온 유형 선택

Figure 2-6-3은 설정하고자 하는 애드온의 유형을 선택하는 단계이다.

```
*******************************************************************************

애드온 유형 선택

*******************************************************************************

구성하려는 애드온의 유형을 선택하십시오. 종료하려면 엔터를 누르십시오.
    1. 사용자 정의 함수
        사용자 정의 함수를 구성합니다.
    2. 콜백 서비스
        콜백 서비스를 구성합니다



선택하십시오. >>
```

Figure 2 - 6 - 3

사용자 정의 함수를 선택하면 2.6.3 사용자 정의 함수 목록으로 이동하게 된다. 콜백 서비스를 선택하면 2.6.4 콜백 서비스 목록으로 이동하게 된다.

2.6.3 사용자 정의 함수 목록

Figure 2-6-4는 등록되었거나 등록 가능한 사용자 정의 함수의 목록을 보여준다.

```
*******************************************************************************

애드온 유형 선택 -> 사용자 정의 함수 목록

*******************************************************************************

다음은 가용한 사용자 정의 함수 목록입니다.

[항목들]
-------------------------------------------------------------------------------
    | 함수 이름        | 클래스                                  | 등록 여부 | 문제점 |
-------------------------------------------------------------------------------
```

```
  1 | CONVERTCOLTOROW | com.innorules.udf.ConvertColToRow      | V            |          |
-------------------------------------------------------------------------------------
  2 | EXECRULEPARAMS   | com.innorules.udf.FuncEXECRULEPARAMS | V            |          |
-------------------------------------------------------------------------------------
  3 | RID              | com.innorules.udf.FuncRID            |              |          |
-------------------------------------------------------------------------------------


[가능한 작업들]
a. 등록      b. 등록 해제      c. 모두 등록      d. 새로 고침

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 6 - 4

마법사는 기존에 등록된 사용자 정의 함수들을 목록의 위쪽에 보여준다. 등록 여부 필드의 "V"는 함수가 기존에 등록되어 있는 사용자 정의 함수임을 나타낸다. 등록된 함수들에 추가하여 마법사는 lib/addons 디렉터리의 jar 파일들 중 사용자 정의 함수 인터페이스 (com.innoexpert.innorulesj.suite.udf.UserDefinedFunction)를 구현한 클래스를 검색하여 리스트에 보여준다. 기존에 등록된 함수의 클래스는 이 jar 파일들에 존재해야 한다. 그렇지 않으면 문제점 필드에 클래스를 찾을 수 없다는 경고 메시지가 나타난다. 또한, 기존에 등록된 함수들은 함수의 이름이 중복되지 않아야 한다. 그럴 경우 문제점 필드에 같은 이름의 사용자 정의 함수가 이미 등록되었다는 경고 메시지가 나타난다.

각 필드의 의미는 다음과 같다.

- ❒ 함수 이름
  사용자 정의 함수의 이름이다. 사용자 정의 함수 이름은 UserDefinedFunction.getName 메소드를 호출하여 가져온다. 함수 이름은 등록 후에도 변경이 가능하나(사용자 정의 함수 코드를 수정하여 getName에서 다른 이름이 반환되도록 하여), 그럴 경우 기존에 등록된 다른 함수와 이름이 겹칠 수도 있다. 이 경우 위에서 서술한 바와 같이 경고 메시지가 출력된다.
- ❒ 클래스
  사용자 정의 함수 클래스 이름이다.
- ❒ 등록 여부
  이 사용자 정의 함수를 룰 시스템에서 사용할 수 있도록 등록한 경우 "V"가 표시된다.
- ❒ 문제점
  등록된 사용자 정의 함수에 문제가 있는 경우 경고 메시지가 출력된다. 이 경고를 적절히 해소하지 않을 경우 런타임에 에러가 발생할 수도 있다.

작업 메뉴는 다음과 같다.

- ❒ 등록
  아직 등록되지 않은 사용자 정의 함수를 등록한다. 기존에 등록된 사용자 정의 함수 중에 등록하려는 함수의 이름과 같은 이름의 함수가 있는 경우 등록할 수 없다.
- ❒ 등록 해제

이미 등록된 사용자 정의 함수를 등록 해제한다.

- ❒ 모두 등록
  등록 가능한 사용자 정의 함수를 모두 등록한다. 기존에 등록된 사용자 정의 함수 중에 등록하려는 함수의 이름과 같은 이름의 함수가 있는 경우 해당 함수를 제외 하고 등록한다.
- ❒ 새로 고침
  사용자 정의 함수의 가용 상태를 새로 고친다. 마법사는 lib/addons 디렉터리의 라이브러리들을 다시 분석하여 리스트를 갱신하고 기존에 등록된 사용자 정의 함수들에 문제점이 있는가 검사한다.

2.6.4 콜백 서비스 목록

Figure 2-6-5는 등록되었거나 등록이 가능한 콜백 서비스 팩토리들의 목록을 보여준다.

```
*******************************************************************************

애드온 유형 선택 -> 콜백 서비스 목록

*******************************************************************************

다음은 가용한 콜백 서비스 팩토리 목록입니다.

[항목들]
-------------------------------------------------------------------------------
    | 클래스                                                | 서비스 이름 | 등록 여부 | 문제점 |
-------------------------------------------------------------------------------
  1 | com.innorules.callbacks.EAIDispatcherFactory    | EAI         | V         |        |
-------------------------------------------------------------------------------
  2 | com.innorules.callbacks.WSInvokerFactory        |             |           |        |
-------------------------------------------------------------------------------

[가능한 작업들]
a. 등록     b. 등록 해제     c. 새로 고침

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 6 - 5

마법사는 기존에 등록된 콜백 핸들러 팩토리들을 목록의 위쪽에 보여준다. 등록 여부 필드의 "V"는 팩토리가 콜백 서비스로 등록되어 있다는 것을 나타낸다. 이미 등록된 콜백 핸들러 팩토리들에 추가하여 마법사는 lib/addons 디렉터리의 jar 파일들 중 콜백 핸들러 팩토리 인터페이스(com.innorules.rulesclient.CallbackHandlerFactory)를 구현한 클래스를 검색하여 리스트에 보여준다. 기

존에 등록된 팩토리의 클래스는 이 jar 파일들에 존재해야 한다. 그렇지 않으면 문제점 필드에 클래스를 찾을 수 없다는 경고 메시지가 나타난다.

각 필드의 의미는 다음과 같다.

- ❒ 클래스
  콜백 핸들러 팩토리 클래스의 이름이다.
- ❒ 서비스 이름
  이 콜백 핸들러 팩토리가 어떤 콜백 서비스에 등록되어 있는가 보여준다. 콜백 서비스로 등록되지 않은 팩토리에는 공란으로 표시된다. 두 개 이상의 팩토리가 동일한 콜백 서비스로 등록될 수는 없다.
- ❒ 등록 여부
  콜백 핸들러 팩토리가 콜백 서비스로 등록되어 있는 경우 "V"가 표시된다.
- ❒ 문제점
  이미 등록되어 있는 팩토리 클래스를 lib/addons 경로에서 찾을 수 없는 경우 경고 메시지가 보인다. 이 경고를 적절히 해소하지 않을 경우 런타임에 에러가 발생할 수 있다.

작업 메뉴는 다음과 같다.

- ❒ 등록
  아직 등록되지 않은 콜백 핸들러 팩토리를 콜백 서비스로 등록한다. 이 메뉴를 선택하면 2.6.5 콜백 서비스 등록으로 이동하게 된다.
- ❒ 등록 해제
  콜백 서비스로 등록된 콜백 핸들러 팩토리를 등록 해제한다.
- ❒ 새로 고침
  콜백 핸들러 팩토리의 사용 상태를 새로 고친다. 마법사는 lib/addons 디렉터리의 라이브러리들을 다시 분석하여 리스트를 갱신하고 기존에 등록된 콜백 서비스 팩토리들에 문제점이 있는가 검사한다.

2.6.5 콜백 서비스 등록

Figure 2-6-6은 콜백 핸들러 팩토리를 콜백 서비스로 등록하는 단계를 보여준다.

```
********************************************************************************

애드온 유형 선택 -> 콜백 서비스 목록 -> 콜백 서비스 등록

********************************************************************************

콜백 핸들러 팩토리 com.innorules.callbacks.WSInvokerFactory에 대한 서비스 이름을 지정하십시오.

 >>
```

Figure 2 - 6 - 6

콜백 서비스 이름은 64자 이내의 영문자와 숫자 또는 언더스코어로 이루어져야 한다. 또한, 이미 등록되어 있는 서비스의 이름을 사용할 수는 없다.

2.7 마법사 사용 팁

여기서는 마법사를 사용하는 데에 필요한 간단한 팁들을 설명한다.

2.7.1 입력

2.7.1.1 엔터의 용법

입력을 하지 않아도 되는 필드가 있는 경우 레이블 앞에 [선택]이라는 태그가 붙는다. [선택] 필드에서 값을 입력하지 않고 엔터를 누르면 그 값을 설정하지 않은 것으로 간주된다.

```
********************************************************************************

룰 시스템 선택 -> 룰 애플리케이션 설정 -> 룰 애플리케이션 설정의 기본 정보

********************************************************************************

룰 애플리케이션 설정의 기본 정보를 입력하십시오.

[필수] 룰 애플리케이션 설정 이름 >> batch
[선택] 룰 애플리케이션 설정에 대한 설명 >>
```

Figure 2 - 7 - 1

Figure 2-7-1의 "룰 애플리케이션 설정에 대한 설명"은 선택 필드이다. 여기서 값을 입력하지 않고 엔터를 치면 이 필드의 값을 지정하지 않은 것으로 간주된다.

이미 설정이 된 내용을 수정하거나 룰 시스템이 추천하는 값이 있을 때는 기본값이 출력된다.

```
********************************************************************************

설정 대상 선택 -> 단위 룰 시스템 관리 -> 단위 룰 시스템 기본 정보

********************************************************************************

단위 룰 시스템의 기본 정보를 입력하십시오.

[필수] 이름(기본값: Development) >>
```

Figure 2 - 7 - 2

질문 바로 뒤의 괄호 안에 "기본값: "이라는 레이블로 기본값이 있음을 알 수 있다. 기본값이 있는 경우 엔터만 입력하면 별도의 입력 없이 기본값으로 주어진 값을 설정할 수 있다. Figure 2-7-2의 예에서 이름 란에서 엔터를 치면 이름으로 "Development"가 지정된다.

목록을 조회하는 화면에서 작업을 선택하면 그 화면의 부 메뉴로 이동한다. 그 화면에서 작업을 선택하지 않고 엔터를 입력하면 다음 단계로 진행한다.

```
*******************************************************************************

설정 대상 선택 -> 단위 룰 시스템 관리

*******************************************************************************

등록된 단위 룰 시스템의 목록입니다. 새로운 단위 룰 시스템을 추가하거나 기존의 것을 수정할 수 있습니다.

[항목들]
--------------------------
    | ID   | 이름          |
--------------------------
  1 | DEV  | Development |
--------------------------
  2 | PROD | Production  |
--------------------------

[가능한 작업들]
a. 추가     b. 수정 및 룰 DB 관리     c. 마지막 단위 룰 시스템 삭제

작업을 선택하십시오. 건너 뛰려면 엔터를 누르십시오. >>
```

Figure 2 - 7 - 3

Figure 2-7-3의 단위 룰 시스템 관리에는 "추가", "수정 및 룰 DB 관리" 등의 부 메뉴가 있다. 그러나, 이를 선택하고 않고 엔터를 입력하면 다음 단계로 진행하게 된다.

2.7.1.2 특수 문자

값을 입력하는 란에서 다음의 특수 문자를 이용할 수 있다. 특수 문자를 이용하려는 경우 다른 문자들과 같이 사용할 수 없고 앞뒤의 공백도 있어서는 아니 된다.

- ^b

  "뒤로 가기" 작업을 수행한다. 이전에 수행한 단계로 이동한다.

- ^e
  공백을 입력한 것으로 간주한다.
- ^q
  마법사를 종료한다. Ctrl-C를 이용하여 종료하는 경우 마법사가 정상적으로 종료되지 않아 락 파일이나 UUID 파일이 남겨질 수 있다.

2.7.1.3 확인 메시지에 대한 응답

```
설정을 그만 마치시겠습니까? (Yes/No) >>
```

Figure 2 - 7 - 4

Figure 2-7-4과 같이 "Yes" 또는 "No"를 묻는 확인 메시지에서 "Yes"나 "No" 전체를 입력하지 않고 대소문자 구분 없이 답변의 앞 일부분만 입력해도 된다. 예를 들어, 다음은 모두 "Yes"로 간주된다.

- Yes
- yes
- y

2.7.2 마법사 사용 시나리오

설치 마법사를 이용하여 일단 설치가 완료되면 필요에 따라 순서에 관계없이 설정 마법사를 사용할 수 있다. 다음은 흔히 발생하는 시나리오와 사용되는 마법사들을 순서대로 기술한 예이다.

- 개발 시스템 구성: 룰 개발자가 룰 빌더를 이용하여 룰을 개발할 수 있도록 요청받음
  개발 시스템에 룰 시스템 설치를 요청받았다.
  - 설치 마법사를 이용하여 개발 장비에 InnoRules를 설치한다.
  - 룰 시스템 구성 마법사를 이용하여 개발 단위 시스템을 설정한다.
  - 서비스 구성 마법사를 이용하여 룰 빌더 서비스를 활성화하고 InnoRules 서버를 실행한다.
- 애플리케이션 구성: 개발 장비에 업무 애플리케이션이 설치되고 이들이 룰 서비스를 호출할 수 있도록 요청받음
  - 룰 애플리케이션 구성 마법사를 이용하여 룰 애플리케이션을 설정하고, 애플리케이션에 룰 초기화를 할 수 있도록 구성
- 업무 데이터베이스 추가: 룰 개발자가 업무 데이터베이스를 접근하는 룰 서비스를 개발할 수 있도록 요청함
  - 룰 시스템 구성 마법사를 이용하여 업무 데이터베이스를 등록
  - 룰 애플리케이션 구성 마법사를 이용하여 애플리케이션의 업무 데이터베이스 접속 방법 등록
- 운영 시스템 구성: 운영 장비들이 설치되고 여기에 룰 시스템 설치를 요청받음.
  - 설치 마법사를 이용하여 운영 장비들에 InnoRules를 설치한다.
  - 설정 클러스터 마법사를 이용하여 운영 장비에 설치된 InnoRules를 클러스터로 구성한다.
  - 룰 시스템 구성 마법사를 이용하여 운영계 클러스터에 개발 단위 시스템, 운영 단위 시스템을 등록한다. 또한 업무 데이터베이스를 등록한다.
  - 서비스 구성 마법사를 이용하여 룰 빌더 서버를 이관 전용으로 활성화한다.

- 룰 애플리케이션 구성 마법사를 이용하여 운영 애플리케이션이 룰 서비스를 호출할 수 있도록 한다.

2.7.3 설정 적용 시점

마법사를 이용하여 설정을 변경하면 그 내용은 설정 클러스터 내 모든 설치 노드의 config/rule-systems.xml에 저장된다. 그러나, 이것이 클러스터 내에서 실행되는 모든 InnoRules 서비스, 룰 애플리케이션에 반영되는 것을 의미하지는 않는다. InnoRules 서비스는 InnoRules 서버를 재시작할 때 반영이 되며 룰 애플리케이션 설정 변경도 룰 애플리케이션이 재시작 되어야 반영이 된다.

3. Rule System Concepts

이 장에서는 InnoRules 설치와 운영을 하는 데에 필요한 여러 개념에 대해 설명한다.

3.1 InnoRules를 설치하는 것이란?

InnoRules를 **설치(installation)**한다는 것은 InnoRules와 관련된 실행 스크립트, 프로그램 라이브러리, 설정 파일 등을 미리 정의된 디렉터리 구조로 파일 시스템에 위치시키는 것을 말한다. 설치에 의해 생성된 일련의 디렉터리와 파일들을 **설치 노드(Installation Node)**라고 한다. 하나의 호스트에 디렉터리를 달리 설치하여 여러 개의 설치 노드를 만들 수 있다. 단, 각각의 설치 노드는 각각 라이선스를 필요로 한다. 각각의 설치 노드의 설정을 변경하기 위해서는 그 노드에 포함되어 있는 마법사 스크립트를 실행하면 된다.

3.2 설정 클러스터

두 개 이상의 설치 노드의 설정을 동기화하여 관리할 수 있다. 설정이 동기화 된 설치 노드의 묶음을 **설정 클러스터(Configuration Cluster)**라고 한다. 설정 클러스터의 노드 중 한 곳에서 설정을 변경하면 그 변경 사항은 나머지 모든 노드에도 동일하게 변경되어 동기화된다.
설정 클러스터에 설치 노드를 추가하기 위해서는 그 노드의 설정 파일에 원격으로 접근하고 수정할 수 있는 방법이 지정되어야 한다. 원격 파일 접근 프로토콜로 SFTP와 FTP가 지원된다. 임의의 설치 노드에서 나머지 설치 노드들에 접근할 수 있어야 하므로, localhost, 127.0.0.1과 같은 루프백(loopback) 주소를 설치 노드의 주소로 지정할 수 없다.
설정 클러스터에 설치 노드를 추가하기 위해서는 **클러스터 구성 마법사(Cluster Configuration Wizard)**를 이용한다. 어떤 노드에서 마법사를 이용하여 설정 클러스터의 구성을 변경하려면 그 노드가 먼저 설정 클러스터에 추가되어 있어야 한다.

설치 후에 설정 클러스터를 필히 구성해야 하는 것은 아니다. 설치 노드가 하나만 있는 경우 굳이 설정 클러스터를 구성할 필요가 없다.

3.3 룰 시스템

룰 시스템(Rule System)이란 룰을 작성, 조회, 저장하고 실행하며 룰의 라이프 사이클을 관리하기 위해 유기적으로 동작하는 소프트웨어 컴포넌트와 자원을 통칭한다.

3.3.1 단위 룰 시스템

애플리케이션은 운영 환경에 배포되기까지 개발, 테스트 등의 단계를 거친다. 룰들도 개발부터 운영에 이르기까지 여러 단계를 거칠 수 있다. 룰은 애플리케이션과 밀접한 관계를 가지므로 일반적으로 룰의 개발 단계들과 애플리케이션의 개발 단계들은 1:1로 대응된다. 룰 시스템에서 애플리케이션 개발 단계

에 대응하는 개념이 단위 룰 시스템이다.

단위 룰 시스템(Unit Rule System)이란 동기화된 일련의 룰 저장소들과 이들을 공유하는 룰 서버들 및 룰 빌더 서버들의 집합을 말한다. 여기서 동기화(synchronization)된다는 것은 룰 저장소들의 내용이 모두 동일하도록 관리된다는 것을 의미한다.

일반적으로 단위 룰 시스템의 이름은 대응하는 애플리케이션 개발 단계의 이름을 따른다. 예를 들어, 애플리케이션의 개발 단계가 개발, 테스트, 운영으로 구성된 시스템의 룰 시스템은 이에 대응하는 세 개의 단위 룰 시스템으로 이루어질 수 있다. 이 단위 룰 시스템들은 대응하는 애플리케이션 개발 단계의 이름을 따서 개발 룰 시스템, 테스트 룰 시스템, 운영 룰 시스템으로 부여하는 것이 좋다.

3.3.1.1 룰 저장소

룰, 항목 등 룰의 내용을 구성하는 요소들을 **룰 요소(Rule Element)**라고 한다. 그리고, 룰 요소들이 영구적으로 저장되는 곳을 **룰 저장소(Rule Repository)**라고 한다. 룰 요소들은 룰 빌더 서버에 의해 조회 및 수정이 된다.

일반적으로 데이터베이스가 룰 저장소로 사용된다. **룰 DB(Rule DB)**는 룰 저장소의 용도로 작성된 독립적인 스키마가 부여된 일련의 테이블들을 가리킨다. 데이터베이스 서버로는 SQL-99와 JDBC3.0을 준수하는 모든 데이터베이스 서버가 사용될 수 있다.

하나의 단위 룰 시스템에 하나 이상의 룰 DB를 설정할 수 있다. 룰을 편집하는 등 룰 요소를 변경하게 되면 룰 빌더 서버는 모든 룰 DB를 동일하게 변경하여 실시간으로 동기화한다.

3.3.2 이관

애플리케이션 시스템과 마찬가지로 룰 시스템에서도 개발 룰 시스템에서 개발이 완료된 룰 요소들을 테스트 룰 시스템으로 복사하거나 테스트 룰 시스템에서 테스트가 완료된 룰 요소들을 운영 룰 시스템으로 복사를 할 수 있다. 이와 같이 단위 룰 시스템 저장소의 룰 요소들 중 일부를 다른 단위 룰 시스템의 저장소로 복사를 하는 것을 **이관(Transition)**이라고 한다.

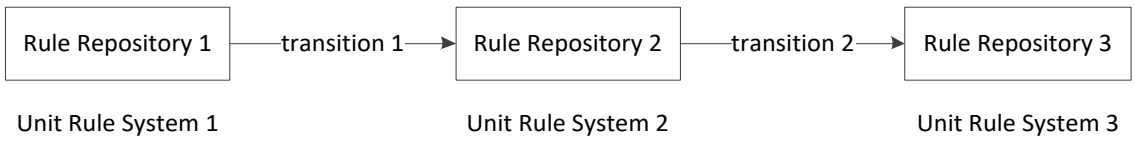

Figure 3 - 3 - 1 이관의 예

Figure 3-3-1에서 Unit Rule System 1의 룰 요소들은 Unit Rule System 2로 이관이 되며(transition 1), Unit Rule System 2의 룰 요소들은 Unit Rule System 3으로 이관이 된다(transition 2). 이관에서 복사할 룰 요소가 위치한 단위 룰 시스템을 **원본 룰 시스템(Source Rule System)**이라고 하며, 룰 요소가 복사되는 단위 룰 시스템을 **대상 룰 시스템(Target Rule System)**이라고 한다. 원본 룰 시스템과 대상 룰 시스템을 각각 줄여서 원본 시스템과 대상 시스템이라고 부르기도 한다.

이관은 다음의 제약을 갖는다.

- ❒ 하나의 단위 룰 시스템이 두 개 이상의 단위 룰 시스템의 원본 시스템이 될 수 없다. 또한, 하

나의 단위 룰 시스템이 두 개 이상의 단위 룰 시스템의 대상 시스템이 될 수도 없다. (Figure 3-3-2의 (a))

- ❒ 하나의 단위 룰 시스템으로부터 이관된 룰 요소가 결국 자기 자신으로 이관될 수 없다. 바꿔 말해서, 순환적인 형태의 이관은 불가능하다. (Figure 3-3-2의 (b))

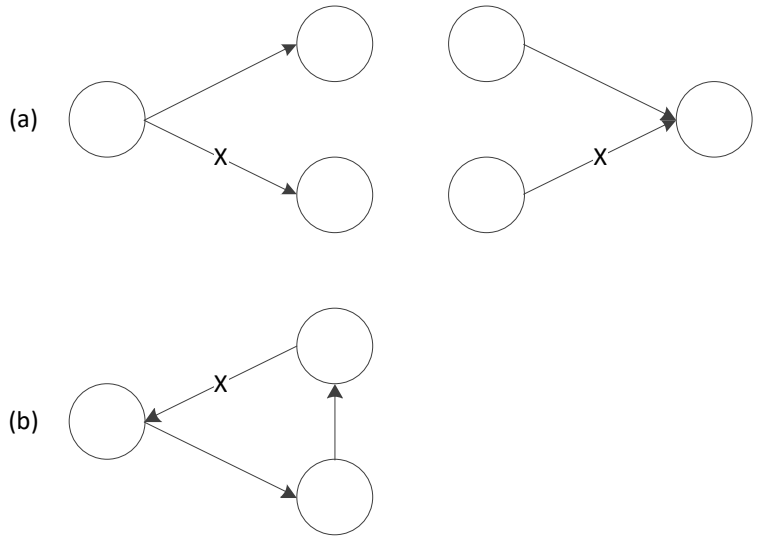

Figure 3 - 3 - 2 이관의 제약

이관된 룰 요소들의 변경 이력은 대상 시스템 상에 기록되며, 룰 요소의 변경 히스토리 및 그 내용을 대상 시스템에서 확인할 수 있다.

3.4 업무 데이터베이스

비즈니스 룰을 처리하는 과정 중에 업무 데이터를 조회해야 할 경우가 있다. 룰 DB와 대조적인 개념으로 업무 데이터가 저장되어 있는 데이터베이스를 **업무 데이터베이스(Business Database)** 또는 **업무 DB**라고 한다. 룰 시스템은 업무 데이터베이스를 접속하고 쿼리할 수 있는데, 주로 룰 엔진이 DB 룰 또는 데이터 룰을 실행할 때 조회하게 된다.

