JEUS JMX 안내서

Tmax Soft

Document info

Document name: JEUS JMX 안내서

Document date: 2005-06-06

Manual release version: 3

Software Version: JEUS 5

차 례

그림목차

표 목차

매뉴얼에 대해서

매뉴얼의 대상

본 매뉴얼은 JEUS 에서 JMX 를 관리하고, 유지하는 시스템 관리자나 관계자를 대상으로 한다.

매뉴얼의 전제조건

분 문서를 읽기 전에 다음의 두 가지 전제 조건을 만족해야 한다.

1. JEUS Server 안내서를 읽고, 이해 해야 한다.
2. JMX개발에 대한 기본 지식이 있어야 한다. JMX에 대해 잘 모른다면 http://java.sun.com에서 스펙이나 기타 JMX관련 문서를 먼저 보길 바란다.

주의 : 이 매뉴얼에서는 J2EE 나 JMX 기술에 대한 기본적인 내용은 다루지 않는다.

매뉴얼의 구성

본 매뉴얼은 5 개의 장으로 나눠져 있다.

1. 소개: 인사와 개요
2. 따라 하기: 간단하게 따라해 보기
3. JEUS JMX Manager 설정: JMX Manager 에 대한 설정
4. JMX 어플리케이션 개발: JMX 클라이언트 어플리케이션 개발
5. 결론

부록

A 모니터링 정보 레퍼런스: JMX 모니터링 정보에 대한 레퍼런스를 설명한다.

B JEUSMain.xml XML 설정 레퍼런스: JEUSMain.xml구성 파일에 대한 참조

관련 매뉴얼

관련 문서들은 다음과 같다.

- J2EE 1.4 Specification
- J2EE management 1.0 Specification
- JMX 1.2 Specification
- JMX Remote API 1.0 Specification
- JEUS Server 안내서

일러두기

표기 예	내용
텍스트	본문, 12 포인트, 바탕체 Times New Roman
텍스트	본문 강조
CTRL+C	Ctrl 과 동시에 C 를 누름
`public class myClass { }`	자바 코드
`<system-config>`	XML 문서
참조: / 주의:	참조 사항과 주의할 사항

표기 예	내용
Configuration 메뉴를 연다	GUI 의 버튼 같은 컴포넌트
JEUS_HOME	JEUS 가 실제로 설치된 디렉토리 예)c:\jeus50
`jeusadmin nodename`	콘솔 명령어와 문법
[파라미터]	옵션 파라미터
< xyz >	'<'와 '>' 사이의 내용이 실제 값으로 변경됨. 예)<node name>은 실제 hostname 으로 변경해서 사용
|	선택 사항. 예) A|B: A 나 B 중 하나
…	파라미터 등이 반복되어서 나옴
?, +, *	보통 XML 문서에 각각 "없거나, 한 번", "한 번 이상", "없거나, 여러 번"을 나타낸다.
. . .	XML 이나 코드 등의 생략
<<FileName.ext>>	코드의 파일명
그림 1.	그림 이름이나 표 이름

OS 에 대해서

본 문서에서는 모든 예제와 환경 설정을 Microsoft Windows™의 스타일을 따랐다. 유닉스같이 다른 환경에서 작업하는 사람은 몇 가지 사항만 고려하면 별무리 없이 사용할 수 있다. 대표적인 것이 디렉토리의 구분자인데, Windows 스타일인 "\"를 유닉스 스타일인 "/"로 바꿔서

사용하면 무리가 없다. 이외에 환경 변수도 유닉스 스타일로 변경해서 사용하면 된다.

그러나 Java 표준을 고려해서 문서를 작성했기 때문에, 대부분의 내용은 동일하게 적용된다.

용어설명

다음에 소개되는 용어는 본 문서 전체에 걸쳐서 사용되는 용어이다. 용어가 이해하기 어렵거나 명확하지 않을 때는 아래 정의를 참조하기 바란다.

용어	정의
HTML adaptor	'HTML adaptor'는 HTML 을 지원하는 프로토콜 어댑터이다. 프로토콜 어댑터들과 커넥터는 원격 관리 어플리케이션이 agent 로 접근할 수 있도록 해 준다.
HTML adaptor	"**HTML adaptor**"는 HTML 을 지원하는 JMX 의 프로토콜 어댑터이다.
JMX	JMX 는 "Java Management eXtensions"의 약자로, J2EE 스펙 중 하나이다. 이것은 자바 프로그래밍 언어의 아키텍처, 디자인 패턴, API, 그리고 네트웍 관리와 감시에 대한 서비스를 규정한다.
JMX agent	MBean 서버들을 관리 한다.
JMX client application	"Management Application"이라고 부르며, Client application 이다. 이것은 MBean 에 접근하고, JMX 에이전트를 사용함으로써 시스템을 모니터링 한다.
JMX manager	JMX 매니저는 JMX 에이전트 아키텍처의 컴포넌트이다. 이것은 JMX 의 분산 서비스를 제공한다.