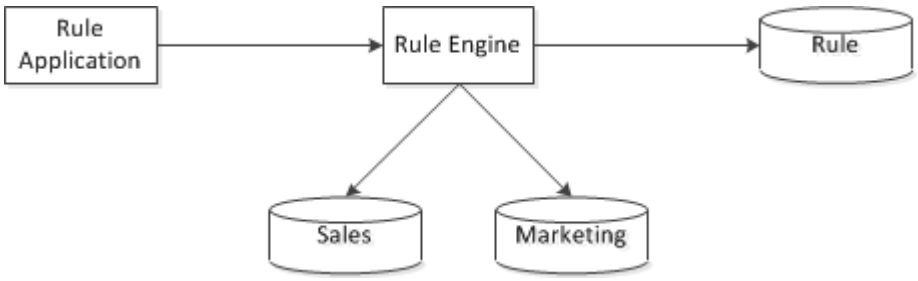

Figure 3 - 4 - 1 Business Database

Figure 3-4-1은 업무 데이터베이스를 접근하는 단위 룰 시스템의 예를 보여준다. 이 그림의 룰 엔진은 서로 다른 목적으로 세 개의 데이터베이스를 접근한다.[*1] *Rule*로 표기된 데이터베이스는 룰 DB로서 이 단위 시스템의 룰들이 저장되어 있으며, 룰 엔진이 룰 데이터를 읽어 들이기 위해 조회한다. *Sales*와 *Marketing*으로 표기된 데이터베이스에는 각각 영업 데이터와 마케팅 데이터가 저장되어 있으며, 각각

에 대응하는 업무 애플리케이션이 이들의 데이터를 관리할 것이다. 룰 애플리케이션이 룰 서비스를 호출하면 룰 엔진은 (필요하다면) 룰 DB로부터 룰 데이터를 읽어 들여 룰들을 실행하고, 이 과정 중에 DB 룰 또는 데이터 룰이 실행되면 이들 업무 데이터베이스를 접근하게 된다.

룰 시스템에서 접근할 수 있는 업무 데이터베이스들은 시스템 마법사(System Wizard)를 이용하여 등록할 수 있다. 업무 데이터베이스를 등록할 때 Sales, Marketing과 같이 업무 데이터베이스를 식별할 수 있는 이름과 물리적인 접속 정보를 등록하게 된다. DB 룰 또는 데이터 모델이 어떤 업무 데이터베이스를 접근할지는 룰 빌더를 이용하여 각각의 룰 또는 데이터 모델에 지정한다. 단, 룰 빌더에는 물리적인 접속 정보는 보이지 않으며 업무 데이터베이스의 이름만 보인다.
같은 업무 데이터베이스라도 룰 애플리케이션마다 다른 데이터베이스 서버를 접속할 필요가 있다. 예를 들어, Sales 데이터 조회를 위해 온라인 애플리케이션과 배치 애플리케이션이 서로 다른 서버에 접속할 필요가 있다. 업무 데이터베이스에 대한 정보는 룰 애플리케이션 설정에서 오버라이드(override)할 수 있다(3.6.8 참조). 그렇지 않은 경우 시스템 마법사를 이용하여 등록한 접속 정보를 기본으로 사용하게 된다.

*1. 일반적으로 각 데이터베이스의 접속 정보는 상이할 것이다.

3.5 룰 서비스

룰 서비스(Rule Service)는 애플리케이션으로부터 주어진 입력들을 이용하여 지정된 룰을 판정하고 그 결과를 애플리케이션으로 돌려주는 서비스를 말한다. 룰 서비스는 룰 서버가 제공하며 룰 서비스를 호출하는 애플리케이션을 **룰 애플리케이션(Rule Application)**이라고 한다.

룰 서비스는 룰 애플리케이션과 룰 서버와의 통신 방법에 따라 로컬 룰 서비스와 원격 룰 서비스로 구분된다. **로컬 룰 서비스(Local Rule Service)**는 룰 서버가 애플리케이션과 동일한 JVM에서 실행되어 통신 계층을 거치지 않고 자바 메소드만을 이용하여 호출되는 룰 서비스이다. 룰 서버가 자바로 작성되어 있으므로 자바 애플리케이션만이 로컬 룰 서비스를 사용할 수 있다. 반면, **원격 룰 서비스(Remote Rule Service)**는 서비스 호출을 위해 네트워크 계층을 이용하는 룰 서비스를 말한다. 네트워크 계층을 이용하므로 룰 애플리케이션은 자바로 작성된 애플리케이션에 국한되지 않는다.

원격으로 룰을 서비스하기 위해서는 네트워크를 통해 룰 서비스 요청을 받아들여 룰 엔진으로 전달하고, 반대로 룰 실행 결과를 네트워크를 통해 애플리케이션으로 전달하는 서버가 필요하다. 이러한 서버에는 다음이 있다.

❒ TCP 커넥터 서버

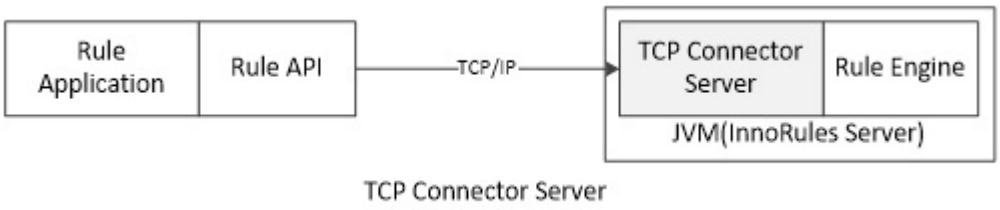

Figure 3 - 5 - 1 Remote Rule Server

TCP 커넥터 서버(TCP Connector Server)는 TCP/IP와 InnoRules 고유의 메시지 프로토콜을 이용하여 원격 룰 서비스를 제공한다. TCP 커넥터 서버를 이용하여 룰 서비스를 호출하기 위해서는 InnoRules Rule API를 사용하여야 한다. API만 사용할 수 있다면 어떤 프로그래밍 언어로 작성된 룰 애플리케이션도 TCP 커넥터 서버를 이용할 수 있다. 룰 API가 지원하는 프로그래밍 언어들에 대해서는 Rule Application Programming Interface Guide를 참조하라.

TCP 커넥터 서버는 InnoRules 서버에서 활성화시킬 수 있다(3.8.4.1 참조).

3.6 룰 애플리케이션 설정

룰 시스템은 다양한 서비스들을 제공한다. 룰 서버는 로컬 룰 서비스를 제공하며, 룰 빌더 서버는 룰 빌더 서비스, TCP 커넥터 서버는 원격 룰 서비스를 제공한다. 이들 서버들은 자바로 작성되었으며 JVM 위에서 동작한다. 이 서버들이 정상적으로 동작하기 위해서 또는 서버가 제공하는 서비스들을 호출하기 위해서는 다양한 구성 요소(component)들이 JVM 위에 구성이 되어야 한다. 다음은 대표적인 구성 요소들이다.*2

- ❒ 로깅
- ❒ 데이터 소스: 룰 DB 및 업무 데이터베이스에 대해
- ❒ 룰 저장소 캐시
- ❒ 룰 인터페이스 클러스터: 로컬 또는 원격 룰 서버로의 인터페이스 관리

이러한 구성 요소들은 다른 구성 요소들 또는 서버들과 의존성을 가진다. 따라서, 어떤 서버를 실행하기 위해서는 서버가 필요로 하는 구성 요소들, 그리고 이 구성 요소들이 필요로 하는 또 다른 구성 요소들이 JVM에 구성이 되어야 한다.

룰 시스템 서비스를 제공하기 위해서 또는 서비스를 호출하기 위해서 JVM에 이 구성 요소들과 서버들을 구성해 놓은 것은 **룰 런타임(Rule Runtime)**이라고 하며, 룰 런타임을 구성하는 과정을 **룰 런타임 초기화(Rule Runtime Initialization)**이라고 한다. 룰 런타임에 어떤 구성 요소와 서버들을 어떻게 구성할지를 지정한 것을 **룰 런타임 설정(Rule Runtime Configuration)**이라고 한다.

룰 애플리케이션은 룰 서비스를 호출해야 하므로 룰 서비스 호출을 위한 룰 런타임이 룰 애플리케이션의 JVM에 구성이 되어야 한다. 룰 서비스를 호출할 수 있도록 룰 런타임을 초기화 하는 것을 특별히 **룰 애플리케이션 초기화(Rule Application Initialization)**라고 한다. 룰 애플리케이션은 룰 서비스를 호출하기 전에(일반적으로 애플리케이션이 시작될 때) 먼저 룰 애플리케이션 초기화를 해야 하고 룰 서비스 호출이 모두 끝나면(일반적으로 애플리케이션이 종료되기 직전) 룰 런타임을 종료해야 한다. 룰 애플리케이션을 어떤 식으로 초기화할 것인가에 대한 설정을 **룰 애플리케이션 설정(Rule Application Configuration)**이라고 한다.

룰 애플리케이션 설정은 **룰 애플리케이션 설정 마법사(Rule Application Configuration Wizard)**를 이용한다. 룰 서비스를 호출하는 모든 애플리케이션은 룰 애플리케이션이라고 부를 수 있지만, 마법사는 자바 룰 애플리케이션만 지원하므로 이 문서에서는 룰 애플리케이션을 자바 룰 애플리케이션으로 한정하다. 다른 언어 환경에서의 룰 애플리케이션 설정에 대해서는 Rule Application Programming Guide를

참조하라.

*2. 이 구성 요소들 중 몇몇에 대한 세부적인 사항은 문서의 범위를 벗어나므로 생략한다.

3.6.1 단위 룰 시스템 선택

대개의 룰 애플리케이션은 자신이 속한 단계에 매핑된 단위 시스템의 룰 서비스만을 호출한다. 예를 들면, 개발 단계의 업무 애플리케이션은 개발 룰 시스템의 룰들만, 운영 단계의 업무 애플리케이션은 운영 룰 시스템의 룰들만을 호출한다.

룰 애플리케이션 설정은 사용하고자 하는 단위 룰 시스템의 ID를 선택하는 것으로부터 시작한다. 마법사의 이후 단계들에서는 선택된 단위 룰 시스템의 자원들만을 지정할 수 있다. 이러한 자원들에는 룰 DB, TCP 커넥터 서버 그룹 등이 있다.

3.6.2 복수의 룰 애플리케이션 설정과 설정 이름 지정

하나의 단위 룰 시스템에 여러 개의 룰 애플리케이션 설정을 만들 수 있다. 각각의 설정들은 이름으로써 식별된다. 이 이름을 **룰 애플리케이션 설정 이름(Rule Application Configuration Name)**이라고 한다. 이 이름은 단위 룰 시스템 안에서 고유하다.

여러 룰 애플리케이션이 하나의 룰 애플리케이션 설정을 이용하여 초기화될 수 있다. 예를 들어, 웹 애플리케이션으로 만들어진 업무 애플리케이션이 있고 이 애플리케이션은 클러스터로 묶인 다수의 WAS 컨테이너에 배치(deploy)된다고 하자. 각 컨테이너에는 배치될 때 동일한 설정이 적용될 것이며 여기에는 룰 애플리케이션 설정 이름도 포함이 될 것이다. 그러면, 각 컨테이너에 배치되는 애플리케이션들은 동일한 룰 애플리케이션 설정으로 초기화 될 것이다.
동종의 애플리케이션이 아니더라도 룰 애플리케이션 설정이 공유될 수 있다. 예를 들어, 배치 애플리케이션과 온라인 애플리케이션이 있고 이들이 동일한 구성의 룰 런타임을 이용하여 룰 서비스를 이용한다면 하나의 룰 애플리케이션 설정을 이용하여 초기화할 수 있다.

3.6.3 로그 레벨 지정

로그에는 룰 애플리케이션이 초기화 되거나 룰 서비스가 호출되는 동안 발생한 이벤트가 기록된다. 이벤트들은 중요도에 따라서 등급이 나누어져 있다. 등급과 그에 해당하는 이벤트들은 다음과 같다.

- ❒ ERROR: 가장 높은 수준
  룰 서비스가 정상적으로 호출되지 않을 수도 있는 수준의 장애
- ❒ WARN
  룰 서비스는 호출되었지만 룰 실행 도중 에러가 발생함. 룰 서비스 호출 가능 여부와 관계없는 업무 수준의 에러
- ❒ INFO
  룰 애플리케이션 초기화 및 종료 등에 관련된 주요 정보
- ❒ DEBUG
  룰 실행 시간, 처리 소요 시간 등 추적에 관련된 이벤트

- TRACE: 가장 낮은 수준
  룰 서비스의 입력 항목 및 결과

룰 애플리케이션 설정에는 로그로 남길 이벤트의 등급인 로그 레벨을 지정할 수 있다. 지정된 로그 레벨보다 같거나 높은 수준의 이벤트만이 로그에 남게 된다.

로그는 파일에 남는다. 로그 파일은 제품을 설치할 때 지정한 디렉터리의 하위에 다음의 경로에 생성이 된다.

[단위 룰 시스템 ID]/[룰 애플리케이션 설정 이름]/innorules.log

로그 파일의 기본 이름은 innorules.log이다. 하나의 룰 애플리케이션 설정을 이용하여 여러 룰 애플리케이션이 실행될 수 있기 때문에 룰 애플리케이션별로 로그 파일을 구분하기 위해 **로그 태그(Log Tag)**를 지정할 수 있다. 로그 태그를 지정하면 다음의 경로에 로그 파일이 생성된다.

[단위 룰 시스템 ID]/[룰 애플리케이션 설정 이름]/[로그 태그]innorules.log

로그 태그에는 시스템 프로퍼티 및 경로 구분자를 사용할 수 있다. 다음은 그 사용 예들이다.

- ${container.name}-
  동일한 룰 애플리케이션 설정을 이용하는 웹 애플리케이션을 여러 컨테이너에 배치할 때 컨테이너 이름을 이용하여 파일을 분리할 수 있다. 각 컨테이너의 container.name 시스템 프로퍼티에 컨테이너의 이름이 저장된 경우 로그 태그를 위와 같이 설정하면 각 룰 애플리케이션의 로그 파일 이름의 앞에 컨테이너 이름(그리고 하이픈)이 붙는다.
- ${container.name}/ 또는 ${container.name}${file.separator}
  위의 예에서 컨테이너별로 룰 애플리케이션 로그를 다른 서브 디렉터리에 저장하기 위해서 로그 태그에 경로 구분자를 사용할 수 있다. 경로 구분자는 문자 또는 file.separator와 같은 시스템 프로퍼티를 사용할 수 있다.

로그 파일에는 룰 애플리케이션에서 발생한 이벤트만이 기록된다. 룰 애플리케이션이 로컬 룰 서비스를 이용하도록 설정한 경우 룰 서버가 룰 애플리케이션의 JVM 내에서 실행되기 때문에 룰 서버의 로그들이 로그 파일에 기록된다. 그러나, 원격 룰 서비스를 이용하도록 설정한 경우 룰 서버는 원격지에서 실행되므로 룰 서버의 로그는 원격 룰 서버의 로그 파일에 기록된다.

3.6.4 룰 서비스의 형태 지정

룰 애플리케이션이 로컬 룰 서비스를 사용할 것인지 원격 룰 서비스를 사용할 것인지 지정해야 한다. 마법사의 이후에 단계들에서 서비스 형태에 의존적인 추가적인 정보를 입력하게 된다. 로컬 룰 서비스를 선택하는 경우 룰 DB 접속을 위한 설정을 해야 한다. 원격 룰 서비스를 선택하는 경우 호출할 TCP 커넥터 서버들에 대한 정보를 입력해야 한다.

3.6.5 룰 DB 접속 설정(로컬 룰 서비스를 사용하는 경우에만)

룰 애플리케이션이 로컬 룰 서비스를 사용하도록 설정하면 애플리케이션의 JVM 내에 룰 서버가 실행이 되도록 구성이 된다. 룰 서버는 룰 데이터를 조회해야 하므로 룰 DB에 접속할 수 있는 방법에 대한 추가적인 설정이 필요하다.

룰 서버가 사용할 데이터 소스를 지정함으로써 룰 DB 접속 방법을 설정한다. 데이터 소스로는 네이밍 시스템에 미리 등록되어 있는 데이터 소스를 참조하도록 할 수도 있고 룰 애플리케이션 전용의 커넥션 풀도 생성이 가능하다. 전자의 방식을 **데이터 소스 참조 방식(Data Source Referencing Type)**이라고 하고 후자의 방식을 **전용 커넥션 풀 방식(Private Connection Pool Type)**이라고 한다.

데이터 소스 참조 방식을 사용할 경우 데이터 소스의 JNDI 이름을 지정하게 된다. 룰 서버는 데이터 소스를 검색하기 위해 JNDI API를 사용한다. 이 방식을 사용하게 될 대표적인 애플리케이션 형태는 웹 애플리케이션이다. 웹 애플리케이션이 아니더라도 네이밍 시스템을 사용할 수 있다면 이 방식으로 지정할 수 있다. 그러나, 설정 방법에서 웹 애플리케이션과 약간 차이가 있다. 아래에서 웹 애플리케이션 환경에서의 설정의 방법과 의미를 먼저 설명하고 비 웹 애플리케이션의 그것들에 대해서 설명한다.

웹 애플리케이션은 웹 애플리케이션 서버(WAS)에서 동작한다. 일반적으로 데이터 소스는 WAS에 의해 생성 및 관리되고 웹 애플리케이션은 이를 참조하여 사용한다. 웹 애플리케이션이 WAS의 데이터 소스를 참조하기 위해서는 웹 애플리케이션 배치 기술자(Web Application Deployment Descriptor, web.xml)에 참조 이름을 기술해야 한다. [*3] 이 참조 이름을 웹 애플리케이션에서 참조하기 위해서는 그 앞에 접두어(prefix) java:/comp/env/를 붙여야 하므로 룰 애플리케이션 설정에는 참조 이름에 접두어를 붙여 지정해야 한다.
예를 들어, WAS에 룰 DB에 대한 데이터 소스가 등록되어 있고 이 자원의 이름이 RULEDB_POOL이라고 하자. 룰 애플리케이션의 web.xml에는 룰 DB의 자원 참조가 기술되어 있어야 하며, 아래 표와 같이 그 이름이 RULEDB로 지정되어 있다고 하자.

```
<web-app>
...
    <resource-ref>
        ...
        <res-ref-name>RULEDB</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        ...
    </resource-ref>
    ...
</web-app>
```

Table 3 - 6 - 1 Part of Rule Web Application Deployment Descriptor

룰 애플리케이션이 룰 DB에 대한 데이터 소스로서 RULEDB_POOL이라는 데이터 소스를 사용하기 위해서는 다음 두 가지를 만족해야 한다.

- ❒ 룰 애플리케이션의 설정에서 룰 DB 접속 방식을 데이터 소스 참조 방식으로 설정하고 참조 이름을 java:comp/env/RULEDB로 설정한다. 이는 룰 애플리케이션 관리자의 역할이다.
- ❒ 룰 웹 애플리케이션을 배치(deploy)할 때 자원 참조 RULEDB가 WAS의 데이터 소스 RULEDB_POOL을 참조하도록 매핑한다. 웹 애플리케이션 배치는 WAS 관리자의 역할이다.

이 방식으로 룰 웹 애플리케이션이 배치된 것을 그림으로 표현하면 Figure 3-6-1과 같다.

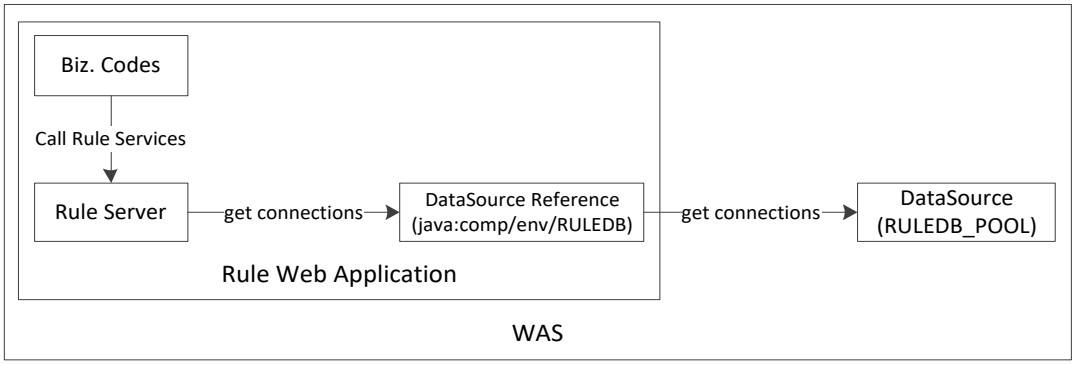

Figure 3 - 6 - 1 Referencing a Data Source from a Rule Web Application

웹 애플리케이션이 아닌 일반 자바 애플리케이션도 기본적으로는 동일한 방법을 사용한다. 룰 애플리케이션 설정에서 데이터 소스 참조 이름에 애플리케이션 실행 환경에 등록된 데이터 소스의 JNDI 이름을 입력한다. 추가적으로 입력 가능한 파라미터로 이니셜 컨텍스트 팩토리(Initial Context Factory)와 프로바이더 URL(Provider URL)이 있다.*4

전용 커넥션 풀 방식은 룰 애플리케이션이 초기화될 때 룰 DB 전용 커넥션 풀을 생성하는 방식이다. 커넥션 풀의 DB 접속 정보로 단위 룰 시스템에 등록된 룰 DB 접속 정보들(3.3.1.1 참조) 중 하나를 선택할 수 있다. 단위 룰 시스템에 룰 DB의 접속 정보를 등록하는 방법은 시스템 마법사를 참조하라. 커넥션 풀에서 관리할 커넥션의 개수는 룰 서비스의 동시 사용자 수에 맞게 적절히 설정된다. 룰 서비스 동시 사용자 수는 3.6.6에서 설명한다. 커넥션 풀은 룰 애플리케이션이 종료될 때 소멸된다.
전용 커넥션 풀 방식은 웹 애플리케이션을 포함한 모든 형태의 룰 애플리케이션에 적용 가능하다. 그러나, 애플리케이션에 룰 DB로의 커넥션들을 관리하는 또 다른 데이터 소스가 있는 경우 자원의 낭비가 될 수도 있으니 전용 커넥션 풀 방식을 선택할 때에는 이 점을 고려해야 한다.

*3. 웹 애플리케이션 배치 기술자 및 자원 참조와 관련하여서는 서블릿 사양(Servlet Specification)을 참조하라. 이 절은 독자가 웹 애플리케이션 설치에 관하여 충분한 지식을 갖고 있다고 가정한다.

*4. 자세한 내용은 JNDI 관련 자료를 참조하라.

3.6.6 동시 사용자 수

룰 애플리케이션은 동시에 여러 건의 룰 서비스들을 독립적으로 호출이 가능하다. 멀티 스레드 애플리케이션들이 이러한 형태로 룰 서비스들을 호출한다. 각각의 서비스 호출은 다른 서비스 호출에 대해 독점적으로 일정량의 자원을 점유하게 된다. 점유하는 자원은 룰 서비스의 형태에 따라 차이가 있다. 로컬 룰 서비스의 경우 룰 서비스가 호출되는 동안 일정량의 자바 힙(heap)을 점유한다. 원격 룰 서비스의 경우에는 원격 룰 서버로의 TCP/IP 세션 하나를 점유한다. 룰 서비스의 형태에 따라 서로 다른 자원들을 공통의 인터페이스로 추상화시킨 것이 **룰 인터페이스(Rule Interface)**이다. 애플리케이션 개발자는 룰 서비스의 형태에 관계없이 룰 인터페이스만을 이용하여 룰 서비스 호출 코드를 작성한다. 이를 그림으로 표현하면 Figure 3-6-2과 같다.

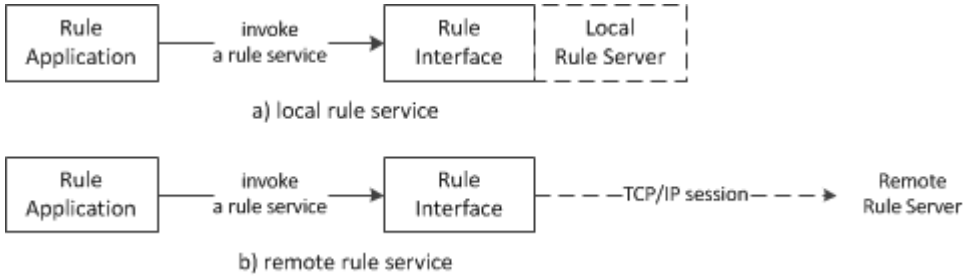

Figure 3 - 6 - 2 Rule Interface

Figure 3-6-2에서 보이듯이 룰 애플리케이션은 룰 인터페이스만을 이용하여 룰 서비스를 호출하며, 서비스의 형태에 의존적인 자원은 룰 애플리케이션에 대해 숨겨진다.

룰 서비스를 호출할 때마다 룰 인터페이스 및 연관되는 자원을 생성하게 되면 오버헤드가 발생한다. 예를 들어, 원격 룰 서비스를 호출할 때마다 룰 인터페이스를 생성하면 매번 새로운 TCP/IP 세션을 맺게 되어 성능이 저하될 것이다. **룰 인터페이스 클러스터(Rule Interface Cluster)**는 한 번 생성된 룰 인터페이스를 소멸시키지 않고 재사용을 할 수 있도록 해 주는 룰 런타임 구성 요소이다. 룰 인터페이스 클러스터는 다음의 기능을 수행한다.

- ❒ 룰 애플리케이션이 룰 인터페이스를 요청할 때, 재사용을 위해 저장된 룰 인터페이스가 있는 경우 이를 제공한다.
- ❒ 룰 애플리케이션이 룰 인터페이스를 요청할 때, 재사용을 위해 저장된 룰 인터페이스가 없는 경우 새로운 룰 인터페이스를 생성하여 제공한다.
- ❒ 룰 애플리케이션이 룰 인터페이스의 사용을 마치고 클러스터에 반환하면 이를 재사용을 위해 보관하거나 소멸시킨다.

룰 애플리케이션은 여러 개의 스레드를 이용하여 동시에 룰 서비스를 호출할 수 있기 때문에 룰 인터페이스 클러스터는 다수의 룰 인터페이스를 보관할 수 있다. 그러나, 보관된 룰 인터페이스의 개수가 평균적인 동시 룰 서비스 호출보다 많은 경우 여분의 룰 인터페이스는 대부분의 시간 동안 사용되지 않고 자원만 점유하게 된다. 원격 룰 서비스의 경우를 예로 들면, 클러스터에 10개의 룰 인터페이스가 보관되어 있는 경우 10개의 TCP/IP 세션이 존재하게 된다. 그러나, 평균적인 동시 룰 서비스 요청이 5건 정도라면 사용되지 않는 5개의 TCP/IP 세션은 자원의 낭비이다.

룰 애플리케이션이 동시에 호출하는 룰 서비스의 개수를 **동시 사용자 수(The Number of Concurrent Users)**라고 한다. 동시 사용자 수를 지정하여 룰 인터페이스 클러스터의 자원 사용의 한계와 성능을 조절할 수 있다. 룰 인터페이스에는 다음을 지정할 수 있다.

- ❒ 최대 동시 사용자 수(Maximum Number of Concurrent Users)
- ❒ 평균 동시 사용자 수(Average Number of Concurrent Users)

최대 동시 사용자 수는 룰 인터페이스 클러스터가 점유하는 자원의 최대값에 영향을 미친다. 아무리 많은 룰 서비스가 동시에 요청이 된다고 하더라도 룰 인터페이스는 최대 동시 사용자 수에 지정된 이상으로 생성되지 않는다. 그 이상의 룰 서비스 요청은 다른 룰 서비스가 처리될 때까지 대기하게 된다. 예를 들어, 원격 룰 서비스에 대해 최대 동시 사용자 수가 10으로 지정된 경우, 동시에 20건의 룰 서비스 요청이 들어오더라도 TCP/IP 세션은 최대 10개까지만 생성이 되고 10개의 룰 서비스만이 동시에 처리된다. 나머지 요청은 대기하며 다른 룰 서비스 처리가 완료되어 유휴 TCP/IP 세션이 생겼을 때 처리된다.

평균 동시 사용자 수는 재사용을 위해 보관할 룰 인터페이스의 개수를 지정하며 평균적인 자원 사용량

과 성능에 영향을 미친다. 룰 애플리케이션이 룰 인터페이스의 사용을 마치고 반환할 때, 현재 사용 중인 룰 인터페이스와 보관 중인 룰 인터페이스의 개수의 합이 평균 동시 사용자 수보다 큰 경우 룰 인터페이스는 소멸되고, 그렇지 않을 경우에는 재사용을 위해 클러스터에 보관된다. 룰 애플리케이션에 의해 사용되지 않고 클러스터에 보관되어 있는 룰 인터페이스를 **유휴 룰 인터페이스(Idle Rule Interface)**라고 한다. 평균 동시 사용자 수가 평균적인 부하에 비해 작게 설정된 경우, 잦은 룰 인터페이스 생성으로 성능이 저하된다. 그에 반해 너무 크게 설정된 경우 사용되지 않는 유휴 인터페이스로 인해 자원이 낭비된다.

3.6.7 TCP 커넥터 서버 그룹 선택 - 원격 룰 서비스에만 해당

원격 룰 서비스를 선택한 경우 서비스를 호출할 TCP 커넥터 서버를 지정해야 한다. 일련의 TCP 커넥터 서버들은 사용 목적별로 그룹화 되어 있는데 이를 TCP 커넥터 서버 그룹이라고 하며 룰 애플리케이션 설정에는 TCP 커넥터 서버 그룹을 지정하게 된다. TCP 서버 그룹에 대한 자세한 사항은 3.8.4.2를 참조하라.

3.6.8 업무 DB 접속 방법 설정

업무 DB의 목록과 접속 정보는 시스템 마법사를 이용하여 관리한다고 설명하였다. 룰 애플리케이션은 이 접속 정보를 이용하여 업무 DB 연결을 관리하는 커넥션 풀을 생성할 수도 있다. 그러나, 룰 애플리케이션이 등록된 접속 정보를 이용하지 말아야 할 경우가 있다. 대표적인 예로,

A. 룰 애플리케이션별로 접속해야 할 업무 데이터베이스가 상이한 경우. 예를 들어, 애플리케이션 환경이 온라인과 배치로 구분되어 있고 사용하는 데이터베이스 서버가 분리되어 있을 때, 각 애플리케이션에는 업무 DB에 대한 접속 정보가 다르게 지정되어야 한다.
B. 애플리케이션 또는 애플리케이션 프레임워크가 업무 DB에 대한 커넥션 풀을 직접 관리하는 경우. 이 경우 룰 애플리케이션은 별도의 커넥션 풀을 생성하는 것이 아니라 이미 존재하는 커넥션 풀을 참조해야 한다. 대표적인 사례가 웹 애플리케이션 환경으로, 룰 애플리케이션은 WAS에 생성된 업무 DB에 대한 데이터 소스를 참조해야 한다.

룰 애플리케이션이 다음 중 하나의 방법으로 업무 DB에 대한 연결을 얻어 오도록 설정할 수 있다.

❒ 전용 커넥션 풀 방식

룰 애플리케이션은 업무 DB에 대한 커넥션 풀 데이터 소스를 생성하고 이로부터 데이터베이스 연결을 얻어온다. DB 접속 정보는 시스템 마법사에 등록된 접속 정보가 기본으로 사용되지만, 직접 접속 정보를 지정할 수도 있다. 위의 예외 상황 A의 경우, 전용 커넥션 풀 방식을 이용하여 직접 접속 정보를 입력하여 해결할 수 있다.

❒ 데이터 소스 참조 방식

JNDI를 이용하여 네이밍 시스템에서 데이터 소스를 검색하여 데이터베이스 연결을 얻어온다. 3.6.5의 룰 DB에 대한 데이터 소스 참조 방식과 같은 방법으로 데이터 소스의 이름을 지정해야 하며, 옵션으로 이니셜 컨텍스트 팩토리와 프로바이더 URL을 지정할 수 있다.

룰 애플리케이션이 웹 애플리케이션인 경우, 웹 애플리케이션 배치 기술자에는 업무 DB에 대한 자원 참조 이름이 지정되어 있어야 하고, 이 이름을 외부 DB의 데이터 소스 이름으로 지정해야 한다. 위의 예외 상황 B의 경우, 데이터 소스 참조 방식을 사용하여 해결할 수 있다.

룰 애플리케이션 설정에 시스템 마법사로 등록한 모든 업무 DB에 대한 접속 방법을 지정해야 할 필요는 없다. 애플리케이션이 호출하는 룰들이 어떤 업무 DB들에 접근하는지 알고 있다면, 그 DB들에 대해서만 접속 방법을 지정하면 된다. 그러나, 그렇지 못할 경우 모든 업무 DB에 대한 접속 방법을 지정하기를 권장한다. 만약, 룰을 실행하는 도중 접속 방법이 지정되지 않은 업무 DB를 접근하게 될 경우 접속 방법이 지정되지 않았다는 에러가 발생하게 된다.

3.7 룰 애플리케이션 초기화

애플리케이션이 룰 서비스를 사용하기 위해서는 룰 애플리케이션 초기화가 이루어져야 한다. 룰 애플리케이션 초기화는 룰 애플리케이션 설정을 이용하여 룰 런타임을 구성하는 과정으로서, 애플리케이션이 사용할 룰 애플리케이션 설정 이름을 인수로 하여 룰 애플리케이션 초기화 루틴(3.7.1)을 호출하는 과정이다. 초기화 루틴이 룰 시스템 설정 파일과 설치 디렉터리의 여러 파일들을 접근하기 때문에 룰 애플리케이션은 이 파일들에 접근 가능해야 하며 읽기 권한이 있어야 한다. 하나의 룰 애플리케이션 설정으로 여러 애플리케이션을 초기화 할 수도 있다.

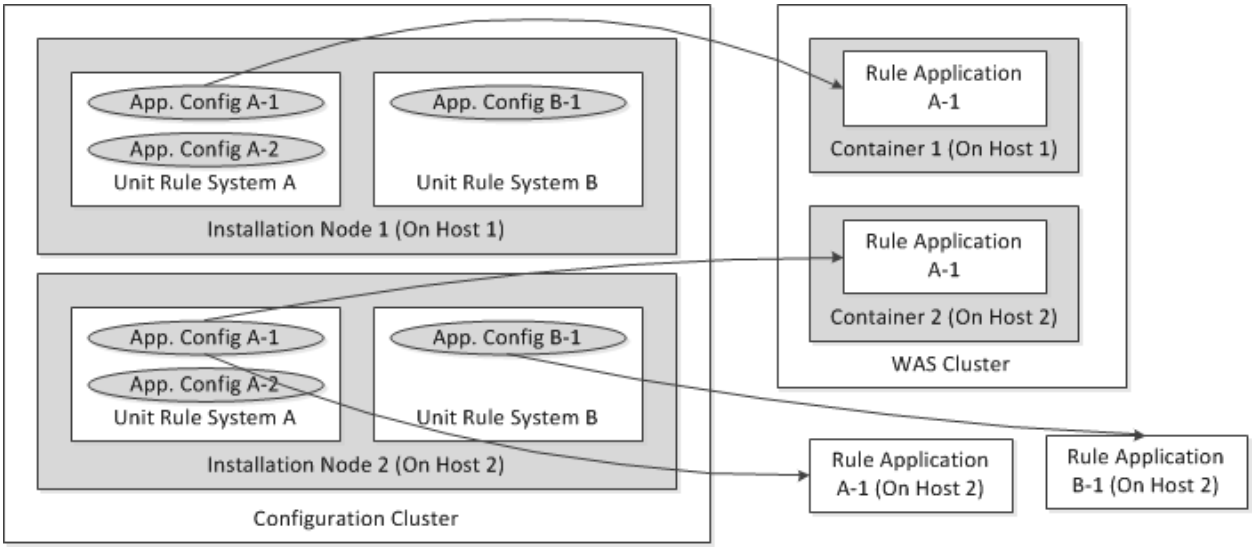

Figure 3 - 7 - 1 Examples of Rule Application Initializations

Figure 3-7-1은 룰 애플리케이션 설정과 이를 이용하여 초기화 된 애플리케이션의 예이다. 그림의 좌측은 InnoRules가 어디에 설치되었고 어떻게 설정이 되었는가를 보여준다. 두 개의 설치 노드가 있으며, 이들은 각각 호스트 1과 호스트 2에 설치되었다. 이 두 개의 설치 노드는 설정 클러스터로 묶여 있고 동일한 설정이 저장된다. 이 설정 클러스터에는 단위 룰 시스템 A와 단위 룰 시스템 B가 구성되어 있고, 단위 룰 시스템 A에는 룰 애플리케이션 설정으로 A-1과 A-2가 존재하며 단위 룰 시스템 B에는 룰 애플리케이션 설정 B-1이 각각 존재한다.

그림의 우측 상단은 WAS 클러스터를 보여준다. 두 개의 웹 애플리케이션 컨테이너가 WAS 클러스터로 묶여 있다. 컨테이너 1은 호스트 1에 설치되어 있으므로 같은 호스트의 파일 시스템에 위치한 설치 노드 1의 설정만 접근이 가능하고 컨테이너 2는 같은 이유로 설치 노드 2의 설정만 접근이 가능하다. 웹 애플리케이션들은 클러스터로 묶여 있으므로 모든 설정들이 동일해야 한다. 비록 각 웹 애플리케이션이 접근하는 룰 시스템 설정 파일은 다른 호스트에 위치하지만 설정 클러스터에 의해 그 내용이 동기화 되므로 룰 웹 애플리케이션들은 동일한 설정으로 초기화 될 수 있다. 그림은 WAS 클러스터에 배치된 두 웹 애플리케이션이 "A-1"이라는 룰 애플리케이션 설정을 이용하여 초기화 된 것을 보여준다. 여기서, 컨테이너 1의 애플리케이션은 설치 노드 1의 설정 파일을, 컨테이너 2의 애플리케이션은 설치 노드 2의 설정 파일을 접근하였다.

그림의 우측 하단에는 호스트 2에서 동작하는 또 다른 룰 애플리케이션들을 보여준다. 이 애플리케이션들은 평범한 자바 애플리케이션으로서 웹 애플리케이션이 아니다. 좌측의 애플리케이션은 다른 웹 애플리케이션이 사용하고 있는 "A-1" 설정을 이용하여 초기화 되었다. 이와 같이 하나의 룰 애플리케이

션 설정을 다른 애플리케이션이 공유하여 사용 가능하다. 우측의 애플리케이션은 단위 시스템 B의 "B-1" 애플리케이션 설정을 사용하고 있다. 이와 같이 하나의 호스트에서 서로 다른 단위 시스템의, 서로 다른 설정을 사용하는 룰 애플리케이션을 실행할 수 있다.

3.7.1 룰 애플리케이션 초기화 루틴

룰 애플리케이션 초기화 루틴(Rule Application Initialization Routine)이란 룰 애플리케이션 설정을 읽어 들여 룰 애플리케이션 환경에 룰 런타임을 구성하거나 구성되어 있는 룰 런타임 제거하는 자바 코드이다. 이 코드들은 innorules-core.jar 라이브러리에 포함되어 있다. 이 라이브러리는 *INNORULES_INSTALLATION*/lib 디렉터리에 위치한다.

초기화 루틴에 대한 자바 언어 수준의 설명은 Rule Application Programming Interface Guide를 참조하라. 본 절에서는 초기화와 클린업 메소드에 전달되는 인자들에 대해서만 설명한다.

3.7.1.1 초기화 메소드

초기화 메소드는 룰 애플리케이션에 룰 런타임을 구성해 주는 메소드로서 다음의 정보들을 인자로 받는다.

- ❒ InnoRules 설치 디렉터리(필수)
- ❒ 단위 룰 시스템 ID(필수)
- ❒ 룰 애플리케이션 설정 이름(필수)
- ❒ 라이브러리 템플릿 이름(선택)

InnoRules 설치 디렉터리는 InnoRules가 설치된 디렉터리를 말한다. 한 호스트에 InnoRules가 여러 노드 설치되었을 경우, 사용할 룰 애플리케이션 설정이 있는 설치 노드의 디렉터리를 지정하면 된다.
설치 디렉터리로서 네트워크 파일 시스템의 경로를 지정할 수도 있다. InnoRules 설치 디렉터리에는 설정 파일, 라이브러리뿐만 아니라 라이선스 정보도 위치하고 있으며, 이 라이선스에는 허용되는 IP 주소 등 네트워킹 관련 제약사항이 포함되었을 수도 있다. 따라서, 설치 노드의 네트워크 주소와 룰 애플리케이션의 네트워크 주소가 다른 경우 적절한 라이선스가 없다면 정상적으로 실행되지 않을 수도 있다.

단위 룰 시스템 ID는 애플리케이션이 속한 단위 룰 시스템의 이름이다. 선택된 단위 룰 시스템의 룰이 서비스 된다.

룰 애플리케이션 설정 이름은 애플리케이션에 적용할 룰 애플리케이션 설정의 이름이다. 이 이름은 선택된 단위 룰 시스템에 존재하는 설정의 이름이어야 한다.

라이브러리 템플릿은 룰 런타임을 구성할 때 참조할 라이브러리들의 목록 파일이다. 이 템플릿들은 *INNORULES_INSTALLATION*/lib에 위치하며 확장자는 .properties로서 자바 프로퍼티즈 파일 형식을 갖고 있다. **라이브러리 템플릿의 이름**은 템플릿 파일 이름에서 .properties를 제외한 나머지 이름이다. InnoRules가 설치될 때 기본적으로 두 개의 템플릿 파일, rule-service.properties와 innorules-service.properties가 생성된다.

룰 애플리케이션의 클래스 경로에는 innorules-api.jar와 innorules-core.jar만 등록하면 된다. 이 라이브러리들에는 룰 서비스 API와 룰 애플리케이션 초기화 루틴 등 룰 애플리케이션을 작성하는 데에 필요한 클래스와 인터페이스들이 모두 포함되어 있다. 그러나, 룰 런타임이 실행되기 위해서는 더 많은 클래스들과 서브 파티 라이브러리를 필요로 한다. 라이브러리 템플릿 파일은 룰 런타임이 사용할 라이브러리들을 기술한 파일이다. 이 추가적인 클래스들은 초기화 루틴이 생성한 전용 클래스로더에 의해 로딩된다.

rule-service.properties는 룰 서비스를 위한 기본 라이브러리 템플릿이다. 이 파일에는 다음의 라이브러리들이 나열되어 있다.

```
commons-logging-1.2.jar=
log4j-api-2.21.0.jar=
log4j-core-2.21.0.jar=
axiom-api-1.4.0.jar=
axiom-impl-1.4.0.jar=
jakarta.activation-api-1.2.2.jar=
netty-3.10.4.jar=
trove-3.0.3.jar=
innoutils.jar=com.innoexpert.utility.Release
innorulesj.jar=com.innoexpert.innorulesj.engine.Release
configuration-wizard.jar=com.innorules.configurator.initializer.ConfigurationConverter
innorules-extra.jar=com.innoexpert.extra.Release
log4j2-intf.jar=com.innorules.log.impl.log4j2.Release
```

log4j-core-2.21.0.jar, innoutils.jar와 같이 프로퍼티의 키에는 라이브러리의 경로가 지정된다. 경로는 *INNORULES_INSTALLATION*/lib에 대한 상대 경로를 지정하거나 절대 경로를 지정할 수 있다. 초기화 루틴은 템플릿에 지정된 파일들을 소스로 하는 클래스로더를 생성한다. 애플리케이션의 클래스 로더가 생성된 클래스 로더의 상위가 된다. 초기화 루틴은 생성된 클래스 로더를 이용하여 룰 런타임을 초기화 한다.
프로퍼티의 값은 라이브러리가 정상적으로 로드 되었는지 확인하기 위한 테스트 클래스이다. 초기화 루틴은 클래스 로더를 생성한 후에 테스트 클래스가 정상적으로 적재되는지 확인한다. 정상적으로 적재되지 않은 경우 예외가 던져진다. 테스트 클래스는 생략할 수 있다.

비록 rule-service.properties의 라이브러리 구성이 룰 서비스에 필요한 기본적인 구성이기는 하지만 경우에 따라 다른 구성을 사용해야 할 경우도 있다. 다음과 같은 경우이다.

- ❒ 애플리케이션의 클래스 경로에 동일한 라이브러리가 이미 존재하는 경우
  애플리케이션의 클래스 경로에 log4j와 같은 서드 파티 라이브러리가 등록되는 경우가 종종 있다. 라이브러리의 충돌을 방지하기 위해 중복되는 라이브러리들을 템플릿에서 제거해야 할 경우도 있다.
- ❒ 애드온 라이브러리
  룰 런타임에는 사용자 정의 함수, 빌더 서비스 모듈 등의 다양한 애드온을 추가할 수 있다. 애드온으로 인해 추가적인 라이브러리가 필요한 경우 템플릿에 라이브러리들을 추가해 줘야 한

다.

위의 이유들로 라이브러리 구성이 변경된다 하더라도 rule-service.properties를 직접 수정하는 것은 권장되지 않으며, 새로운 템플릿 파일을 생성하여 변경된 라이브러리 구성을 기술하는 것이 좋다. 초기화 메소드에 라이브러리 템플릿 이름을 지정하여 룰 애플리케이션이 어떤 라이브러리 구성을 사용할 것인지 지정할 수 있다. 지정하지 않은 경우, rule-service.properties가 사용된다.

3.7.1.2 클린업 메소드

룰 애플리케이션은 클린업 메소드를 호출하여 룰 런타임이 점유하고 있는 자원을 해제할 수 있다. 클린업 이후에는 더 이상 룰 서비스를 호출할 수가 없다. 클린업 이후에도 다시 초기화 메소드를 호출하여 룰 런타임을 구성할 수도 있다. 그렇지만, 애플리케이션이 시작될 때 초기화를 하고 애플리케이션이 종료되기 전에 클린업을 하기를 권장한다.

3.7.1.3 사용 가능한 애플리케이션 형태

초기화 루틴은 모든 형태의 자바 애플리케이션에서 사용할 수 있다. 초기화를 호출하고 클린업을 호출하는 시점과 호출하는 코드는 애플리케이션에 의존적이다.

public static void main(String[] args)으로부터 작성해야 하는 애플리케이션의 경우, main 메소드의 시작 부분에서 초기화를 하고 종료되기 직전에 클린업을 하는 것이 좋다.

웹 애플리케이션의 경우 서블릿 컨텍스트 리스너(servlet context listener)를 사용할 수 있다. 기존에 웹 애플리케이션에 등록되어 있는 리스너를 이용하고자 한다면 contextInitialized 메소드에 초기화 루틴 호출코드를, contextDestroyed 메소드에 클린업 루틴 호출 코드를 삽입한다. 만약, 룰 애플리케이션 초기화 용도로 새로이 리스너를 작성하려고 한다면 다음의 과정을 따른다.

- ❒ 서블릿 컨텍스트 리스너 클래스를 생성하고 contextInitialized 메소드에서 초기화 루틴을, contextDestroyed 메소드에서 클린업 루틴을 호출하도록 작성한다.
- ❒ 작성된 클래스를 웹 애플리케이션의 클래스 경로에 위치시킨다.
- ❒ 웹 애플리케이션 배치 기술자(Web Application Deployment Descriptor)에 리스너를 등록한다.
- ❒ 재배포 등의 방법으로 변경 사항을 적용한다.

애플리케이션의 형태는 매우 다양하기 때문에 여기서 모든 케이스를 언급할 수는 없다. 또한, 동일한 형태의 애플리케이션이라고 하더라도 초기화 루틴을 호출하는 방법이 한 가지만 있는 것이 아니다. 따라서, 초기화 루틴의 호출 시점과 호출 코드의 위치는 애플리케이션의 형태와 목적에 따라 개발자가 적절히 선택해야 한다.

3.7.2 룰 애플리케이션 초기화 리스너

룰 애플리케이션이 웹 애플리케이션인 경우, 빌트인 초기화 리스너를 이용하여 초기화 루틴을 호출할 수 있다. 초기화 리스너 클래스의 이름은 다음과 같다.

com.innorules.rrt.RuleAppInitializerListener

이 리스너는 javax.servlet.ServletContextListener를 구현하고 있으며, contextInitialized 메소드에서 초기화 루틴을, contextDestroyed 메소드에서 클린업 루틴을 호출하도록 되어 있다. 웹 애플리케이션이 시작 및 종료될 때 리스너가 호출되도록 web.xml에 설정함으로써 룰 애플리케이션 초기화를 할 수 있다. 이 리스너는 innorules-core.jar에 포함되어 있다.

초기화에 필요한 정보들은 컨텍스트 파라미터에 지정해야 한다. 파라미터들은 다음과 같다.

- ❒ innorules.home
  (필수) InnoRules 설치 디렉터리
- ❒ innorules.system
  (필수) 단위 룰 시스템
- ❒ innorules.rule-application-config
  (필수) 룰 애플리케이션 설정 이름
- ❒ innorules.library-template
  (선택) 라이브러리 템플릿 이름. 지정하지 않은 경우, "rule-service"가 설정된다.

웹 애플리케이션의 클래스 경로에는 innorules-api.jar와 innorules-core.jar를 추가해야 한다. 이 라이브러리들은 *INNORULES_INSTALLATION*/lib에 위치하고 있다. 이 파일들을 복사하여 웹 애플리케이션의 클래스 경로에 복사해도 되나, 가급적이면 이 디렉터리의 라이브러리를 직접 참조하는 것이 업그레이드나 패치와 같은 유지보수에 용이하다. 만약, 이 라이브러리들을 복사하여 사용해야 한다면 유지보수에서 누락되지 않도록 적절한 관리 방안을 강구해야 한다.

다음은 web.xml에 룰 애플리케이션 초기화 리스너를 설정한 예이다.

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
 2 <!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
 3 <web-app>
 4     <display-name>Sample Rule Web Application</display-name>
 5
       ...
 6     <context-param>
 7         <param-name>innorules.home</param-name>
 8         <param-value>/opt/innorules-home</param-value>
 9     </context-param>
10     <context-param>
11         <param-name>innorules.system</param-name>
12         <param-value>PRODUCTION</param-value>
13     </context-param>
14     <context-param>
15         <param-name>innorules.rule-application-config</param-name>
```

```
16            <param-value>online</param-value>
17        </context-param>
18        <context-param>
19            <param-name>innorules.library-template</param-name>
20            <param-value>rule-service</param-value>
21        </context-param>
          ...
22         <listener>
23              <listener-class>com.innorules.rrt.RuleAppInitializerListener</listener-class>
24         </listener>
          ...
25 </web-app>
```

위의 예제에서 굵은 글씨로 표시된 부분은 여러분의 환경에 맞게 변경해야 하는 부분이다. 6~21라인은 컨텍스트 파라미터를 설정하는 부분이다. 6~9라인에서는 InnoRules 설치 디렉터리를 /opt/innorules-home으로 설정하고 있다. 10~13라인에서는 이 애플리케이션이 속한 단위 룰 시스템의 ID를 지정하고 있다. 예제는 PRODUCTION 단위 룰 시스템의 서비스를 이용하도록 설정되었다. /opt/innorules-home의 설치 노드에 PRODUCTION 단위 룰 시스템이 설정되어 있어야 한다. 14~17라인에서는 이 애플리케이션이 online이라는 룰 애플리케이션 설정을 사용하도록 설정하고 있다. PRODUCTION 단위 룰 시스템에는 online이라는 룰 애플리케이션 설정이 존재해야 한다. 18~21라인에서는 라이브러리 템플릿으로 rule-service를 지정하고 있다. 이 애플리케이션은 rule-service.properties 파일에 지정된 자바 라이브러리들을 이용하여 초기화 된다.

만약 애플리케이션에 다른 컨텍스트 리스너들이 존재하고 이들과 룰 애플리케이션 초기화 리스너의 실행 순서를 조정하고 싶다면 22~24라인의 위치를 변경하여 순서를 조정한다. 서블릿 명세 3.0(Servlet Specification 3.0) 이상에서는 web.xml에 기술된 순서대로 컨텍스트 리스너가 호출된다고 되어 있다.

아래의 표는 룰 애플리케이션 초기화 리스너인 RuleAppInitializerListener 클래스를 간략하게 보여준다. 룰 애플리케이션 초기화 루틴은 contextInitialized 메소드에서 호출한다고 하였다. 만약, 설정 또는 운영 환경의 문제로 인하여 초기화가 실패한 경우 리스너는 onError 메소드를 호출한다.

```
package com.innorules.rrt;
...

public class RuleAppInitializerListener implements ServletContextListener
{
    public void contextInitialized( ServletContextEvent event );
    public void contextDestroyed( ServletContextEvent event );
    protected void onError( Exception e );
}
```

onError 메소드는 기본적으로 표준 에러 출력에 예외 정보를 출력한다. 만약, 초기화 에러가 발생하였을 때 다른 처리를 하고 싶다면 RuleAppInitializerListner를 상속 받아 새로운 리스너를 작성하고 onError 메소드를 오버라이드 하여 원하는 처리를 수 있다. web.xml에는 새로이 작성된 리스터 클래스를 등록한다.

3.7.3 룰 애플리케이션 초기화 클래스

룰 애플리케이션이 웹 애플리케이션 초기화 리스너를 사용할 수 없는 경우, 기본으로 제공되는 룰 애플리케이션 초기화 클래스를 이용하여 룰 애플리케이션을 초기화 할 수 있다. 초기화 클래스의 이름은 아래와 같다.

com.innorules.rrt.Initializer

다음은 이 클래스를 사용하여 룰 애플리케이션을 초기화하고 클린업까지 호출하는 예제이다.

```
1 public static void main( String[] args )
 2 {
 3          Initializer init = null;
 4          try
 5          {
 6                    Properties props = new Properties();
 7                    props.put( "innorules.home", "/opt/innorules-home" );
 8                    props.put( "innorules.system", "DEV" );
 9                    props.put( "innorules.rule-application-config", "batch" );
10                    props.put( "innorules.library-template", "rule-service" );
11                    init = new Initializer( props );
12                    init.initialize( null );
13
14                    // rule call process
15          }
16          catch( Exception e )
17          {
18                    e.printStackTrace();
19          }
20          finally
21          {
22                    if( init != null && init.hasInitialized() )
23                              init.cleanup();
24          }
25 }
```

위 예제의 11라인에서 Initializer 클래스의 생성자에서 java.util.Properties 클래스를 인자로 사용하고 있다. 7~10 라인에서는 룰 런타임 시스템을 초기화하는데 필요한 인자를 Properties 클래스에 설정한다. 이때 사용되는 인자의 키 값은 3.7.2 룰 애플리케이션 초기화 리스너에서 설명한 4개의 파라미터와 동일한 값을 사용한다.

아래의 표는 룰 애플리케이션 초기화 클래스인 Initializer 클래스를 간략하게 보여준다.

```
package com.innorules.rrt;


public class Initializer
{
    public Initializer( Properties props ) throws Exception;
    public void initialize( Object appctx ) throws Exception;
    public void cleanup();
    public boolean hasInitialized();
}
```

Initializer클래스는 룰 애플리케이션 안에서 하나의 객체만 존재하여야 하며, 룰 애플리케이션 개발자는 애플리케이션이 시작하는 시점에 초기화를 하고, 애플리케이션이 종료하는 시점에 클린업을 호출하여야 한다. InnoRules에서는 룰 애플리케이션 개발자가 위 코드를 손쉽게 사용하도록 InitializerHelper 클래스를 제공하고 있다.

다음은 InitializerHelper 클래스를 사용하여 룰 애플리케이션을 초기화 하고 클린업까지 호출하는 예제이다.

```
1 public static void main( String[] args )
 2 {
 3        try
 4        {
 5                Properties props = new Properties();
 6                props.put( "innorules.home", "/opt/innorules-home" );
 7                props.put( "innorules.system", "DEV" );
 8                props.put( "innorules.rule-application-config", "batch" );
 9                props.put( "innorules.library-template", "rule-service" );
10                InitializerHelper.initialize( props );
11
12                // rule call process
13        }
14        catch( Exception e )
15        {
16                e.printStackTrace();
```

```
17          }
18
19          InitializerHelper.cleanup();
20 }
```

위 예제는 Initializer 클래스와 유사하지만 InitializerHelper 클래스에서는 Initializer 클래스를Singleton 방식으로 관리하고 있다. 개발자는 InitializerHelper 클래스를 사용함으로 룰 애플리케이션의 중복 초기화를 방지하고 서비스를 제공할 수 있다.

3.8 InnoRules 서비스

이 절에서는 InnoRules의 다양한 서비스들과 이 서비스를 제공하는 InnoRules 서버에 대해서 설명한다.

3.8.1 InnoRules 서버

InnoRules 서버는 룰 시스템과 관련된 다양한 서비스를 제공하는 서버이다. InnoRules 서버는 Apache Tomcat과 InnoRules 웹 애플리케이션으로 이루어져 있다. 서버는 제품을 설치할 때 자동으로 설치되며 설치 단계에서 HTTP 서비스 포트와 제어 포트를 설정하도록 되어 있다. 서버는 *INNORULES_INSTALLATION*/innorules-server에 설치된다. 이 디렉터리에는 다음의 하위 디렉터리가 있다.

- ❒ apache-tomcat-X
  Apache Tomcat의 바이너리가 위치하는 디렉터리.
- ❒ default
  InnoRules 웹 애플리케이션이 포함된 Apache Tomcat 기본 인스턴스. 이 디렉터리의 webapps/innorules에 InnoRules 웹 애플리케이션이 설치되어 있다.
- ❒ logs
  기본 로그 디렉터리. 제품을 설치할 때 로그 디렉터리를 다른 곳으로 지정하지 않았다면 이 디렉터리 하위의 system 디렉터리에, 지정하였다면 그 디렉터리 하위의 system 디렉터리에 서버의 로그가 기록된다

서버가 설치된다는 것이 그 서버가 실행된다는 것을 의미하지 않는다. 서버는 별도의 스크립트를 이용하여 실행해야 하며, 그 설치 노드에 서버를 실행할 필요가 없다면 실행하지 않아도 된다. 서버는 *INNORULES_INSTALLATION*/script 디렉터리의 start-server.sh과 stop-server.sh(또는 start-server.bat과 stop-server.bat) 스크립트를 이용하여 시작 또는 종료시킨다.

InnoRules 서버는 다음의 서비스를 제공할 수 있다.

- ❒ 관리 서비스
- ❒ 룰 빌더 서비스
- ❒ TCP 커넥터 서비스
- ❒ 룰 REST 서비스

서비스 마법사(Service Wizard)를 이용하여 위의 서비스들을 개별적으로 활성화 또는 비활성화 할 수

있다. 서버는 활성화된 서비스만을 제공한다.

시스템에 다수의 InnoRules 설치 노드가 있을 수 있고, 따라서 서버가 여러 개 있을 수 있다. 제품이 설치될 때 각 서버에는 UUID라는 고유한 이름이 부여된다. 이 이름을 이용하여 각 서버를 식별할 수 있다.

3.8.2 관리 서비스

관리 서비스(Management Service)는 설정 클러스터에 속한 룰 애플리케이션의 룰 런타임들의 상태를 모니터링 하고 관리하는 서비스이다. 이 서비스는 하나의 IP 주소와 포트[*5]를 점유하고 의 접속을 받아들이며 접속된 런타임들을 관리한다.

관리 서비스가 사용할 IP 주소와 포트는 관리 서비스를 활성화 할 때 설정한다. 관리 서비스를 활성화 할 호스트에 네트워크 어댑터가 여러 개 있는 경우 그 주소들 중 하나를 선택해야 한다. 이 주소는 설정 클러스터에 속한 모든 룰 런타임들이 접근할 수 있는 주소이어야 한다.

하나의 설치 노드에는 관리 서비스를 하나만 활성화 할 수 있지만, 설정 클러스터에는 다수의 설치 노드가 있을 수 있으므로 활성화 된 관리 서비스가 두 개 이상 있을 수 있다. 이 관리 서비스들에는 우선 순위가 부여된다. 룰 런타임들은 가용한 최우선 순위의 관리 서비스에 접속한다.

관리 서비스 활성화 및 비활성화와 우선 순위 변경에는 서비스 마법사가 이용된다. 단, 서비스 마법사는 마법사가 실행된 설치 노드의 관리 서비스에 대해서만 활성화, 비활성화 및 우선 순위 변경을 할 수 있다. 이 작업들을 하는 방법에 대해서는 서비스 마법사를 참조하라.

룰 애플리케이션들은 초기화 단계에 가장 우선 순위가 높은 관리 서비스에 접속을 시도한다. 접속이 실패한 경우 다음 순위의 관리 서비스에 접속을 시도한다. 이 과정에서 관리 서비스 중 하나에라도 접속을 하지 못해도 초기화가 실패하지는 않으며, 관리 서비스에 접속하지 못한 상태로 초기화는 진행이 된다. 가장 높은 순위의 관리 서비스에 접속하지 못한 경우 주기적으로 현재 접속 중인 관리 서비스보다 높은 순위의 관리 서비스들로 접속을 시도한다. 접속이 된다면 현재의 관리 서비스 연결을 끊고(연결이 맺어져 있다면) 새로이 접속된 서버와 통신한다.
룰 애플리케이션이 동작 중에 관리 서비스와의 연결이 끊어지더라도(서버, 네트워크 등의 장애 또는 서버 종료 등으로) 주기적으로 가장 높은 순위의 서버로부터 낮은 순위의 서버로 재접속을 시도한다.

Figure 3-8-1은 설정 클러스터에 관리 서비스들이 활성화되어 있는 것을 보여주고 있다. 설정 클러스터는 설치 노드 1, 설치 노드 2, 설치 노드 3의 세 개의 노드로 구성되어 있다. 관리 서비스는 노드 1과 3에만 활성화되어 있으며, 우선 순위는 노드 1, 노드 3 순이다. 그림과 같이 룰 애플리케이션들은 가장 우선 순위가 높은 노드 1의 관리 서비스에 접속하게 된다. 만약 노드 1의 관리 서비스가 종료된다면, 룰 애플리케이션들은 노드 3의 관리 서비스에 접속을 하게 될 것이며, 주기적으로 노드 1의 서버에 접속을 시도할 것이다.

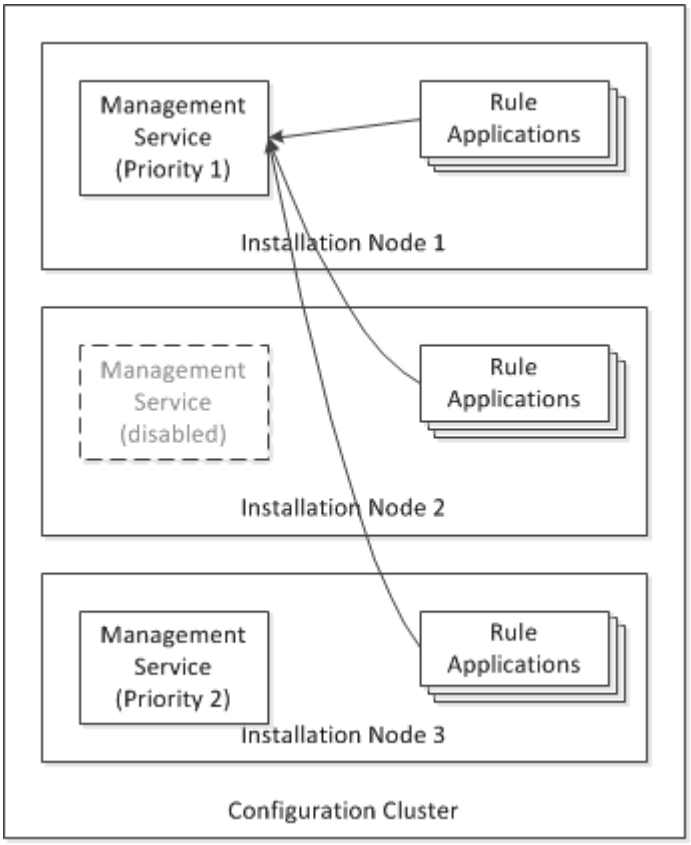

Figure 3 - 8 - 1 An Example of Management Service Configuration

관리 서비스에 접속된 룰 애플리케이션들의 상태는 관리 웹 페이지에 접속하여 확인할 수 있다. 관리 웹 페이지는 추후 제공될 예정이다.

*5. InnoRules 서버의 HTTP 서버의 바인딩 주소 및 포트와 별개이다.

3.8.3 룰 빌더 서비스

룰 빌더 서비스(Rule Builder Service)는 룰 요소들을 저작 및 관리하는 서비스이다. 룰 요소(Rule Element)란 룰, 항목, 룰 웹 서비스 등 룰 시스템에서 관리하는 데이터 요소를 말한다. 룰 빌더 서비스의 주된 기능은 다음과 같다.

- ❒ 룰 요소를 생성, 편집 및 삭제
- ❒ 룰 요소를 이관
- ❒ 작성된 룰을 테스트

3.8.3.1 모니터 서버와 모니터 세션

룰 데이터는 룰 빌더 서버에 의해서 갱신이 된다. Figure 3-8-2를 보자.

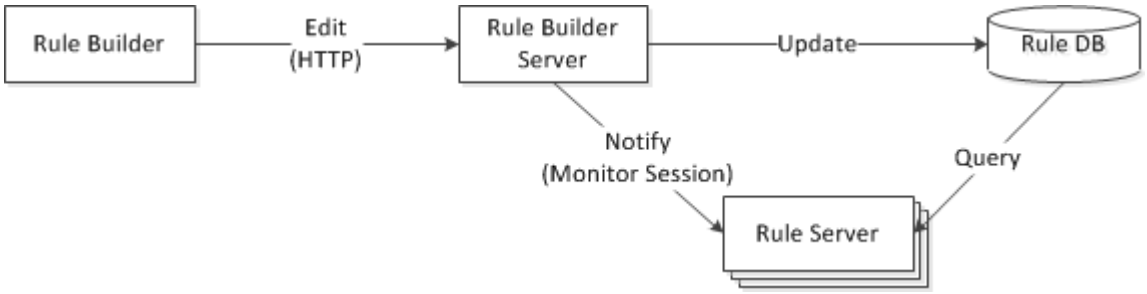

Figure 3 - 8 - 2 Monitor Session

룰 개발자가 룰 빌더를 이용하여 편집 작업을 하면 HTTP를 통하여 룰 빌더 서버에 서비스 요청이 전달된다. 편집 작업에는 룰 생성, 수정, 삭제뿐만 아니라 룰 요소들을 변경하는 모든 작업이 포함된다. 룰 빌더 서버는 서비스 요청을 분석하여 관련된 룰 DB 테이블들의 데이터를 적절히 갱신하게 된다.
그런데, 룰 DB를 접근하는 것은 룰 빌더 서버뿐만이 아니다. 룰 서버들도 룰 데이터를 조회한다. 룰 서버들은 TCP 커넥터 서버와 같이 원격 룰 서비스를 제공하는 서버일 수도 있고 룰 애플리케이션에 포함된 로컬 룰 서버일 수도 있다. 룰 서버들은 필요한 시점에 그때그때 룰 데이터를 조회하기도 하지만, 경우에 따라서 데이터가 변경될 때 실시간으로 조회를 해야 할 경우도 있다. 실시간으로 룰 데이터의 변경 목록을 주고받기 위하여 룰 빌더 서버와 룰 서버 사이에 TCP/IP 연결을 맺게 되는데, 이를 **모니터 세션(Monitor Session)**이라고 한다.

룰 서버들은 룰 실행 성능을 향상시키기 위해 룰 데이터의 일부분을 메모리에 캐시를 한다. 따라서, 룰 데이터가 갱신되었을 경우 메모리의 캐시도 같이 갱신되어야 한다. 그렇지 못했을 경우 룰 서버는 변경된 데이터로 서비스하지 못하게 된다.

룰 서버는 시작될 때 룰 빌더 서버로 모니터 연결을 시도한다. 룰 빌더 서버는 모니터 연결을 받아들이고 연결된 세션들을 관리하는 기능을 갖고 있다. 이 기능을 **모니터 서버(Monitor Server)**라고 한다. 모니터 서버는 이 연결 전용으로 TCP/IP 주소와 포트를 열고 대기한다. 이 주소와 포트는 룰 빌더 서비스를 활성화 할 때 지정된다.
룰 빌더 서버는 룰 데이터를 갱신하고 커밋한 후 연결된 모든 모니터 세션으로 변경된 목록을 통지한다. 통지를 받은 룰 서버들은 룰 DB로부터 변경된 내역을 조회하고 필요하다면 캐시를 갱신한다. 룰 서버는 변경을 통지한 이후, 룰 서버들이 그 내역을 모두 갱신할 때까지 추가적으로 룰 DB를 변경하지 않는다.

3.8.3.2 빌더 서버 클러스터링

하나의 설정 클러스터에 다수의 룰 빌더 서비스들을 활성화할 수 있다. 활성화된 빌더 서버들 사이에는 우선 순위가 부여된다. 우선 순위는 서비스 마법사를 이용하여 변경할 수 있다.

클러스터 내에 여러 개의 빌더 서버가 활성화되어 있다고 하더라도 가용한 빌더 서버 중에서 가장 우선 순위가 높은 빌더 서버만이 서비스를 제공한다. "가용한"의 의미는 서비스를 제공할 수 있는 상태라는 것을 의미한다. 우선 순위가 가장 높다고 해도 장애 또는 서버 정지 등으로 서비스를 제공할 수 없는 빌더 서버는 순위에서 제외된다. 설정 클러스터 내의 "가용한 최우선 순위의 빌더 서버"를 **마스터**

빌더 서버(Master Builder Server)라고 한다. 그에 반해, 서비스를 제공 가능한 상태에 있으나 마스터 빌더 서버보다 우선 순위가 낮은 빌더 서버들을 **슬레이브 빌더 서버(Slave Builder Server)**라고 한다.

마스터 빌더 서버로 들어오는, 즉 마스터 빌더 서버의 서비스 URL로 들어오는 빌더 서비스 요청은 이 빌더 서버가 처리한다. 그러나, 슬레이브 빌더 서버는 자신에게 들어오는 서비스 요청을 마스터 빌더 서버의 서비스 URL로 재전송(redirect) 한다. 마스터 빌더 서버에서만 모니터 서버가 활성화된다. 클러스터 내의 모든 룰 서버들은 마스터 빌더 서버의 모니터 서버로 모니터 세션을 맺게 된다.
다수의 빌더 서버들이 서비스 요청을 재전송하여 마스터 빌더 서버가 서비스를 처리하도록 하는 구성을 **빌더 서버 클러스터링(Builder Server Clustering)**이라고 한다. 빌더 서버 클러스터링의 목적은 하나의 빌더 서버만이 룰 DB를 변경하게 함으로써 룰 서버들이 룰 데이터 변경에 대한 정합성을 유지할 수 있도록 하는 데에 있다.

마스터 빌더 서버가 중지되거나 장애가 발생하여 서비스를 제공할 수 없는 상태가 되면 그 다음 순위의 슬레이브 빌더 서버가 마스터로 전환된다. 새로운 마스터 빌더 서버는 모니터 서버를 활성화한다. 기존의 마스터 빌더 서버의 장애로 인해 모니터 세션이 끊어지면, 룰 서버들은 새로운 마스터 빌더 서버로 모니터 세션을 연결한다.

마스터 빌더 서버보다 우선 순위가 높은 빌더 서버가 서비스를 재개하게 되면 이 서버가 마스터 빌더 서버가 되며 기존의 마스터 빌더 서버는 슬레이브로 전환되고 모니터 서버가 중지된다. 모니터 서버는 새로운 마스터 빌더 서버에서 활성화되며 룰 서버들은 그 서버로 모니터 세션을 맺는다.

Figure 3-8-3는 룰 빌더 서버 클러스터링의 예를 보여준다.

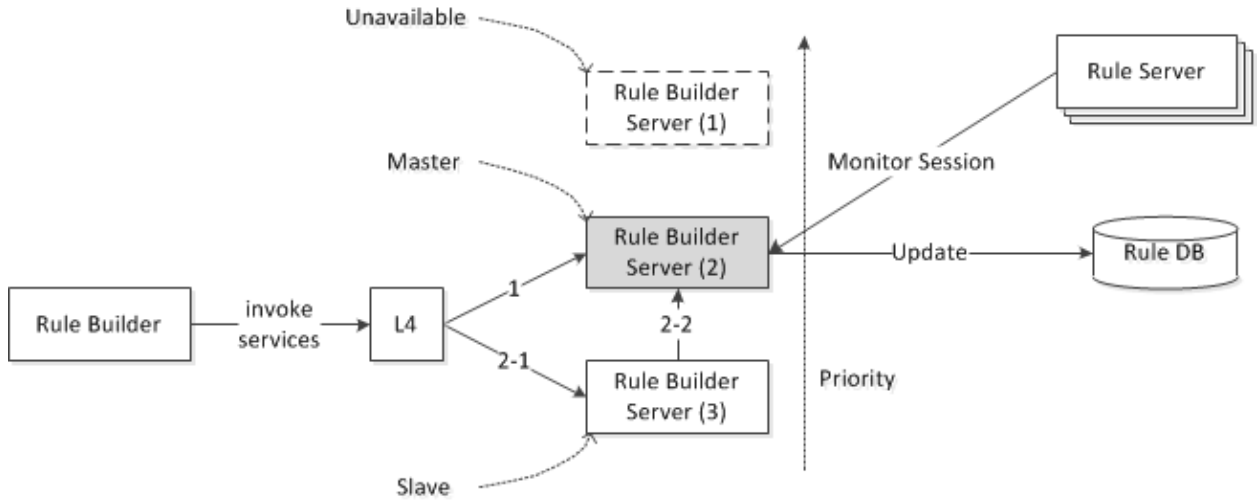

Figure 3 - 8 - 3 An example of Rule Builder Server Clustering

이 설정 클러스터에는 세 개의 룰 빌더 서버가 활성화되어 있다. 우선 순위는 Rule Builder Server 1, Rule Builder Server 2, Rule Builder Server 3의 순이다. Rule Builder Server 1은 우선 순위는 가장 높지만, 정지되어 있는 상태이다. 따라서, 현재 상태에서 마스터 룰 빌더 서버는 Rule Builder Server 2이다. Rule Builder Server 2에는 모니터 서버가 실행되며 모든 룰 서버는 여기로 모니터 세션을 맺는다.

룰 빌더는 특정 빌더 서버의 서비스 URL로 직접 서비스 요청을 하기도 하지만, 통상적으로 이러한 다중화 구성에서는 L4 스위치가 빌더 서버의 앞에 위치하고, 룰 빌더는 L4의 가상 URL로 서비스 요청을 한다. L4는 정책에 따라 Rule Builder Server 2 또는 3으로 요청을 재전송한다.
Rule Builder Server 2로 요청이 들어온 경우(화살Table1) 그 요청은 Rule Builder Server 2에서 처리된다. Server 2는 룰 DB를 업데이트하고 Rule Server들에 그 내역을 통지한다.
Rule Builder Server 3으로 요청이 들어온 경우(화살Table2-1) 슬레이브인 Rule Builder Server 3은 요청을 다시 Rule Builder Server 2로 재전송 한다(화살Table2-2). Rule Builder Server 2는 요청을 처리한 후에 Rule Builder Server 3에 결과를 반환하고, 이 결과는 다시 Rule Builder로 반환된다.

3.8.3.3 다단계 룰 시스템과 이관 전용 모드

하나의 설정 클러스터에는 하나의 마스터 빌더 서버만이 존재한다. 설정 클러스터에 속한 모든 룰 서버들은 마스터 빌더 서버에 모니터 세션을 맺는다. Figure 3-8-4은 이러한 사례를 보여준다.

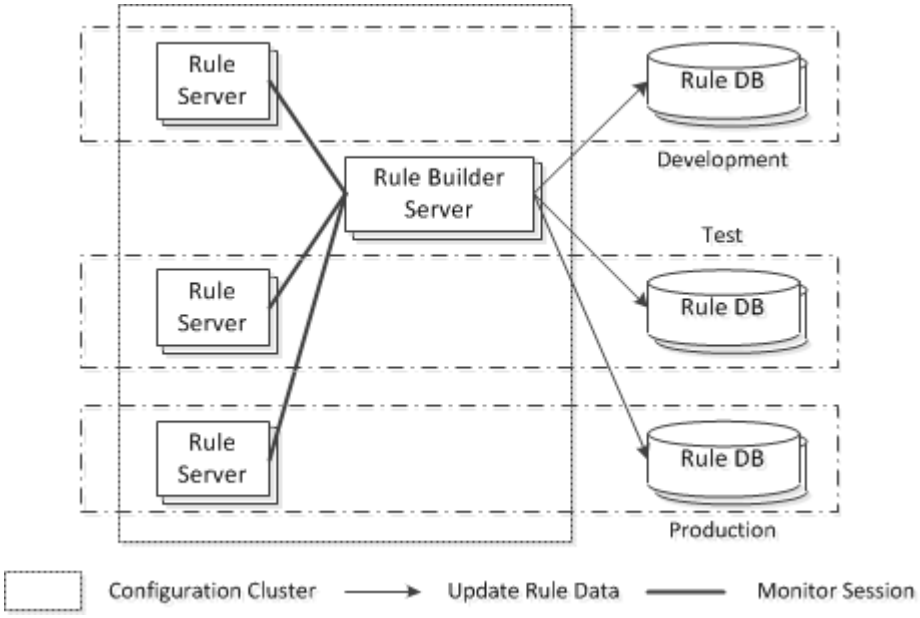

Figure 3 - 8 - 4 Single Configuration Cluster

Figure 3-8-4은 세 개의 애플리케이션 단계를 갖는 시스템을 보여주고 있다. 애플리케이션 단계는 개발(Development), 테스트(Test), 운영(Production)로 이루어져 있다. 룰 시스템은 다음과 같이 설치 및 설정되어 있다.

- ❒ 각 단계는 여러 개의 물리적인 서버로 구성될 수 있다. 개발 단계에는 여러 대의 개발 서버가 있으며, 이는 테스트 및 운영 단계도 마찬가지이다. 이 서버들에는 각각 InnoRules가 설치되어 있다. 모든 설치 노드는 하나의 설정 클러스터로 묶여 있다.
- ❒ 애플리케이션 단계에 대응하도록 단위 룰 시스템 개발계, 테스트계, 운영계가 설정이 되어 있다. 룰 DB는 단위 룰 시스템에 대응되는 단계의 서버 상에 위치한다. 예를 들면, 개발계의 룰 DB는 개발 서버 상에 존재한다.
- ❒ 각 단위 룰 시스템에는 룰 애플리케이션 설정들이 등록되어 있다. 룰 애플리케이션들은 이 설정들을 이용하여 각 단계의 물리적인 서버들에서 실행이 된다.

- ❒ 하나 이상의 설치 노드에 룰 빌더 서버가 활성화되어 있다. 룰 빌더 서버는 세 개의 단위 룰 시스템의 룰 DB 모두에 연결을 하고 업데이트를 한다.
- ❒ 룰 애플리케이션에 임베드된 룰 서버 또는 TCP 커넥터 서버와 같은 원격 룰 서버는 마스터 빌더 서버에 모니터 연결을 맺고 있다.

이 구성은 간단하게 설치하고 설정할 수 있는 구성이지만 시스템 운영 정책에 따라 제약들이 있을 수 있다.

첫째로 모니터 세션과 관련된 제약이다. 룰 서버들은 모니터 세션을 위해 마스터 빌더 서버에 TCP/IP 연결을 맺는다. 만약 마스터 빌더 서버가 개발 서버에서 실행된다면 테스트와 운영의 룰 서버들은 개발 서버로 TCP/IP 연결을 맺어야 한다. 반대로, 마스터 빌더 서버가 운영 서버에서 실행된다면 개발과 테스트의 룰 서버들은 운영 서버로 TCP/IP 연결을 맺어야 한다. 일반적으로 단계를 넘어가는 TCP/IP 연결에는 제약이 많고, 하위 단계에서 상위 단계로의 연결은 더욱 제약이 심하다(예: 테스트→운영).

둘째로 데이터베이스 연결과 관련된 제약이다. 빌더 서버는 모든 단계의 룰 DB에 연결하고 업데이트 해야 한다. 예를 들어, 운영 서버에서 실행되는 마스터 빌더 서버는 개발과 테스트의 룰 DB에 연결을 맺고 업데이트 해야 한다. 모니터 연결과 마찬가지로 단계를 넘어가는 DB 연결에 대한 제약이 있을 수 있다. 개발 서버에서 운영 DB로의 연결은 더더욱 제약이 심할 것이며, 이로 인해 개발 서버에 빌더 서버를 운영하는 것은 불가능할 개연성이 높다.

셋째로 설정 동기화에 제약이 있을 수 있다. 마법사를 이용하여 설정을 변경할 때, 마법사는 SSH나 FTP를 이용하여 다른 설치 노드의 설정을 같이 변경하여 동기화 한다. 그러나, 단계를 넘어가는 SSH 또는 FTP 접속에 제약이 있는 경우 설정 동기화가 불가능할 수 있다.

위의 문제점을 극복하기 위해, 각 단계별로 설정 클러스터를 분리하는 것을 고려해 볼 수 있다. 이 경우 설정은 각 단계별로만 동기화된다. Figure 3-8-5이 그 구성을 보여준다.

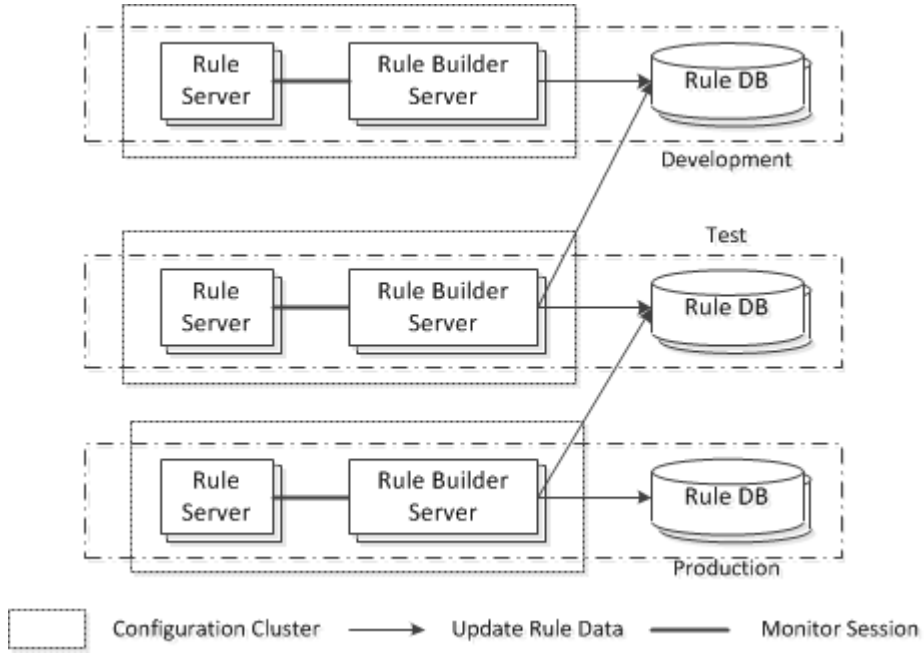

Figure 3 - 8 - 5 Separated Configuration Cluster

Figure 3-8-4와의 차이점은 단계별로 설정 클러스터를 분리했다는 점이다. 이 구성의 특징은 다음과 같다.

- ❒ 각 단계의 설정 클러스터는 그 단계의 서버들에 설치된 노드들로만 구성되어 있다.
- ❒ 설정 클러스터에는 하나 또는 연속되는 단위 룰 시스템 두 개가 등록되어 있다.
  - ○ 단위 룰 시스템은 이관 방향을 기준으로 개발계, 테스트계, 운영계 순이다.
  - ○ 개발 설정 클러스터에는 단위 룰 시스템으로 개발계가 등록되어 있다.
  - ○ 테스트 및 운영 설정 클러스터에는 단위 룰 시스템으로 각각 개발계와 테스트계, 테스트계와 운영계가 등록되어 있다.
- ❒ 인접한 설정 클러스터들은 하나의 단위 룰 시스템이 공통적으로 등록되어 있다.
  - ○ 개발 클러스터와 테스트 클러스터에는 개발계가 공통적으로 등록되어 있다.
  - ○ 테스트 클러스터와 운영 클러스터에는 테스트계가 공통적으로 등록되어 있다.
- ❒ 각각의 설정 클러스터에는 룰 빌더 서버들이 별도로 실행된다.
  - ○ 단위 룰 시스템이 하나만 등록된 클러스터의 룰 빌더 서버는 그 시스템의 룰을 편집한다 (ex. 개발 클러스터).
  - ○ 단위 룰 시스템이 두 개 이상 등록된 클러스터의 룰 빌더 서버는 앞쪽의 단위 룰 시스템의 룰을 편집, 뒤쪽의 단위 룰 시스템에는 이관을 한다(테스트 및 운영 클러스터).
  - ○ 이하 빌더 서버의 이름은 애플리케이션 단계의 이름을 따른다. 개발 단계의 빌더 서버는 개발 빌더 서버, 테스트 단계와 운영 단계의 빌더 서버는 각각 테스트 빌더 서버와 운영 빌더 서버라고 한다.
- ❒ 모든 룰 서버는 자신과 동일한 단계에 있는 룰 빌더 서버에 모니터 연결을 한다. 개발 단계의 룰 서버들은 개발 빌더 서버, 테스트 단계의 룰 서버들은 테스트 빌더 서버, 운영 단계의 룰 서버는 운영 빌더 서버에 각각 모니터 연결을 한다.

이 구성에서는 룰 서버들이 단계를 넘어가는 모니터 연결을 맺는 문제가 발생하지 않는다. 또한, 설정 클러스터링은 단계 내의 설치 노드 사이에서만 이루어지므로 문제가 없다. 룰 빌더 서버가 이관을 위

해서 연속된 두 단계의 DB들을 접근해야 한다는 점이 제약 사항이지만, 자기보다 하위 단계의 DB를 접근하므로 제약이 덜 할 것이다.

그러나, 이 구성에서도 모니터와 관련된 문제가 있을 수 있다. 개발 빌더 서버와 테스트 빌더 서버가 개발계 룰을 수정할 수 있다. 개발 단계의 룰 서버들은 개발 빌더 서버에 모니터 연결을 맺고 있으므로 테스트 빌더 서버를 이용하여 룰을 저장하면 이 서버들에 변경 사항이 통지되지 않는다. 모니터 문제가 발생하지 않으려면 상위 단계에서 운영되는 룰 빌더 서버가 하위 단계의 룰을 변경하지 못하게 해야 할 필요가 있다.

빌더 서버가 연속된 두 개 이상의 단위 룰 시스템을 관리할 때, 가장 앞 단계 시스템의 룰을 수정할 수 없도록 설정하는 것을 **이관 전용 모드(Transition-Only Mode)**라고 한다. 이관 전용 모드가 설정된 빌더 서버는 그 빌더 서버가 관리하는 단위 룰 시스템 중 가장 앞 단계의 룰 조회는 가능하나 편집은 하지 못한다. 반면 이관 전용 모드가 설정되지 않은 룰 빌더 서버는 가장 앞 단계에 대해 조회 및 편집이 가능하다. 이관 전용 모드의 설정 여부에 관계없이 두 번째 이후의 단위 룰 시스템에 대해서는 편집은 불가능하고 이관은 가능하다.

이관 전용 모드는 서비스 마법사를 이용하여 설정한다. 이관 전용 모드를 이용하여 Figure 3-8-5의 문제점을 해결한 것이 Figure 3-8-6이다.

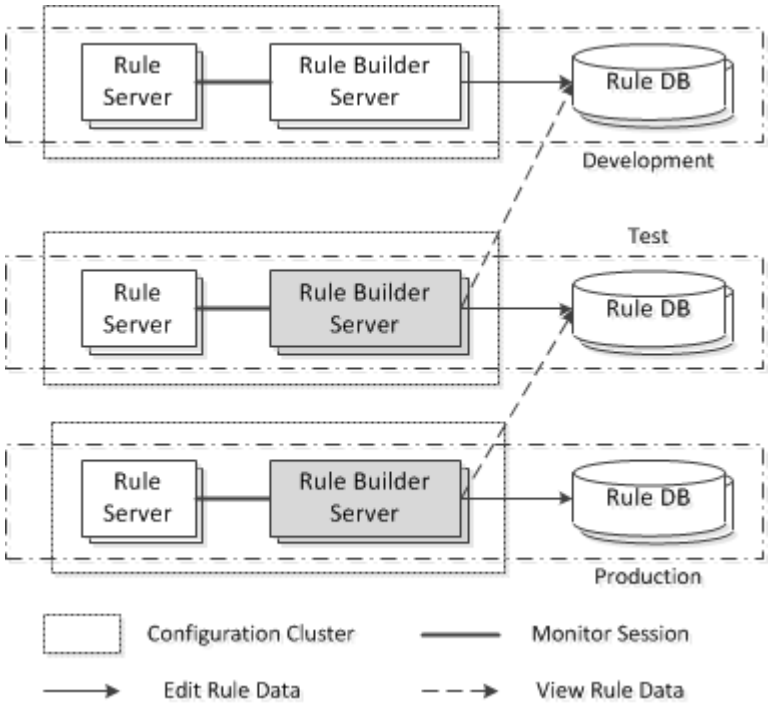

Figure 3 - 8 - 6 Multi-Step Rule System

그림에서 회색으로 표시된 테스트 빌더 서버와 운영 빌더 서버가 이관 전용 모드로 설정되어 있다. 이 빌더 서버들은 점선 화살표로 표시된 바와 같이 앞 단계에 위치한 원본 시스템에 대해 조회만 가능하다. 모든 빌더 서버들은 편집이든 이관이든 자신이 속한 단계의 룰 데이터만을 수정하며 그 룰 데이터를 조회하는 모든 룰 서버들은 이 빌더 서버에 모니터 연결을 하기 때문에 모니터 문제가 발생하지 않는다.

빌더 서버가 앞 단계의 룰 데이터를 변경하지 않는다는 의미가 앞 단계의 룰 DB의 내용을 전혀 변경하지 않는다는 의미는 아니다. 룰 DB에는 룰 요소들뿐만 아니라 히스토리와 이관 관련 플래그도 관리가 된다. 룰 빌더 서버는 이관을 할 때 대상 시스템의 룰 데이터를 변경할 뿐만 아니라 원본 시스템의 히스토리 데이터 및 플래그도 변경한다. 따라서, 두 설정 클러스터에 공유되고 있는 단위 룰 시스템의 데이터가 두 개 이상의 룰 DB에 저장되어야 한다면, 룰 DB들은 두 클러스터에 동일하게 등록되어야 한다. Figure 3-8-6의 예로 들면, 개발 클러스터와 테스트 클러스터에 단위 룰 시스템 *개발계*가 등록되어 있다. 개발 클러스터에 개발계의 룰 DB로 RULEDB1, RULEDB2가 등록이 되어 있다면 테스트계에 등록된 개발계에도 RULEDB1과 RULEDB2가 모두 등록이 되어야 한다. 단계별로 클러스터가 분리된 구성에서 룰 시스템 관리자는 이 점에 각별히 유의해야 한다.

다단계 룰 시스템을 구성할 때는 다음의 규칙을 준수하여야 한다.

- ❒ 설정 클러스터는 단계 단위로 묶는다. 설정 동기화를 위한 네트워크 연결이 가능하다면 두 단계 이상을 하나의 설정 클러스터에 묶을 수 있다.(규칙1)
- ❒ 인접한 설정 클러스터는 하나의 단위 룰 시스템을 공유해야 한다.(규칙2)
- ❒ 각 클러스터에는 적어도 하나의 설치 노드에서 빌더 서버를 활성화 해야 한다.(규칙3)
- ❒ 빌더 서버의 이관 전용 모드는 가장 앞 단계 클러스터에서는 비활성화, 이외의 단계에서는 활성화 한다.(규칙4)

만약 네트워킹 정책이 개발 단계와 테스트 단계의 설정 동기화를 허용한다면, 위 규정을 적용하여 다음과 같은 구성을 할 수도 있다.

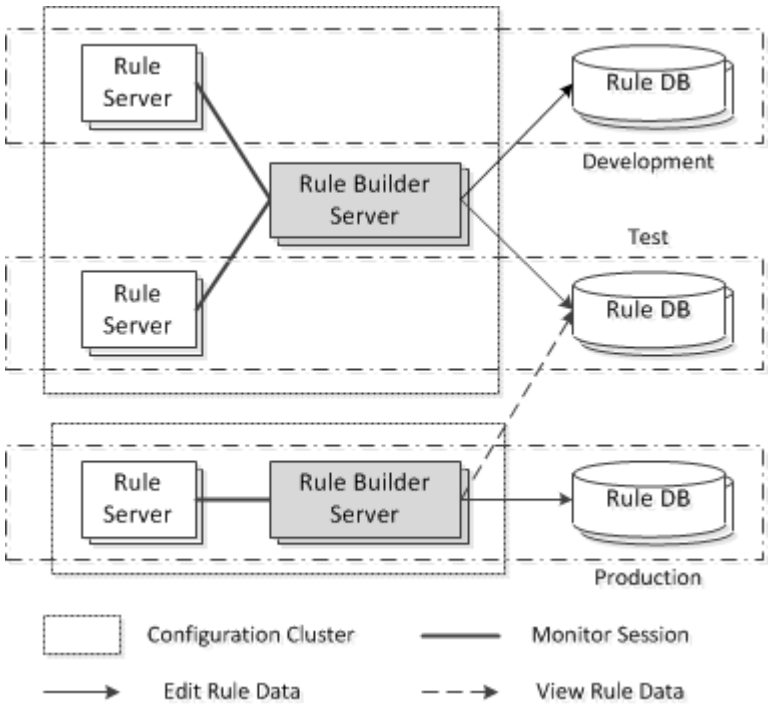

Figure 3 - 8 - 7 Another Multi-Step Rule System

개발 서버들과 테스트 서버들의 설치 노드들이 하나의 설정 클러스터로 묶였으며 운영 서버들의 설치 노드들이 하나의 설정 클러스터로 묶였다(규칙1). 단위 룰 시스템 *테스트*는 개발-테스트 설정 클러스터

와 운영 설정 클러스터에 모두 등록되었다(규칙2). 개발-테스트 클러스터와 운영 클러스터에는 룰 빌더 서버가 활성화되었다(규칙3). 개발-테스트 클러스터의 빌더 서버는 개발 서버 또는 테스트 서버에 활성화될 수 있으며 양쪽 모두에 활성화되어도 관계없다. 개발-테스트 클러스터는 가장 앞 단계의 설정 클러스터이므로 빌더 서버의 이관 전용 모드가 비활성화 되어 있으며, 운영 클러스터에는 이관 전용 모드가 활성화되어 있다(규칙4).

3.8.4 TCP 커넥터 서비스

3.8.4.1 서비스 활성화

TCP 커넥터 서비스(TCP Connector Service)는 TCP/IP와 InnoRules 자체의 프로토콜을 사용하는 원격 룰 서비스이다. TCP 커넥터 서비스는 서비스 마법사를 이용하여 설치 노드별로 활성화할 수 있다. TCP 커넥터 서비스를 활성화하면 InnoRules 서버에 TCP 커넥터 서버가 실행되어 서비스를 제공한다.

TCP 커넥터 서버는 독립적인 주소와 포트를 사용한다. TCP 커넥터 서버를 활성화할 때 주소와 포트를 지정할 수 있다. 이 주소는 서비스를 호출할 룰 애플리케이션이 접근 가능한 네트워크 주소이어야 한다.

TCP 커넥터 서비스는 단위 룰 시스템에 의존적이다. 활성화하기 위해서는 이 서비스가 어떤 단위 시스템의 룰을 서비스할 것인가를 먼저 선택해야 한다. 여러 개의 단위 룰 시스템이 등록되어 있는 경우 단위 룰 시스템별로 서비스를 활성화하며 주소와 포트도 각각 부여해야 한다.

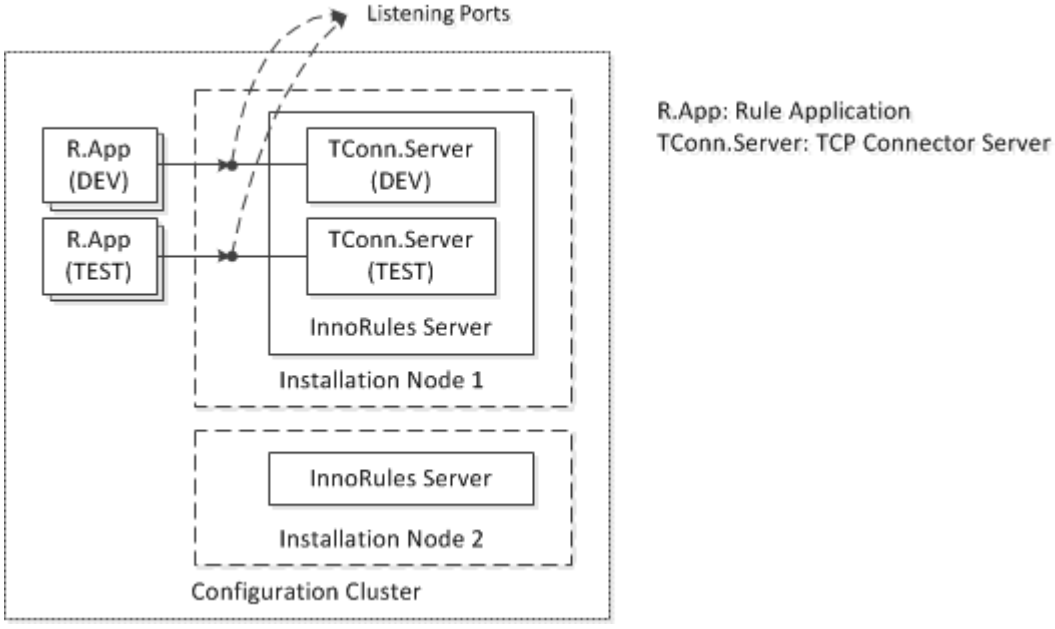

Figure 3 - 8 - 8 TCP Connector Service

Figure 3-8-8은 두 개의 설치 노드로 이루어진 설정 클러스터를 보여준다. 설치 노드들의 이름은 *설치 노드 1*과 *설치 노드 2*이다. 각각의 설치 노드에는 InnoRules 서버가 실행되고 있다. TCP 커넥터 서버는

설치 노드 1에만 활성화되었고, 따라서 이 노드의 InnoRules 서버에 TCP 커넥터 서버가 실행되고 있다. TCP 커넥터 서버는 단위 룰 시스템 *DEV*와 *TEST*에 대해 각각 활성화되었다. 따라서, 두 개의 서버가 각각의 주소와 포트를 점유하고 실행되고 있다. 개발 단계의 애플리케이션들은 DEV의 룰 애플리케이션 설정을 사용하므로 3.6에서 기술한대로 DEV TCP 커넥터 서버에 접속하여 룰 서비스를 요청하게 된다. 같은 방식으로 테스트 단계의 애플리케이션은 TEST TCP 커넥터 서버에 룰 서비스를 요청하게 된다.

3.8.4.2 TCP 커넥터 서버 그룹

다양한 업무 또는 목적의 룰 애플리케이션들이 TCP 커넥터 서비스를 호출할 것이다. 업무별로 또는 목적별로 서로 다른 TCP 커넥터 서버들을 사용하게 하여 자원 사용을 분리할 필요가 있다. 동일한 목적을 갖는 한 개 이상의 TCP 커넥터 서버를 묶은 것을 **TCP 커넥터 서버 그룹(TCP Connector Server Group)**이라고 한다.
서비스 마법사를 이용하여 TCP 커넥터 서버 그룹과 TCP 커넥터 서버를 등록을 할 수 있다. TCP 커넥터 서버를 등록할 때 그룹을 지정해야 한다. 기본적으로 *default* 그룹이 만들어져 있다. 등록된 TCP 커넥터 서버 그룹은 삭제 가능하지만 *default* 그룹은 삭제할 수 없다.
하나의 그룹에 여러 TCP 커넥터 서버를 등록할 수 있다. 이 서버들은 그룹 내에서의 순번이 부여되어 있다. 이 순번은 마법사를 이용하여 변경이 가능하다.

룰 애플리케이션의 서비스 요청을 어떤 서버들이 처리할지를 지정하기 위해 룰 애플리케이션 설정에 TCP 커넥터 서버 그룹을 지정한다. 룰 애플리케이션 설정에 서버 그룹을 설정하는 것은 3.6.7을 참조하라. 애플리케이션이 그룹 내의 어떤 서버에 서비스를 요청할지, 또는 어떤 순서로 서버들에 요청할지를 지정하기 위해 다음의 고가용 방식을 지정할 수 있다.

- ❒ 장애극복(Fail Over)
- ❒ 부하분산(Load Balance)

장애극복(Fail Over) 방식은 서버들에 그룹 내의 우선 순위를 정해 놓고 애플리케이션들이 가용한 최우선 순위의 룰 서버로만 룰 서비스 요청을 보내도록 하는 방식이다. 우선 순위는 순번에 의해 부여된다. 더 낮은 순번의 서버가 우선 순위가 높다. 특정 시점에 애플리케이션은 단 하나의 서버로만 룰 서비스를 요청하게 되며 이 서버를 이 애플리케이션의 **액티브 서버(Active Server)**라고 한다. 액티브 서버가 중단되거나 장애가 발생하여 사용할 수 없는 상태가 되면 그 다음 우선 순위의 룰 서버가 액티브 서버가 된다.

부하분산(Load Balance) 방식에서는 애플리케이션이 모든 서버들에 골고루 서비스 요청을 보내도록 하는 방식이다. 애플리케이션은 서버의 순번 순으로 서비스 요청을 보낸다. 설명하자면, 최초의 요청은 1번 서버에, 두 번째 요청은 2번 서버에 보내는 식이다. 마지막 순번 서버에 요청을 보냈다면 그 다음의 요청은 다시 1번 서버로 보낸다. 중단되거나 장애가 발생한 서버는 이 순서에서 제외된다.

이중화(Replication) 또는 고가용(High Availability) 구성을 위해 장애극복 또는 부하분산 방식을 사용할 수 있다.

많은 이중화 구성에서 룰 애플리케이션과 TCP 커넥터 서버는 쌍으로 구성된다. Figure 3-8-9은 이러한

이중화 구성의 예를 보여준다. 애플리케이션과 TCP 커넥터 서버는 동일한 호스트에 구성이 되어 있다. 애플리케이션이 사용해야 할 룰 서버의 우선 순위는 절대적인 것이 아니며, 애플리케이션과 쌍을 이루는 룰 서버가 우선이고 그 서버를 사용할 수 없을 때 다른 호스트의 룰 서버를 사용하도록 되어 있다.

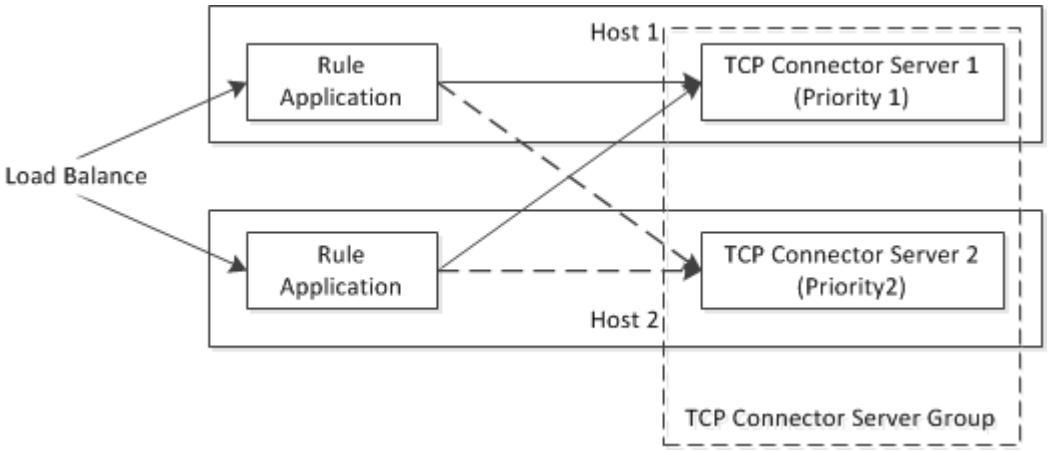

Figure 3 - 8 - 9 An Example of a Replication

이 예는 이중화로 동작하는 두 개의 호스트를 보여주고 있다. 각 호스트에는 룰 애플리케이션과 TCP 커넥터 서버가 동작하고 있다. TCP 커넥터 서버들은 TCP 커넥터 서버 그룹으로 묶여 있고 애플리케이션들은 이 그룹에 룰 서비스 요청을 한다. 외부로부터의 요청은 L4 스위치 등을 이용하여 각 애플리케이션에 분산되므로 두 애플리케이션의 룰 서비스 호출 빈도는 비슷할 것이다. 따라서, TCP 커넥터 서버로의 룰 서비스 요청을 또다시 분산할 필요는 없으므로 부하분산 방식을 지정할 필요는 없다. 그러나, 하나의 TCP 커넥터 서버에 장애가 발생하더라도 룰은 정상적으로 서비스되어야 하므로 TCP 커넥터 서버 그룹을 장애극복 방식으로 설정할 필요는 있다.

TCP 커넥터 서버에는 순번이 부여되어 있고, 그 순번에 따라 TCP 커넥터 서버의 우선 순위는 TCP 커넥터 서버 1, TCP 커넥터 서버2 순이다. 따라서, 서버 1이 정상 동작하고 있는 경우 어떤 룰 애플리케이션이든 서비스 요청을 서버 1로 보낸다. 룰 애플리케이션과 TCP 커넥터 서버 사이의 실선 화살표가 이를 나타낸다. 만약, 서버 1에 장애가 발생하여 서버 2가 액티브 서버가 되면 모든 룰 서비스 요청은 서버 2로 보내진다. 룰 애플리케이션과 TCP 커넥터 서버 사이의 점선 화살표가 이를 나타낸다.

이렇게 구성되는 경우 모든 룰 서비스 부하가 한 쪽 호스트로 집중되는 문제가 발생한다. 이 문제는 룰 애플리케이션이 TCP 커넥터 서버의 우선 순위에 관계없이 자신의 호스트에서 실행되는 서버에 우선적으로 서비스를 요청하게 함으로써 해결할 수 있다. Figure 3-8-10가 이 구성을 보여준다.

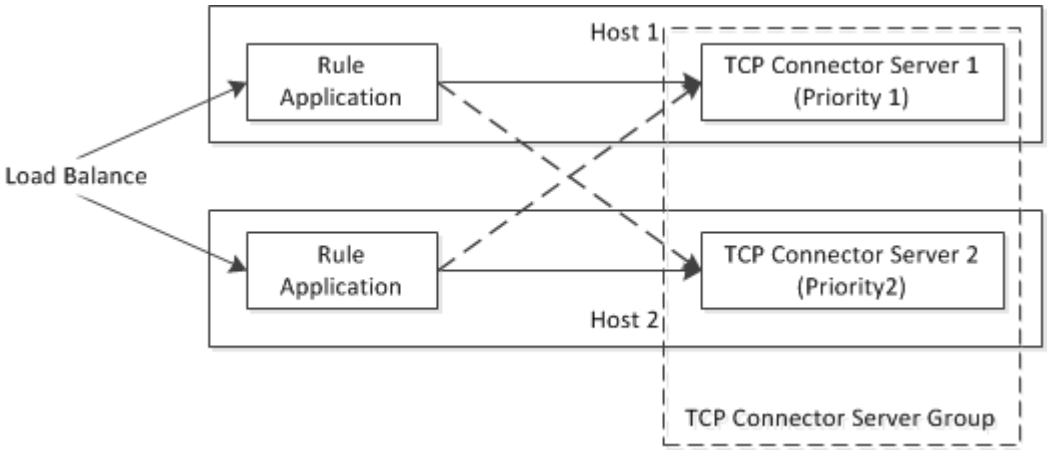

Figure 3 - 8 - 10 An Example of a Replication with Local First option

Figure 3-8-10에서 룰 애플리케이션들은 우선적으로 자신이 실행되고 있는 호스트의 TCP 커넥터 서버로 서비스를 요청한다. 호스트 1의 애플리케이션은 서버 1로, 호스트 2의 애플리케이션은 서버 2로 서비스를 요청한다. 룰 애플리케이션으로부터 서버로의 실선 화살표가 이를 나타낸다. 외부로부터 룰 애플리케이션으로의 부하가 분산되고, 애플리케이션들은 각자의 TCP 커넥터 서버로 서비스를 요청하기 때문에 결과적으로 TCP 커넥터 서버로의 부하도 고르게 분산된다.
서버 1이 중단된다면 호스트 1의 애플리케이션은 서버 2로 서비스를 요청한다. 반면 서버 2가 중단된다면 호스트 2의 애플리케이션이 서버 1로 서비스를 요청한다.

이와 같이 애플리케이션 TCP 커넥터 서버 그룹에 지정된 우선 순위에 관계없이 동일한 호스트의 서버를 우선적으로 사용하도록 하는 옵션이 **로컬 우선(Local First)** 옵션이다. 이 옵션은 TCP 커넥터 서버 그룹에 지정된다. 이 옵션을 활성화한 경우 지정된 고가용 방식에 따라 서버들의 우선 순위가 재조정된다.

장애극복 방식에서는

- ❒ TCP 커넥터 서버 그룹에 룰 애플리케이션과 동일한 호스트에서 동작하는 서버들이 있다면, 그 서버들 중 순번이 가장 높은 서버가 최우선 순위가 된다. 이후의 우선 순위는 최우선 순위의 서버를 제외한 나머지의 순번에 따른다.
- ❒ 선택된 TCP 커넥터 서버 그룹에 룰 애플리케이션과 동일한 호스트에서 동작하는 서버가 없다면 그룹에 등록된 순번에 따른다.

부하분산 방식에서는

- ❒ TCP 커넥터 서버 그룹에 룰 애플리케이션과 동일한 호스트에서 동작하는 서버들이 있다면, 그 서버들 중 순번이 가장 앞선 서버를 선택하여 첫 번째 순번으로 하고 그 서버 이후 순번의 서버들이 뒤를 따른다. 선택된 서버가 그룹에서 첫 번째 순번이 아닌 경우, 그룹에서 첫 번째부터 선택된 서버의 앞 순번까지의 서버들이 그 뒤를 따른다.
- ❒ 선택된 TCP 커넥터 서버 그룹에 룰 애플리케이션과 동일한 호스트에서 동작하는 서버가 없다면 그룹에 등록된 순번에 따른다.

3.8.5 룰 REST 서비스

룰 REST 서비스는 HTTP 기반의 REST를 이용하여 룰 서비스를 호출하도록 해 주는 서비스이다.
룰 REST 서비스를 이용하여 룰 서비스를 호출하기 위해서는 다음이 필요하다.

- ❒ 룰 REST 서비스 활성화
  InnoRules 서버에 룰 REST 서비스를 활성화해야 한다. 두 개 이상의 설치 노드에 룰 REST 서비스를 활성화할 수도 있다. 룰 REST 서비스 활성화에 대해서는 2.4.7 룰 REST 서비스를을 참조하라.
  활성화된 단위 룰 시스템에 등록된 모든 룰은 룰 REST 서비스의 대상이 된다.

3.8.5.1 룰 REST 서비스 호출 규약

HTTP Client 기반의 모든 애플리케이션은 룰 REST 서비스를 호출할 수 있다. 단 다음을 준수해야 한다.

- ❒ HTTP Header
  HTTP 메시지의 Header에 다음의 값을 전달하여야 한다.
  - ○ content-type
    application/json (필수)
- ❒ HTTP METHOD
  룰 REST 서비스는 POST 방식만 지원한다.
- ❒ 전송 데이터 형식
  전송 되는 HTTP 메시지의 형식은 JSON 형태의 Text 만 지원한다.

3.8.5.2 룰 REST 서비스 URL

서비스를 요청을 하기 위해서는 서비스에 대한 URL이 지정이 되어야 한다. URL은 다음과 같이 구성된다.

http://[IP]:[PORT]/innorules/services/rest/[ALIAS]/[RULECODE][/DATE]

URL을 구성하는 요소의 의미는 다음과 같다.

- ❒ [IP]
  InnoRules 서버의 주소
- ❒ [PORT]
  InnoRules 서버의 HTTP 서비스 포트
- ❒ innorules
  InnoRules 서비스 웹 애플리케이션의 컨텍스트 경로
- ❒ services/rest
  InnoRules 서비스 중 룰 REST 서비스임을 나타내는 부분
- ❒ [ALIAS]
  어떤 단위 룰 시스템의 룰을 호출할 것인가를 지정한다. 각 단위 룰 시스템에는 식별을 위한 별칭이 부여될 수도 있다. 이를 룰 웹 서비스 시스템 별칭이라고 한다. 별칭이 부여된 경우 ALIAS에 별칭을 지정한다.

- ❒ [RULECODE]
  서비스 대상 룰 코드, 룰 호출 코드 체계(JSON 입력의 codeType)에 맞게 지정한다.
- ❒ [DATE]
  룰 실행일자, 생략 가능하며 입력되지 않은 경우 현재 일자가 사용된다.

3.8.5.3 룰 REST 서비스 시스템 별칭

하나의 InnoRules 서버가 여러 단위 룰 시스템의 룰들을 서비스할 수 있기 때문에 URL에는 단위 시스템을 구분할 수 있는 식별자가 부여될 필요가 있다. URL은 외부에 제공이 된다는 특성이 있기 때문에 단위 룰 시스템의 ID나 이름을 사용할 경우 그 정보가 외부에 유출될 수 있다. 단위 룰 시스템의 정보를 URL에서 숨기기 위해서 별칭을 부여하는데 이를 룰 웹 서비스 시스템 별칭이라고 한다. 이 별칭은 3.8.6.1에서 설명한 바와 같이 룰 REST 서비스 URL를 구성하는 한 요소이다.

3.8.5.4 고가용 설정

여러 InnoRules 서버에 룰 REST 서비스를 활성화하고 부하분산(load balance)이나 장애극복(fail over)으로 고가용 설정을 해야 할 경우가 있다. InnoRules에서는 고가용을 위해 별도의 기능을 지원하지 않으므로, L4 스위치 등 일반적인 웹 애플리케이션 아키텍처에서의 고가용 설정과 동일한 방법을 이용하면 된다.

설정 클러스터 내에서 가용한 REST 서비스들과 그들의 URL에 대해 조회하기 위해서는 2.4.7.1의 룰 REST 서비스 URL 조회 메뉴를 사용하면 된다.

4. 에러 코드와 조치 방법

룰 시스템에 장애*6가 발생하면 룰 시스템은 자바 Exception을 던지거나 로그에 문제점을 기록한다. Exception이나 로그에는 장애의 현상, 원인을 파악할 수 있는 에러 코드가 부여된다. 이 에러 코드는 장애를 처리하는 데에 유용하게 사용될 수 있다.

이 장에서는 에러 코드별로 그 원인과 대처 방법을 설명한다. 본 문서에 기술되지 않은 에러 코드에 대해서는 기술지원을 요청하도록 한다.

*6. 장애의 유형은 다양하다. 룰 시스템 자체의 장애일 수도 있고 서비스를 호출하는 애플리케이션에서 잘못된 인자를 넘겼을 수도 있다. 장애의 정도도 복구 불가능한 것에서부터 복구 가능 또는 무시할 수 있는 것까지 다양하다. 장애가 영향을 미치는 범위도 룰 시스템 전체 또는 특정 룰 런타임이 영향을 받거나 특정 서비스 호출만 문제가 될 수도 있다.

4.1 IRE-10XXX

룰 서비스를 호출하는 도중 발생할 수 있는 에러들이다. 에러는 대체로 로그에도 기록이 되고 룰 서비스를 호출한 애플리케이션에도 전달이 된다. 룰 서비스를 호출하다가 에러가 발생하면 애플리케이션은 적절한 예외 처리를 해야 한다.
에러의 원인을 이해하기 위해서는 룰 서비스 API 또는 룰 작성 방법에 대한 이해가 다소 필요할 수도 있다. 이에 대해서는 Rule Application Programming Guide와 InnoRules User's Guide of Rule Builder를 각각 참조하라.

4.1.1 IRE-10000

- ❒ 에러 메시지
  항목이 입력되지 않았습니다.
- ❒ 원인
  룰 실행에 필요한 항목이 애플리케이션으로부터 입력되지 않았음. 그 항목의 ID와 이름, 별칭이 있다면 별칭이 메시지에 포함됨. 다음의 경우 항목이 입력되지 않은 것으로 판정함
  - O I/O 객체 방식으로 룰을 호출하는 경우 RuleReq 객체에 항목의 값이 포함되어 있지 않음
  - O I/O 어댑터 방식으로 룰을 호출하는 경우 ItemProvider의 getAsXXX 메소드가 null을 반환함
- ❒ 조치 방법
  룰 서비스를 호출할 때 누락된 항목의 값을 입력하도록 애플리케이션 담당자는 애플리케이션을 수정한다.

4.1.2 IRE-10001

- ❒ 에러 메시지
  정의되지 않은 룰입니다.
- ❒ 원인
  애플리케이션이 룰을 호출할 때 룰의 코드로 유효하지 않은 ID 또는 이름 또는 별칭을 전달함. "유효하지 않은"의 의미는 그 ID를 갖는 룰(또는 이름 또는 별칭을 갖는 룰)이 정의되어 있지 않다는 의미이다. 문제가 되는 ID(또는 이름 또는 별칭)가 메시지에 포함된다.
- ❒ 조치 방법
  애플리케이션 개발자는 호출하고자 하는 룰의 코드를 룰 개발자로부터 확인하고 필요한 경우 애플리케이션을 수정한다.

4.1.3 IRE-10002

- ❒ 에러 메시지
  정의되지 않은 항목입니다.
- ❒ 원인
  애플리케이션이 룰을 호출할 때 항목의 코드로 유효하지 않은 ID 또는 이름 또는 별칭을 전달함. "유효하지 않은"의 의미는 그 ID를 갖는 항목(또는 이름 또는 별칭을 갖는 항목)이 정의되어 있지 않다는 의미이다. 문제가 되는 ID(또는 이름 또는 별칭)가 메시지에 포함된다.
- ❒ 조치 방법
  애플리케이션 개발자는 전달하고자 하는 항목의 코드를 룰 개발자로부터 확인하고 필요한 경우 애플리케이션을 수정한다.

4.1.4 IRE-10003

- ❒ 에러 메시지
  입력 항목 값이 정의된 데이터 형식과 다릅니다.
- ❒ 원인
  애플리케이션이 I/O 객체 방식으로 룰을 호출할 때, I/O 객체에 설정한 값의 데이터 형식이 룰 시스템에 등록되어 있는 데이터 형식과 다른 경우 발생한다. 예를 들어, 룰 시스템에 항목이 문자형으로 등록이 되어 있는데 애플리케이션은 그 항목의 값으로 숫자값을 설정한 경우이다.
- ❒ 조치 방법
  애플리케이션 개발자는 문제가 되는 항목의 데이터 형식을 룰 개발자로부터 확인하고 필요한 경우 애플리케이션을 수정한다.

4.1.5 IRE-10004

- ❒ 에러 메시지
  올바르지 않은 항목 값.
- ❒ 원인

룰 애플리케이션이 I/O 어댑터 방식으로 룰을 호출할 때 발생할 수 있다. ItemProvider가 항목의 값을 배열 형식으로 반환할 때에는

- 숫자형 항목은 getAsNumber에서 double[]을
- 문자형 항목은 getAsString에서 String[]을

반환해야 하며, 배열의 길이는 1 이상이어야 한다. 특히 String[]의 각 요소는 null이어서는 아니된다. 만약 배열의 길이가 0이거나 String[]의 요소 중 null이 포함되어 있는 경우 이 에러가 발생할 수 있다. 메시지에 문제가 되는 항목의 ID가 포함된다.

❒ 조치 방법

애플리케이션 개발자는 ItemProvider가 지목된 항목의 값으로 길이가 0인 배열을 반환하거나 null을 포함한 String[]을 반환하는지 확인한 후 그렇다면 애플리케이션을 수정한다.

룰 서버는 룰 서비스 실행 도중 ArrayIndexOutOfBoundsException 또는 NullPointerException이 발생하면 입력된 항목의 유효성을 확인하고 위에서 기술한 문제가 있다면 이 에러를 발생시킨다. 올바르지 않은 항목 값이 ArrayOutOfBoundsException 또는 NullPointerException을 발생시켰다고 단정할 수는 없지만 상당한 인과관계가 있을 수 있으므로 항목 값의 유효성을 우선적으로 확인하도록 한다. 만약, 항목의 값이 유효한데도 불구하고 이 에러가 지속적으로 발생한다면 다른 원인이 있을 수 있으므로 기술지원을 요청하도록 한다.

4.1.6 IRE-10006

❒ 에러 메시지

내장 함수의 인자가 올바르지 않습니다.

❒ 원인

룰이 실행되는 도중 내장 함수에 허용되지 않는 인자 값이 전달되었을 때 발생한다. 내장 함수의 인자의 데이터 형식은 룰을 저장할 때 체크가 되지만, 값의 유효성, 배열의 길이 등은 런타임에 체크된다.

에러 메시지에 함수의 이름과 인자 값이 올바르지 않은 사유가 포함된다.

❒ 조치 방법

룰 개발자는 허용되지 않는 인자 값이 전달된 이유를 파악한다. 그 원인이 룰 작성에 있는 경우 룰을 수정하고 애플리케이션으로부터의 입력 값에 있는 경우 애플리케이션 개발자가 애플리케이션을 수정할 수 있도록 한다.

4.1.7 IRE-10007

❒ 에러 메시지

사용자 정의 함수에서 에러를 반환하였습니다.

❒ 원인

룰이 실행되는 도중 사용자 정의 함수에서 예외를 던졌을 때 발생한다. 함수의 메시지에 에러를 반환한 사용자 정의 함수의 이름이 포함된다.

❒ 조치 방법

사용자 정의 함수 개발자에게 에러 메시지를 전달하고 에러의 사유를 확인한 후 적절한 조치를 한다.

4.1.8 IRE-10008

❒ 에러 메시지

사용자 정의 에러가 발생하였습니다.

❒ 원인

룰 개발자는 특정한 조건을 만족하는 경우 ERROR 함수가 호출되도록 하여 더 이상 룰이 실행되지 않도록 할 수 있다. ERROR 함수에는 메시지와 사용자 정의 에러 코드를 전달할 수 있다. 에러 메시지에는 이 메시지와 사용자 정의 에러 코드(부여하였다면)가 포함된다.

룰 서버는 ERROR 함수를 실행하게 되면 이 에러를 발생하고 룰 실행을 중지한다.

❒ 조치 방법

이 에러는 업무적인 수준의 에러이기 때문에 룰 개발자와 애플리케이션 개발자가 협의하여 처리해야 한다. 만약 룰 개발자가 절대 발생해서는 아니 되는 조건 하에서 이 에러가 발생되도록 하였다면, 그 조건에 부합하게 원인을 찾아 룰 또는 애플리케이션 또는 데이터를 수정해야 한다. 그렇지 않고, 룰 개발자가 예외 처리를 위한 방법으로 이 에러를 발생시킨 것이라면 애플리케이션 개발자는 이 에러 코드에 대한 적절한 처리를 하여 애플리케이션이 계속 진행되도록 해야 할 것이다.

4.1.9 IRE-10009

❒ 에러 메시지

룰 호출 기준일자에 적용할 수 있는 버전이 없습니다.

❒ 원인

룰 애플리케이션은 룰을 호출할 때 룰 실행의 기준일자를 전달할 수 있다. 전달하지 않은 경우 룰 서비스 호출일자가 그 기준일자가 된다. 룰 서비스를 실행하는 도중 메인 룰 또는 서브 룰에 기준일자에 적용 가능한 버전이 없는 경우 이 에러가 발생한다. 이 에러의 의미를 업무적으로 말하자면 "주어진 날짜에 적용할 수 있는 규칙이 없다"이다.

에러 메시지에 문제의 룰과 애플리케이션에서 전달된 기준일자가 포함된다.

❒ 조치 방법

메인 룰에서 에러가 발생한 경우 메인 룰의 개시 시점 이전의 날짜로는 룰 서비스를 호출하지 않도록 애플리케이션을 수정한다. 또는 업무적으로 가능한 경우 메인 룰의 개시 시점을 변경한다.

서브 룰에서 에러가 발생한 경우는 하위 룰의 개시 시점이 실행된 상위 룰의 버전의 개시 시점보다 이후가 되지 않도록 룰을 재구성 한다. 예를 들어, 다음과 같은 경우를 가정하자.

- 룰 실행 기준일자는 2015-05-01이다.
- 상위 룰에는 개시 시점이 2015-01-01인 버전만이 있다.
- 하위 룰은 개시 시점이 2015-06-01인 버전만이 있다.

이 경우 상위 룰은 기준일자에 적용할 수 있는 버전이 있기 때문에 정상적으로 실행된다. 그러나, 하위 룰은 적용할 수 있는 버전이 없기 때문에 하위 룰이 호출될 때 이 에러가 발생한다.

이를 해결하기 위해서는 상위 룰을 2015-01-01에 개시하는 버전과 2015-06-01에 개시하는 버전으로 나누고 전자에서는 하위 룰을 참조하지 않도록 버전을 분리해야 한다(업무적으로도 이것이 의미가 있을 것이다).

4.1.10 IRE-10011

- ❒ 에러 메시지
  룰 결과를 I/O 어댑터에 설정하는 도중 에러가 발생하였습니다.
- ❒ 원인
  룰 애플리케이션이 I/O 어댑터 방식으로 룰을 호출하는 경우, 룰 서버는 룰이 정상적으로 실행되면 그 결과를 ResultConsumer의 set 메소드를 이용하여 I/O 어댑터에 설정한다. 이 때 set 메소드 안에서 Exception이 던져지면 이 에러가 발생한다. 메시지에 Exception의 메시지가 포함된다.
- ❒ 조치 방법
  ResultConsumer 개발자에게 이 에러 메시지에 포함된 메시지를 전달하고 그 원인에 따라 조치한다.

4.1.11 IRE-10012

- ❒ 에러 메시지
  0으로 나누기 오류
  또는
  ^ 연산의 첫 번째 피연산자는 0일 수 없습니다.
- ❒ 원인
  에러 메시지에 따라 원인이 다르다. "0으로 나누기 오류"는 다음의 경우 발생한다.
  - ○ 나누기 연산에서 제수가 숫자인 경우 그 값이 0일 때
  - ○ 나누기 연산에서 제수가 다중 값인 경우 다중 값에 값이 0인 요소가 있을 때
  - ○ 나머지 연산에서 제수의 값이 0일 때

  "^ 연산의 첫 번째 피연산자는 0일 수 없습니다"는 ^ 연산의 밑수(첫 번째 피연산자)가 0일 때 발생한다.
- ❒ 조치 방법
  피연산자에 0이 오지 않도록 조치한다.

4.1.12 IRE-10013

- ❒ 에러 메시지
  브랜치 룰의 분기 조건의 결과에 일치하는 브랜치 노드가 없습니다.
- ❒ 원인
  브랜치 룰은 분기 조건을 실행하고 그 결과와 일치하는 이름의 브랜치 노드 룰을 실행하는 룰이다. 분기 조건을 실행하였으나 그 이름의 룰이 없는 경우 또는 룰은 있으나 그 룰이 브랜치 노드로 설정되어 있지 않은 경우 이 에러가 발생한다. 단, 디폴트 값이 있는 경우에는 에러가 발생하지 않고 디폴트 값이 실행된다.
  메시지에는 분기 조건의 결과가 포함된다.
- ❒ 조치 방법

룰 개발자는 다음을 순서대로 체크한다.

- 분기 조건이 올바른지 확인한다.
- 분기 조건에 사용된 항목의 값이 유효한 값인가 확인한다.
- 새로운 브랜치 노드를 추가하거나, 기존을 룰을 브랜치 노드로 설정해야 하는지 검토한다.
- 업무적으로 디폴트 값을 설정해야 하는지 확인한다.

4.1.13 IRE-10015

- 에러 메시지
  문법 에러가 있어 룰을 실행할 수 없음.
- 원인
  룰 서비스 도중 실행하려는 메인 룰 또는 그 하위 룰에 문법 에러가 있어 실행을 할 수 없는 경우 발생한다. 메시지에 에러가 있는 룰의 정보가 포함된다.
- 조치 방법
  룰 개발자에게 에러가 발생한 룰의 정보를 전달하고 문법 에러를 고치도록 한다.

  룰 작성이 완료되지 않아 아직 문법 에러가 있는 룰도 저장이 가능하며, 저장할 때에는 문법 에러가 없던 룰에도 연관 관계에 있는 룰의 변경으로 인하여 문법 에러가 발생할 수 있다. 이 경우 룰을 저장하거나 변경하는 룰 작성자에게 경고 메시지가 전달된다. 가능하다면 룰 작성자는 이 경고 메시지를 받는 즉시 문법 에러를 해소하는 것이 좋다. 그 전에 이 룰이 호출된다면 이 에러가 발생하게 된다.

4.1.14 IRE-10016

- 에러 메시지
  항목 값에 null이 포함되었습니다.
- 원인
  사용자 정의 함수에서 룰을 호출할 때 넘겨진 항목이 null인 경우 발생한다.
- 조치 방법
  사용자 정의 함수 개발자는 룰을 호출할 때 항목의 값에 null이 전달되지 않도록 함수의 코드를 수정한다.

  사용자 정의 함수의 execute 메소드에는 com.innoexpert.innorulesj.suite.udf.Engine 객체가 전달되고 이 객체의 executeRule, executeRuleByName, executeRuleAlias 메소드를 이용하여 하위 룰을 호출할 수 있다. 이 때 추가적인 항목들을 전달할 수 있는데 이 항목의 값들은 null이어서는 아니 된다. 사용자 정의 함수에 대한 자세한 사항은 Rule Application Programming Guide를 참조하라.

4.1.15 IRE-10017

- 에러 메시지

정의되지 않은 룰입니다.

- ❒ 원인
  사용자 정의 함수에서 호출하려는 룰이 정의되지 않은 룰인 경우 발생한다. 메시지에 사용자 정의 함수가 요청한 룰의 ID 또는 이름 또는 별칭이 포함된다.
- ❒ 조치 방법
  사용자 정의 함수 개발자는 룰 코드를 확인하고 함수의 코드를 수정한다.

4.1.16 IRE-10018

- ❒ 에러 메시지
  정의되지 않은 항목입니다.
- ❒ 원인
  사용자 정의 함수에서 룰을 호출할 때 전달한 항목이 정의되지 않은 항목인 경우 발생한다. 메시지에 사용자 정의 함수가 전달한 항목의 ID 또는 이름 또는 별칭이 포함된다.
- ❒ 조치 방법
  사용자 정의 함수 개발자는 항목 코드를 확인하고 함수의 코드를 수정한다.

4.1.17 IRE-10019

- ❒ 에러 메시지
  항목의 데이터 형식이 올바르지 않습니다.
- ❒ 원인
  사용자 정의 함수에서 룰을 호출할 때 전달한 항목의 데이터 형식이 룰 시스템에 등록된 데이터 형식과 다른 경우 발생한다. 메시지에 그 항목의 ID 또는 이름 또는 별칭이 포함된다.
- ❒ 조치 방법
  사용자 정의 함수 개발자는 항목의 데이터 형식을 확인하고 함수의 코드를 수정한다.

4.1.18 IRE-10020

- ❒ 에러 메시지
  재귀 룰 호출이 발생하였습니다.
- ❒ 원인
  하나의 룰은 실행 중에 다른 룰을 호출할 수 있다. 이 호출 관계에서 자기가 호출한 룰 또는 그 룰의 하위에서 호출된 룰이 자기 자신을 다시 호출하는 것을 재귀 호출이라고 한다. 이 에러는 재귀 호출이 일어났을 때 발생한다. 메시지에는 재귀 호출된 룰의 ID가 포함된다.
  InnoRules는 룰을 저장할 때 가능한 호출 관계를 확인하고 재귀 호출이 일어날 가능성이 있는 경우 에러를 발생시킨다. 그러나, EXEC 함수 또는 EXECRULE 함수를 이용하는 경우 또는 사용자 정의 함수에서 동적으로 룰을 호출하는 경우 어떤 룰이 호출될지는 런타임에 결정되므로 룰을 저장할 때 재귀호출을 감지할 수가 없다.
- ❒ 조치 방법
  룰 개발자는 EXEC 함수, EXECRULE 함수 또는 사용자 정의 함수가 재귀 호출을 발생할 수 있

는지 확인하고, 그렇다면 재귀 호출이 발생하지 않도록 수정한다. 그러나, 그렇지 않았음에도 불구하고 이 에러가 발생한다면 기술지원을 요청하도록 한다.

4.1.19 IRE-10021

- ❒ 에러 메시지
  정의되지 않은 코드 체계입니다.
- ❒ 원인
  사용자 정의 함수에서 com.innoexpert.innorulesj.suite.udf.Engine.execute 메소드에 전달된 코드 체계가 유효하지 않은 코드 체계인 경우 발생한다. 사용자 정의 함수에서 전달한 코드 체계의 값이 메시지에 포함된다.
- ❒ 조치 방법
  사용자 정의 함수 개발자는 코드 체계의 값을 확인하고 함수의 코드를 수정한다. 지원되는 코드 체계는 Rule Application Programming Guide를 참조하라.

4.1.20 IRE-10022

- ❒ 에러 메시지
  문법에 맞지 않는 표현식입니다.
- ❒ 원인
  사용자 정의 함수에서 com.innoexpert.innorulesj.suite.udf.Engine.execute 메소드로 실행하려고 하는 룰 표현식에 문법 오류가 있는 경우 발생한다. 메시지에 문법 오류 메시지가 포함된다.
- ❒ 조치 방법
  사용자 정의 함수 개발자는 문법 오류 메시지를 확인하고 문법에 맞지 않는 표현식이 전달되지 않도록 수정한다.

4.1.21 IRE-10023

- ❒ 에러 메시지
  항목의 값이 중복 입력되었습니다.
- ❒ 원인
  룰 애플리케이션이 I/O 객체 방식으로 룰을 호출할 때, RuleReq 객체에 전달된 항목들 중에 코드가 중복되는 것들이 있는 경우 발생한다.
- ❒ 조치 방법
  룰 애플리케이션 개발자는 RuleReq 객체에 동일한 코드의 항목이 한 번만 저장되도록 애플리케이션을 수정한다.

4.1.22 IRE-10024

- ❒ 에러 메시지
  아이템 프로바이더가 허용되지 않은 객체 형식을 반환하였습니다.

❒ 원인

룰 애플리케이션이 I/O 어댑터 방식으로 룰을 호출할 때 사용한 ItemProvider가 허용되지 않은 객체 형식을 반환하였을 때 이 에러가 발생한다. 항목의 데이터 형식에 따라 ItemProvider는 getXXX 메소드에서 다음의 자바 객체를 반환할 수 있다.

○ 숫자형 항목

getAsNumber 메소드는 java.lang.Double 또는 double[]을 반환해야 한다.

○ 문자형 항목

getAsString 메소드는 java.lang.String 또는 java.lang.String[]을 반환해야 한다.

에러 메시지에는 실제 반환된 객체의 데이터 형식이 포함된다.

❒ 조치 방법

룰 애플리케이션 개발자는 허용된 형식의 객체만을 반환하도록 ItemProvider를 수정한다.

4.1.23 IRE-10025

❒ 에러 메시지

항목 값 질의 중 아이템 프로바이더가 예외를 던졌습니다.

❒ 원인

룰 애플리케이션이 I/O 어댑터 방식으로 룰을 호출할 때 발생할 수 있다. 룰 서버가 룰 서비스를 실행하는 도중 항목의 값을 질의하기 위해 ItemProvider의 getXXX 메소드를 호출하였으나 이 메소드에서 예외가 던져졌을 때 이 에러가 발생한다. 예외의 메시지가 에러 메시지에 포함된다.

❒ 조치 방법

ItemProvider의 개발자에게 예외 메시지를 전달하고 조치 방법을 문의한다.

4.1.24 IRE-10026

❒ 에러 메시지

EXEC 함수 실행 결과를 요구되는 데이터 형식으로 변환할 수 없습니다.

❒ 원인

EXEC, EXECRULE 함수는 테이블 룰의 조건 셀, 결과 셀 또는 DB 룰, 브랜치 룰 등의 기본값의 셀에서 사용할 수 있다. 이 셀들은 특정한 데이터 형식을 가져야 한다. 예를 들어, 테이블 룰의 조건 셀은 논리값을 가져야 하며, 테이블 룰의 결과 셀 또는 기본값 셀은 룰의 데이터 형식과 호환되는 값을 가져야 한다. EXEC, EXECRULE 함수의 결과가 이 조건을 만족하지 않을 때 이 에러가 발생한다. 셀이 요구하는 데이터 형식과 함수 결과의 데이터 형식이 메시지에 출력된다.

❒ 조치 방법

룰 개발자는 셀에서 요구되는 데이터 형식이 반환되도록 EXEC 또는 EXECRULE 함수의 인자를 수정한다.

4.1.25 IRE-10027

- ❒ 에러 메시지
  사용자 정의 함수에서 실행 오류가 발생하였습니다.
- ❒ 원인
  룰 서버가 사용자 정의 함수의 실행 코드를 호출하였고, 이 코드에서 자바 예외를 던졌을 때 이 에러가 발생한다. 사용자 정의 함수의 에러 메시지가 이 메시지에 포함된다.
- ❒ 조치 방법
  사용자 정의 함수 개발자에게 에러 메시지를 전달하고 발생한 원인을 문의하여 적절한 조치를 취한다.

4.1.26 IRE-10028

- ❒ 에러 메시지
  사용자 정의 함수의 실행 결과가 요구되는 데이터 형식이 아닙니다.
- ❒ 원인
  사용자 정의 함수는 check 메소드와 execute 메소드를 갖고 있다. check는 사용자 정의 함수를 포함한 룰 표현식이 저장될 때 함수의 인자의 유효성을 검증하고 함수의 리턴 형식을 반환하는 함수이다. execute는 인자의 값을 받아 결과를 반환하는 실행 메소드이다. execute의 결과는 check서 반환했던 리턴 형식이어야 한다. 만약 그렇지 않은 경우 이 에러가 발생한다.
- ❒ 조치 방법
  사용자 정의 함수 개발자는 execute 메소드의 결과가 check에서 반환했던 형식과 일치하도록 사용자 정의 함수를 수정한다.

4.1.27 IRE-10029

- ❒ 에러 메시지
  행의 개수가 0인 값은 항목에 할당할 수 없습니다.
- ❒ 원인
  항목을 새로이 할당하여 룰을 호출하는 문법({룰:[항목]=값})에서 "값"이 다중 값인 경우 행의 개수는 적어도 1개 이상이 되어야 한다.
- ❒ 조치 방법
  룰 개발자는 항목에 할당할 값의 행이 1개 이상이 되도록 룰을 수정한다.

4.1.28 IRE-10030

- ❒ 에러 메시지
  병합할 두 값의 행의 개수가 일치하지 않습니다.
- ❒ 원인
  다음의 셀들,
  ○ 테이블 룰에서 동일한 행의 결과 셀들

○ 기본값의 셀들
○ 플로우 룰에서 하나의 결과 노드 셀들
에서 두 개 이상의 셀들에 다중 값이 사용된 경우 다중 값의 행이 개수는 동일해야 한다. 그렇지 않은 경우 이 에러가 발생한다. 메시지에 다중 값들의 행의 개수가 포함된다.

❒ 조치 방법
룰 개발자는 위 다중 값들의 행의 개수가 동일하도록 룰을 수정한다.

4.1.29 IRE-10031

❒ 에러 메시지
콜백 서비스에 대한 핸들러가 등록되어 있지 않습니다.

❒ 원인
룰 서비스를 실행 도중 콜백 룰을 실행하게 되었으나 콜백 룰에서 지정한 이름의 콜백 서비스가 룰 애플리케이션 환경에 등록되어 있지 않을 때 발생한다. 메시지에 콜백 서비스의 이름이 포함된다.

❒ 조치 방법
룰 시스템 관리자는 룰 애플리케이션 환경에 콜백 서비스를 등록한다.

4.1.30 IRE-10032

❒ 에러 메시지
콜백 핸들러를 생성하는 중 또는 핸들러가 서비스를 실행하는 도중 에러를 발생하였습니다.

❒ 원인
룰 서버는 콜백 서비스를 처리하기 위해 CallbackHandlerFactory.createCallbackHandler를 이용하여 CallbackHandler를 생성한다. CallbackHandler가 생성되면 이의 doCallback을 호출하여 서비스를 실행한다. 만약 createCallbackHandler 또는 doCallback에서 예외가 발생하면 이 에러가 발생한다. 예외의 메시지가 에러 메시지에 포함된다.

❒ 조치 방법
콜백 핸들러 개발자에게 에러 메시지를 전달하고 필요한 조치를 한다.

4.1.31 IRE-10033

❒ 에러 메시지
출력 콜백 파라미터들의 행의 개수가 올바르지 않습니다.

❒ 원인
콜백 핸들러가 서비스의 처리 결과를 두 개 이상의 콜백 출력 파라미터들에 저장을 할 때에는 각 파라미터의 행의 개수가 동일하도록 값을 설정해야 한다. 그렇지 않은 경우 이 에러가 발생한다. 에러 메시지에 문제가 되는 콜백 출력 파라미터의 이름이 포함된다.

❒ 조치 방법
콜백 서비스 개발자는 출력 파라미터 값의 행의 개수가 모두 동일하도록 콜백 핸들러를 수정한다. 자세한 내용은 Rule Application Programming Guide를 참조하라.

4.1.32 IRE-10034

- ❒ 에러 메시지
  콜백 출력 파라미터에 null 값이 포함되어 있습니다.
- ❒ 원인
  콜백 핸들러가 문자형 콜백 출력 파라미터에 null 값을 저장한 경우 이 에러가 발생한다. 에러 메시지에 문제가 되는 파라미터의 이름이 포함된다.
- ❒ 조치 방법
  콜백 핸들러 개발자는 문자형 콜백 출력 파라미터에 null 값을 저장하지 않도록 콜백 핸들러를 수정한다.

4.1.33 IRE-10035

- ❒ 에러 메시지
  루프 룰 항목 값들의 행의 개수가 일치하지 않습니다.
- ❒ 원인
  두 개 이상의 루프 항목에 다중 값이 설정되는 경우 이 다중 값들의 행의 개수가 일치하지 않으면 이 에러가 발생한다. 문제가 되는 루프 항목의 ID가 메시지에 포함된다.
- ❒ 조치 방법
  룰 개발자는 루프 항목에 설정되는 다중 값들의 행의 개수가 일치하도록 룰을 수정한다.

4.1.34 IRE-10036

- ❒ 에러 메시지
  룰 서버에 추적 기능이 비활성화 되어 있습니다.
- ❒ 원인
  추적 기능이 비활성화 되어 있는 룰 서버에 룰 서비스 요청을 할 때 실행 경로를 작성하도록 한 경우 이 에러가 발생한다.
- ❒ 조치 방법
  이 룰 서버에 추적 기능을 활성화 하거나 추적 기능이 활성화 되어 있는 룰 서버를 이용하여 실행 경로를 작성한다.
  룰 서버의 추적 기능 활성화 또는 비활성화는 마법사로는 할 수 없으면 기술지원을 요하는 사항이다. 기본적으로 추적 기능은 활성화 되어 있다.

4.1.35 IRE-10037

- ❒ 에러 메시지
  룰의 결과가 없습니다.
- ❒ 원인
  단일 값을 가져야 하는 룰을 실행하였으나 결과가 없을 때 발생한다. 결과가 없는 경우는 다

음의 경우를 포함한다.

- 테이블에서 모든 조건을 만족하는 행이 하나도 없는 경우
- DB 룰에서 쿼리의 결과가 0행이며 기본값도 설정이 되지 않은 경우
- Decision 테이블 또는 데이터 룰의 경우 조건을 만족하는 행이 없는 경우

각 룰 형식별로 결과가 없는 것으로 간주되는 경우는 InnoRules User's Guide of Rule Builder를 참조하라.

❒ 조치 방법

룰 개발자는 단일 값을 가져야 하는 룰이 결과를 필히 갖도록 룰을 수정한다. 만약 예외 조건이 있는 경우 ERROR 함수 등을 이용하여 업무적으로 의미 있는 에러 메시지가 나갈 수 있도록 한다.

4.1.36 IRE-10038

❒ 에러 메시지

바인드할 값의 행이 개수가 0입니다.

❒ 원인

DB 룰에서 바인드 문법(<$$ 표현식 $$>)에서 표현식의 값의 행의 개수가 0인 경우 발생한다.

❒ 조치 방법

룰 개발자는 바인드 문법 내에 다중 값이 사용된 경우 행의 개수가 0이 되지 않도록 룰을 수정한다.

4.1.37 IRE-10039

❒ 에러 메시지

조건에 부합되는 결과가 하나 이상 존재함.

❒ 원인

디시즌 테이블 또는 데이터 룰에서 조건을 만족하는 행이 두 개 이상 존재할 때 이 에러가 발생한다. 룰이 단일인가 다중인가는 관계가 없다.

❒ 조치 방법

룰 개발자는 조건을 만족하는 행이 하나만이 존재하도록 룰 또는 데이터를 수정한다. 만약 다중 룰이라면 "만족하는 조건 모두 실행"을 선택할 수도 있다. 어떤 방법을 선택할지는 업무적인 관점을 고려해야 한다.

4.1.38 IRE-10040

❒ 에러 메시지

데이터 룰을 실행하는 도중 데이터베이스 에러가 발생하였습니다.

❒ 원인

데이터 룰을 실행하기 위해 데이터베이스에 쿼리를 하였으나 JDBC 수준에서 에러가 발생한 경우 이 에러가 발생한다. 메시지에 JDBC 메시지가 포함된다.

❒ 조치 방법

JDBC 메시지에 따라 조치한다. 데이터베이스 서버에서 문제가 발생한 경우 우선적으로 DBA에게 문의한다.

4.1.39 IRE-10041

❒ 에러 메시지

데이터 룰 쿼리의 결과를 룰 시스템으로 변환할 수 없습니다.

❒ 원인

데이터 룰 쿼리를 실행하고 그 결과의 각 컬럼 값을 대응하는 룰 리턴의 데이터 형식으로 변환하려고 하였으나 변환이 지원되지 않을 때 이 에러가 발생한다. 다음은 룰 리턴의 형식에 따른 허용되는 컬럼의 데이터 형식이다. 여기서, 컬럼의 형식이라 함은 컬럼의 SQL 타입이 아니라 ResultSet.getObject()를 했을 때 반환되는 형식이다.

- 숫자형

  java.lang.Number의 하위 클래스인 경우 또는 객체의 toString()의 값이 Double.valueOf()를 이용하여 Double로 변환될 수 있는 모든 클래스
- 문자형

  toString()을 이용하여 문자열로 변환할 수 있는 모든 클래스
- 논리형

  java.lang.Boolean 클래스 또는 객체의 toString()의 값이 Boolean.valueOf()를 이용하여 Boolean으로 변환될 수 있는 모든 클래스

에러 메시지에

❒ 조치 방법

룰 개발자는 룰의 리턴 형식을 데이터베이스의 형식과 호환이 되도록 변경한다.

4.1.40 IRE-10042

❒ 에러 메시지

Unregistered rule interface cluster

❒ 원인

룰 애플리케이션이 클러스터 관리자에 ClusterManager.get(clusterName) 또는 ClusterManager.getInterface(clusterName)을 이용하여 룰 인터페이스 클러스터를 요청하였으나 이 이름의 룰 인터페이스 클러스터가 등록되어 있지 않은 경우 이 에러가 발생한다. 클러스터의 이름이 메시지에 포함된다.

❒ 조치 방법

클러스터가 등록되지 않은 것은 다음의 이유가 있을 수 있다.

- 룰 애플리케이션이 잘못된 클러스터 이름을 전달한 경우. 룰 애플리케이션 개발자는 룰 시스템 관리자에 문의하여 이름을 수정한다. 만약 설정에서 누락된 경우 룰 시스템 관리자는 설정에 추가한다.
- 설정에는 추가되어 있으나 초기화 단계에 룰 인터페이스 클러스터가 생성 도중에 에러가 발생된 경우. 룰 시스템 관리자는 로그를 확인하고 룰 인터페이스가 생성되지 않은 원인을 확인하고 조치한다.

4.1.41 IRE-10043

- ❒ 에러 메시지
  DB룰 쿼리의 셀렉트 아이템의 개와 룰 컬럼의 개수가 다릅니다.
- ❒ 원인
  DB룰 쿼리의 셀렉트 아이템의 개수는 그 룰의 리턴의 개수와 같아야 한다. 그렇지 않은 경우 이 에러가 발생한다. 룰의 ID와 쿼리의 셀렉트 아이템의 개수, 룰의 리턴의 개수가 메시지에 포함된다.
- ❒ 조치 방법
  룰 개발자는 쿼리 또는 룰 정보를 수정하여 셀렉트 아이템의 개수와 룰의 리턴의 개수를 일치시킨다.

4.1.42 IRE-10044

- ❒ 에러 메시지
  DB룰을 업데이트 용도로 사용하기 위해서는 데이터베이스의 커넥션이 자동 커밋(auto commit)으로 설정되어 있어야 합니다.
- ❒ 원인
  UPDATE 문으로 이루어진 DB룰을 실행하기 위해서는 DB 룰 실행에 사용되는 데이터베이스 연결의 자동 커밋 옵션이 설정이 되어 있어야 한다. 그렇지 않은 경우, 이 에러가 발생한다. DB룰이 SELECT 문으로 이루어진 경우에는 해당이 되지 않는다.
- ❒ 조치 방법
  룰 시스템 관리자는 DB 룰이 사용하고 있는 데이터소스에 자동 커밋 옵션을 활성화시킨다. 데이터소스의 관리 주체에 따라서 WAS와 같은 애플리케이션 프레임워크 담당자의 협조를 받아야 할 수도 있다.

4.1.43 IRE-10045

- ❒ 에러 메시지
  숫자형 컬럼 하나로 이루어진 DB룰만 업데이트 용도로 사용할 수 있습니다.
- ❒ 원인
  UPDATE 문으로 이루어진 DB룰은 필히 숫자형 리턴 하나만으로 이루어져 있어야 한다. 그렇지 않은 경우, 이 에러가 발생한다. 단, 룰이 단일인가 다중인가는 관계가 없다.
- ❒ 조치 방법
  룰 개발자는 숫자형 리턴 하나만을 갖도록 룰의 정보를 수정한다.

4.1.44 IRE-10046

- ❒ 에러 메시지
  DB룰을 실행하는 도중 데이터베이스 에러가 발생하였습니다.

❒ 원인

DB룰 실행을 위해 SQL을 실행하였으나 데이터베이스에서 에러가 발생한 경우이다. 에러의 원인은 SQL의 문법, 데이터 결함, 통신상의 문제, 데이터 소스 레벨의 문제 등 여러 가지가 있을 수 있다. 정확한 원인은 메시지에 포함된 데이터베이스 에러를 이용하여 찾아야 한다.

❒ 조치 방법

데이터베이스 에러 메시지를 확인하고 원인을 찾아 조치한다.

4.1.45 IRE-10047

❒ 에러 메시지

DB룰을 실행하는 도중 메모리 부족이 발생하였습니다.

❒ 원인

DB룰을 실행하는 도중 OutOfMemoryError가 발생한 경우이다. 메모리 부족의 원인이 이 DB룰이라고 단정할 수는 없다. JVM의 메모리가 지나치게 작게 설정되어 있거나 다른 애플리케이션에서 메모리 릭 등이 발생하여 가용한 메모리가 부족한 상태에서 DB룰이 실행되어 이 에러가 발생했을 수도 있다. 그러나, 많은 경우 DB룰의 쿼리의 결과가 과도하게 많아 이 에러가 발생되는 경우가 있다. 만약 이 에러 메시지에 포함된 DB룰의 조회 건수가 과도하게 많을 경우 DB룰을 원인으로 추정해 볼 수도 있다.

❒ 조치 방법

메시지에 포함된 DB룰 조회 건수를 확인하고 과도하게 많은 경우 DB룰의 쿼리에 문제가 있는지 점검한다. 정상적인 쿼리임에도 불구하고 조회 건수가 많은 경우 룰 구성을 다시 하여 조회 건수를 줄이도록 한다. 만약, DB 룰의 조회 건수가 얼마되지 않음에도 불구하고 이 에러가 발생하였다면 시스템 관리자에게 문의하여 힙 분석을 통하여 메모리 부족의 원인을 찾아야 한다.

4.1.46 IRE-10048

❒ 에러 메시지

룰 인터페이스 릭이 의심됩니다.

❒ 원인

룰 애플리케이션이 룰 인터페이스 사용 후에 close 메소드를 호출하지도 않고 참조를 끊어버린 경우 발생한다. 룰 애플리케이션은 클러스터 관리자(또는 룰 인터페이스 클러스터)로부터 룰 인터페이스를 임대하여 사용한 후에 필히 close를 호출하여 인터페이스를 반환해야 한다.
메시지에 이 인터페이스를 임대해 간 스레드의 ID와 이름, 임대한 시간이 포함된다.

❒ 조치 방법

룰 시스템 관리자는 스레드 정보와 임대 시각을 이용하여 룰 인터페이스를 반환하지 않은 애플리케이션을 찾고, 이 애플리케이션 개발자는 룰 인터페이스를 반환하도록 애플리케이션을 수정한다.

4.1.47 IRE-10049

❒ 에러 메시지

룰 인터페이스 클러스터에 가용한 풀이 없습니다.

❒ 원인

룰 인터페이스 클러스터에는 1개 이상의 룰 인터페이스 풀이 존재하며, 애플리케이션이 룰 인터페이스를 요청할 때 고가용 정책에 따라 이 중 하나의 풀에서 룰 인터페이스를 제공한다. 모든 인터페이스 풀(또는 이 풀과 연결된 룰 서버)이 장애, 셧다운되어 사용할 수 있는 풀이 없을 때 이 에러가 발생한다.

❒ 조치 방법

룰 시스템 관리자는 룰 인터페이스 풀이 룰 서버를 사용할 수 없는 이유를 조사하고 이를 복구한다. 복구가 불가능한 경우, 이 인터페이스 클러스터를 사용하는 룰 애플리케이션은 룰 서비스를 호출할 수가 없게 되므로 경우에 따라서는 애플리케이션을 중지시켜 다른 정상적인 애플리케이션들이 대신 서비스를 하도록 해야 한다(만약 가능하다면).

4.1.48 IRE-10050

❒ 에러 메시지

룰 인터페이스 클러스터에 가용한 인터페이스가 없습니다.

❒ 원인

룰 인터페이스 클러스터의 가용한 모든 인터페이스 풀의 룰 인터페이스가 사용 중인 상태에서 애플리케이션이 룰 인터페이스를 요청한 경우 발생한다. 애플리케이션의 부하가 룰 인터페이스 클러스터에 설정된 부하보다 과도한 경우 발생한다.

룰 인터페이스 클러스터의 when-exhaust-action이 block으로 설정된 경우에는 이 에러가 발생하지 않고 가용한 룰 인터페이스가 생길 때까지 인터페이스를 요청한 애플리케이션이 블록된다. 마법사를 이용한 경우 이 옵션은 항상 block으로 설정되므로 이 에러가 발생해서는 아니 된다. 만약 룰 인터페이스를 반환하지 않는 룰 애플리케이션이 단시간 내에 여러 번 수행되는 경우 IRE-10048 에러보다 이 에러가 먼저 발생할 수도 있다. 이후 GC가 발생할 때 IRE-10048 에러가 발생할 것이다.

❒ 조치 방법

동시에 실행되는 룰 애플리케이션의 개수를 줄이거나 룰 인터페이스 풀의 용량을 증가시키거나 또는 when-exhaust-action을 block으로 설정하는 것을 고려할 수 있다. 만약, 특정 애플리케이션이 단시간에 반복 실행된 이후에 이 에러가 발생했고 이후 IRE-10048이 발생했다면 인터페이스 릭을 의심할 수도 있다.

4.1.49 IRE-10051

❒ 에러 메시지

업무 데이터베이스로의 데이터베이스 연결을 가져오는 도중 에러가 발생하였습니다.

❒ 원인

DB룰, 데이터 룰 등을 실행하기 위해 지정된 업무 데이터베이스로의 연결을 가져오려고 시도하였으나 JDBC 에러가 발생한 경우이다. 에러의 정확한 원인은 JDBC 에러 메시지를 확인해야 한다.

❒ 조치 방법

JDBC 메시지를 확인하여 그에 따라 조치한다.

4.1.50 IRE-10052

- ❒ 에러 메시지
  단일 값을 가져야 하는 DB룰의 쿼리 결과의 형의 개수가 1이 아닙니다.
- ❒ 원인
  단일로 설정된 DB룰의 쿼리를 실행하였으나 쿼리 결과의 행의 개수가 1개가 아닌 경우에 발생한다.
- ❒ 조치 방법
  룰 개발자는 쿼리, 데이터 등을 확인하여 룰 변경, 데이터 변경을 한다.

4.2 IRE-11XXX

룰 서버가 룰 저장소에 접근하는 도중 발생할 수 있는 에러이다. 룰 서버가 룰 저장소에 접근하는 경우에는 다음의 경우가 있을 수 있다.

- ❒ 룰 서버가 초기화될 때
- ❒ 룰 서비스를 실행할 때
- ❒ 룰 빌더 서버가 룰 데이터를 변경하여 룰 저장소 캐시를 갱신하여야 할 때

4.2.1 IRE-11000

- ❒ 에러 메시지
  룰 DB로부터 데이터를 읽는 중 데이터베이스 에러가 발생했습니다.
- ❒ 원인
  룰 서버가 룰 DB에 룰 데이터를 쿼리를 하는 도중 JDBC 레벨의 에러가 발생한 경우이다. 메시지에 JDBC 에러 메시지가 포함된다. 정확한 원인은 JDBC 에러 메시지를 확인하여야 한다.
  메시지에는 어떤 룰 데이터를 접근하려다 에러가 발생했는가에 대한 정보도 포함된다.
- ❒ 조치 방법
  JDBC 에러 메시지를 확인하여 조치한다.

4.2.2 IRE-11001

- ❒ 에러 메시지
  JDBC 로더에서 데이터베이스 연결을 가져오지 못했습니다.
- ❒ 원인
  룰 DB에 접근하기 위해 데이터베이스 연결을 가져오는 중 에러가 발생한 경우이다. 메시지에 JDBC 에러 메시지가 포함된다. 정확한 원인은 JDBC 에러 메시지를 확인하여야 한다.
- ❒ 조치 방법
  JDBC 에러 메시지를 확인하여 조치한다.

4.2.3 IRE-11002

- ❒ 에러 메시지
  JDBC 로더가 네이밍 시스템으로부터 데이터소스를 찾지 못하였습니다.
- ❒ 원인
  JDBC 로더에 설정된 이름의 데이터소스를 네이밍 시스템으로부터 찾지 못하였을 경우 발생한다. 데이터소스를 생성하는 도중 문제가 있었을 수도 있다. 마법사를 사용하지 않은 경우 JNDI 설정과 JDBC 로더의 설정의 데이터소스 이름이 일치하지 않았을 수도 있다. 메시지에 JNDI 에러 메시지가 포함된다.
- ❒ 조치 방법
  JNDI 에러 메시지와 룰 서버의 로그 파일을 확인하여 조치한다. 데이터소스가 생성되는 중 에러가 발생하였다면 그 원인을 찾아 조치한다. JNDI 설정과 JDBC 로더의 설정이 부합하지 않는 경우 설정을 수정한다. 단, 이는 마법사를 사용한 경우는 해당하지 않는다.

4.2.4 IRE-11003

- ❒ 에러 메시지
  룰 DB에 시스템 정보가 없습니다.
- ❒ 원인
  룰 DB에는 그 DB의 데이터가 어느 단위 룰 시스템인지에 대한 정보가 저장되어 있다. 만약 룰 DB가 손상되어 그 정보가 없다면 이 에러가 발생한다.
- ❒ 조치 방법
  룰 DB가 손상된 것일 수 있으므로 기술지원을 요청한다.

4.2.5 IRE-11004

- ❒ 에러 메시지
  타임스탬프가 일치하지 않습니다.
- ❒ 원인
  룰 빌더 서버는 룰 데이터가 변경되면 변경된 룰 요소의 ID와 타임스탬프를 룰 서버에 전달한다. 타임스탬프에는 룰 데이터의 변경 시점이 기록되어 있으며 DB의 룰 데이터와 룰 빌더의 룰 데이터, 룰 서버에서 캐시하고 있는 데이터의 일치를 보증하는 데에 사용된다.
  만약 룰 서버가 룰 데이터를 갱신한 후에 룰 DB의 타임스탬프와 룰 빌더 서버가 보내온 타임스탬프가 일치하지 않는 경우 이 에러가 발생한다. 룰 서버의 룰 데이터 캐시에는 변경된 내용이 반영되지 않는다.
  이 경우는 잘못된 구성 설정으로 룰 빌더 서버와 룰 서버가 접근하는 룰 DB가 다른 경우 발생할 수 있다.
- ❒ 조치 방법
  구성 설정을 확인하여 룰 빌더 서버와 룰 서버가 동일한 룰 DB를 접근하는지를 확인하고 그렇지 않을 경우 구성을 변경한다. 단, 마법사를 사용하는 경우 이러한 문제가 발생하여서는 아니 되며 발생하는 경우 기술지원을 요청한다.

4.3 IRE-12XXX

룰 런타임의 설정이 잘못되었거나 초기화 도중 발생할 수 있는 에러들이다. 에러 코드에 추가하여 이유 코드가 포함될 수도 있다. 이유 코드는 자세한 문제점을 알려준다. 코드에 * 표시가 된 에러들은 마법사를 이용한 경우 발생해서는 아니 되는 에러이다. 만약에 발생한다면, 그 에러 이전에 발생한 다른 에러들에 대한 조치를 먼저 하고 그 후에도 에러가 발생한다면 기술지원을 요청하라.

4.3.1 IRE-12100*

- ❒ 에러 메시지
  JDBC 로더에 데이터소스 이름 설정이 누락되었습니다.
- ❒ 원인
  JDBC 로더의 설정에 데이터소스 이름이 누락되어 있다.
- ❒ 조치 방법
  JDBC 로더의 설정에 dsname 프로퍼티를 이용하여 데이터소스 이름을 설정한다.

4.3.2 IRE-12101*

- ❒ 에러 메시지
  TCP 모니터의 설정이 잘못되었습니다.
- ❒ 이유 코드
  - ○ 00: 접속 문자열이 지정되지 않음
  - ○ 01: 타임아웃 설정이 올바른 숫자 값이 아님
  - ○ 02: 재접속 간격 설정이 올바른 숫자가 아님
  - ○ 03: 모니터 서버 주소를 알아낼 수 없음
  - ○ 04: 모니터 서버 포트 설정이 올바른 숫자 값이 아님

4.3.3 IRE-12102*

- ❒ 에러 메시지
  업무 데이터베이스에 대한 데이터소스를 네이밍 시스템에서 찾을 수 없습니다.
- ❒ 이유 코드
  - ○ 00: 업무 데이터베이스에 대한 데이터소스가 네이밍 시스템에 등록되어 있지 않음

4.3.4 IRE-12103*

- ❒ 에러 메시지
  룰 저장소 캐시 설정에 문제가 있습니다.
- ❒ 이유 코드
  - ○ 00: 동일한 이름의 룰 저장소 캐시가 이미 등록되어 있음
  - ○ 01: 룰 저장소 캐시가 지정되지 않음

- 02: 룰 저장소 객체를 생성할 수 없음
- 03: 룰 로더가 지정되지 않음
- 04: 룰 로더 객체를 생성할 수 없음

4.3.5 IRE-12104*

- 에러 메시지
  업무 데이터베이스 설정이 누락되었습니다.
- 이유 코드
  - 00: 업무 데이터베이스에 대한 데이터소스가 설정되지 않음

4.3.6 IRE-12105*

- 에러 메시지
  사용자 정의 함수 설정에 문제가 있습니다.
- 이유 코드
  - 00: 동일한 이름의 사용자 정의 함수가 이미 등록된 경우

4.3.7 IRE-12106*

- 에러 메시지
  룰 수잇 설정에 문제가 있습니다.
- 이유 코드
  - 00: 이 형식의 룰에 대한 룰 수잇은 등록되지 않았음
  - 01: 이 형식의 룰에 대한 룰 수잇이 이미 등록되어 있음
  - 02: 룰 수잇 클래스 이름이 지정되지 않음
  - 03: 룰 수잇 클래스를 로드할 수 없거나 생성할 수 없음
  - 04: 이 형식의 룰에 대한 루 수잇 로더는 등록되지 않았음
  - 05: 룰 수잇 로더 클래스 이름이 지정되지 않았음
  - 06: 룰 수잇 로더 클래스를 로드할 수 없거나 생성할 수 없음

4.3.8 IRE-12107*

- 에러 메시지
  룰 인터페이스 클러스터 설정이 잘못되었습니다.
- 이유 코드
  - 00: 룰 인터페이스 풀 설정이 없음
  - 01: 콜백 클래스 경로가 유효하지 않음
  - 02: 콜백 핸들러 클래스를 로딩할 수 없음
  - 03: 콜백 핸들러 팩토리를 생성할 수 없음
  - 04: 콜백 설정 파일을 읽어들일 수 없음

- 05: 설정 XML 오류
- 06: 이미 같은 이름으로 룰 인터페이스 클러스터가 등록되어 있음
- 07: 설정에 룰 인터페이스 클러스터의 이름이 누락됨

4.3.9 IRE-12108*

❒ 에러 메시지

룰 인터페이스 풀 설정이 잘못되었습니다.(일반)

❒ 이유 코드

- 00: 룰 인터페이스 팩토리 클래스를 찾을 수 없음
- 01: 룰 인터페이스 팩토리 객체를 생성할 수 없음
- 02: 설정의 값이 올바른 숫자 값이 아님
- 03: 최대 활성 인터페이스 개수는 0보다 커야 함
- 04: 룰 인터페이스를 생성 중 에러가 발생

4.3.10 IRE-12109*

❒ 에러 메시지

룰 인터페이스 풀 설정이 잘못되었습니다.(로컬 룰 서버)

❒ 이유 코드

- 00: 룰 저장소 캐시의 이름이 누락됨
- 01: 룰 저장소 캐시가 등록되지 않음
- 02: DB룰 타임아웃 설정이 올바르지 않음.
- 99: 기타 오류. 메시지에 따라 조치

4.3.11 IRE-12110*

마법사를 이용한 경우라도 이유 코드 08은 발생할 수 있다. 이는 TCP 커넥터 서버가 활성화된 InnoRules 서버가 중지되어 있는 경우, 이 서버를 포함하는 TCP 커넥터 서버 그룹을 사용하는 룰 애플리케이션에서 이 에러가 발생할 수 있다. 이 에러가 발생하더라도 TCP 커넥터 서버 그룹의 고가용 정책에 의해 다른 TCP 커넥터 서버를 사용할 수 있다면 룰 서비스 호출에는 문제가 없다.

❒ 에러 메시지

룰 인터페이스 풀 설정이 잘못되었습니다.(TCP)

❒ 이유 코드

- 00: 포트가 지정되지 않음
- 01: 포트 값이 올바르지 않음. 양수가 아님
- 02: 연결 타임아웃 값이 올바르지 않음. 0또는 양수이어야 함
- 03: 타임아웃 값이 올바르지 않음. 0 또는 양수이어야 함
- 04: 통신 버퍼의 크기가 올바르지 않음. 최소 1024바이트이어야 함
- 05: 설정의 값이 올바른 숫자 값이 아님

- 06: 원격 룰 서버에 로그인 하는 도중 서버가 연결이 끊음
- 07: 원격 룰 서버에 로그인 하는 도중 IOException이 발생함
- 08: 원격 룰 서버에 연결을 실패함

4.3.12 IRE-12111*

❒ 에러 메시지

룰 인터페이스 풀 설정이 잘못되었습니다.(스펙트럼)

❒ 이유 코드

- 00: 스펙트럼 이름이 지정되지 않음
- 01: 동일한 이름의 스펙트럼 인터페이스 팩토리가 이미 등록되어 있음
- 02: 지정한 이름의 스펙트럼 인터페이스 팩토리가 스펙트럼 관리자에 등록되어 있지 않음. 스펙트럼을 스펙트럼 관리자에서 제거할 수 없음
- 03: 지정한 스펙트럼은 이미 모니터링 중임
- 04: 지정한 스펙트럼은 모니터링 중이 아님
- 05: 스펙트럼 분석에 사용할 대리 인터페이스 팩토리가 지정되지 않음
- 06: 스펙트럼 분석에 사용할 대리 인터페이스 팩토리를 찾을 수 없음

4.3.13 IRE-12112

❒ 에러 메시지

룰 변경 모니터링 중에 에러가 발생하였습니다.

❒ 이유 코드

- 00: 모니터 서버에 연결하지 못함. 활성화된 룰 빌더 서버가 활성화된 InnoRules 서버가 중지되어 있을 경우, 그리고 룰 서버가 이 모니터 서버에 연결을 시도했을 경우 발생할 수 있다. 룰 서버는 다른 빌더 서버에 다시 모니터 연결을 시도할 것이며, 동작 중인 다른 룰 빌더 서버에 연결하게 될 것이다. 룰 시스템 관리자는 로그를 이용하여 재연결 시도 후에 모니터 연결이 되었는가를 확인한다.
- 01: 모니터 서버가 연결을 끊음. 모니터 연결이 되어 있는 룰 빌더 서버가 종료되면 이 에러가 발생할 수 있다. 룰 시스템 관리자는 로그를 확인하여 재연결을 시도하였는지 그리고 연결이 성공했는지 확인한다.
- 02: I/O 에러가 발생. 기술지원 요청
- 03: 모니터 이벤트 핸들러 작업 도중 예외를 던짐. 기술지원 요청
- 99: 예상하지 못한 에러. 기술지원 요청

4.3.14 IRE-12113

❒ 에러 메시지

애플리케이션이 사용 중인 룰 저장소 캐시를 종료하려고 합니다.

❒ 원인

룰 서비스 실행 도중 룰 서버를 종료한 경우 이 메시지가 나타날 수도 있다. 룰 서버가 동작

중인 애플리케이션 프레임워크가 애플리케이션이 동작 중임에도 불구하고 룰 런타임을 종료하면 발생한다.

❒ 조치 방법

애플리케이션 프레임워크 수준에서 애플리케이션이 실행 중일 때는 룰 런타임을 종료하지 않도록 수정한다.

4.3.15 IRE-12200

❒ 에러 메시지

License Error

❒ 이유 코드

라이선스와 관련된 에러이다.

- 00: 설정 파일에 라이선스 정보가 누락된 경우. 마법사를 이용하는 경우 발생하지 않는다.
- 01: 지정된 라이선스 URL이 존재하지 않는 위치를 가리키거나 권한이 없는 경우. 라이선스 파일이 있는지 룰 애플리케이션 또는 InnoRules 서버가 라이선스 파일을 읽을 권한이 있는지 확인한다.
- 02: 네이밍 시스템에 라이선스를 바인딩 하는 도중 오류. 세부 메시지를 첨부하여 기술지원 요청
- 03: 라이선스가 이미 네이밍 시스템에 바인딩 되어 있음. 세부 메시지를 첨부하여 기술지원 요청
- 04: 올바른 형식의 라이선스 파일이 아니거나 파일이 손상됨. 손상되지 않은 라이선스 파일로 교체한다.
- 05: 라이선스 기간이 만료됨. InnoRules 구입처에 문의
- 06: 시스템 프로세서의 개수가 라이선스 제한을 초과함. 시스템의 사양을 낮추거나 InnoRules 구입처에 문의
- 07: 등록되지 않은 단위 시스템. InnoRules 구입처에 문의
- 08: 룰 개수가 라이선스의 제약을 초과함. InnoRules 구입처에 문의
- 09: 라이선스 정보를 네이밍 시스템에서 찾을 수 없음. 세부 메시지를 첨부하여 기술지원 요청
- 10: 사용자 계정의 수가 라이선스 제약을 초과함. InnoRules 구입처에 문의
- 99: 기타 라이선스 에러. 세부 메시지를 첨부하여 기술지원 요청

4.3.16 IRE-12300*

❒ 에러 메시지

TCP 커넥터 서버의 설정에 문제가 있습니다.

❒ 이유 코드

- 00: 서버의 이름이 설정되어 있지 않음
- 01: 동일한 이름의 서버가 이미 등록되어 있습니다.

4.3.17 IRE-12301

- ❒ 에러 메시지
  TCP 커넥터 서버 시동 중에 에러가 발생하였습니다.
- ❒ 원인
  TCP 커넥터 서버 시동 중에 에러가 발생하여 서버 실행이 실패한 경우 발생한다. 서비스 포트가 이미 사용 중이거나 MBean 서버에 등록 도중 JMX 예외가 발생했을 수도 있다. 세부적인 원인은 추가 메시지를 참조하라.
- ❒ 조치 방법
  에러 메시지에 포함된 추가 메시지를 확인하여 조치한다.

4.3.18 IRE-12997

- ❒ 에러 메시지
  MBean 서버에 MBean을 등록 또는 해제하는 중 오류가 발생하였습니다.
- ❒ 원인
  룰 시스템의 각 요소는 관리를 위해 MBean 서버에 등록을 한다. 이 때 문제가 발생하면 이 에러가 발생한다. 세부적인 메시지가 메시지에 포함된다.
- ❒ 조치 방법
  에러 메시지를 확인하여 조치한다.

4.3.19 IRE-12998

- ❒ 에러 메시지
  인터럽트가 발생하였습니다.
- ❒ 원인
  룰 런타임에는 백그라운드로 동작하는 스레드들이 있으며 동기화를 위해 다양한 락들을 사용한다. 룰 런타임은 대기 상태에 있는 스레드를 깨우기 위해서 Thread.interrupt()를 호출하는데 이 때 이 에러가 발생한다. 룰 런타임이 이 메소드를 호출하는 경우는 다음과 같다.
  - ○ 룰 런타임이 종료될 때. 즉, 룰 애플리케이션 또는 InnoRules 서버가 종료될 때
  - ○ 빌더 서버 클러스터링에서 마스터 빌더 서버가 슬레이브로 전환될 때

  Thread.interrupt()는 애플리케이션 프레임워크가 종료될 때 프레임워크에 의해서 호출되기도 한다.
- ❒ 조치 방법
  위에서 열거한 경우에는 이 에러는 문제가 되지 않으며 무시해도 된다. 그러나, 시스템 운영 중에 이유 없이 나타나는 경우 기술지원을 요청하는 것이 좋다.