용어	정의
JMXMP connector	"JMXMP(Message Protocol) connector"는 TCP 기반의 통신과 자바 직렬화를 이용하여 객체전송을 지원하는 일반적인 커넥터이다.
MBean	MBean 은 JMX 의 관리 리소스로, Resource 와 Instrumentation 을 구현한 객체로, Managed Bean 또는 MBean 으로 불린다. JMX Agent 가 사용한다.
MBean server	Managed Bean Server 를 말하며, MBean 객체가 등록되는 서버이다. MBean 서버에 등록된 어떠한 객체라도 어플리케이션에서 액세스할 수 있다.
MBeanServerConnection	"MBeanServerConnection"은 JMX 1.2 의 새로운 인터페이스이다. MBeanServer 로의 접근을 커넥터를 통해 로컬이나 리모트로 접근할 수 있고, 자바 객체를 통해 직접적으로 접근할 수도 있다.
mlet	"mlet" 또는 "m-let" 은 "Management Applet"의 약어이다. Mlet 서비스를 사용하면, 리모트에서 MBean 을 인스턴스화하거나 등록할 수 있다.
ObjectName	"ObjectName"은 MBean 서버 내에서 유일하게 MBean 을 식별하는 이름이다. 어플리케이션이 관리하려는 MBean 을 식별하기 위해서 사용한다.
RMI connector	"RMI connector"는 remote 에서 JMX MBeanServer 에 접근하기 위해 원격 메소드 호출을 사용하는 컨넥터이다.
SNMP adaptor	"SNMP adaptor"는 SNMP 를 지원하는 JMX 의 프로토콜 어댑터이다.

연락처

Korea
Tmax Soft Co., Ltd
18F Glass Tower, 946-1, Daechi-Dong, Kangnam-Gu, Seoul 135-708
South Korea
Tel: 82-2-6288-2114
Fax: 82-2-6288-2115
Email: info@tmax.co.kr
Web (Korean): http://www.tmax.co.kr

USA
Tmax Soft, Inc.
560 Sylvan Ave, Englewood Cliffs NJ 07632
USA
Tel: 1-201-567-8266
FAX: 1-201-567-7339
Email: info@tmaxsoft.com
Web (English): http://www.tmaxsoft.com

Japan
Tmax Soft Japan Co., Ltd.
6-7 Sanbancho, Chiyoda-ku, Tokyo 102-0075
Japan
Tel: 81-3-5210-9270
FAX: 81-3-5210-9277
Email: info@tmaxsoft.co.jp
Web (Japanese): http://www.tmaxsoft.co.jp

China
Beijing Silver Tower, RM 1507, 2# North Rd Dong San Huan,
Chaoyang District, Beijing, China, 100027
Tel: 86-10-6410-6148
Fax: 86-10-6410-6144
E-mail : info@tmaxchina.com.cn
Web (Chinese): http://www.tmaxchina.com.cn

1 소개

JMX Remote API 1.0 스펙, 그리고 J2EE Managemant 스펙(JSR-77)과 더불어 Sun Microsystems Inc.의 JMX1.2 는 엔터프라이즈 모니터링 시스템 개발에 대한 표준적인 방법을 제시하고 있다.

JEUS 시스템에서 JEUS JMX 모듈은 JEUS 제품의 특성을 포함하고, J2EE Management 스펙(JSR-77)과 JMX Remote API 1.0 스펙을 유연하고 완벽하게 구현하고 있다.

이 매뉴얼에서는 JEUS JMX 모듈의 사용을 위해 알아야 할 모든 내용을 설명한다. 그리고, 2장 따라 하기에서 전체적인 모듈의 동작을 간략하게 살펴 본다. 그 후에 각각의 서브 모듈에 대해서 면밀히 살펴 보고, 환경을 설정하는 방법(3장 JEUS JMX Manager 설정)과 JEUS Monitoring Service를 사용하기 위한 자바 어플리케이션에서 어떻게 사용되는지(4장 JMX 어플리케이션 개발) 알아본다. “A 모니터링 정보 레퍼런스”에서는 JEUS 서버에 의해서 관리되는 Resource와 MBean을 참조하는 방법에 대해서 살펴 본다. “B JEUSMain.xml XML 설정 레퍼런스”에서는 JEUSMain.xml 구성 파일에 대해 살펴본다.

2 따라 하기

이 장에서는 JEUS JMX Manager 에 대한 시작, 종료하는 방법, 그리고 샘플 예제를 실행하는 방법 같은 기본적인 내용을 설명한다.

JEUS 를 설치 한 후에는 다음과 같은 작업이 필요하다. JEUS 설치를 하지 않았다면 JEUS 설치 안내서를 참조해서 JEUS 를 설치 하기 바란다.

1. 'java' 명령을 실행하려면 'JAVA_HOME' 이 세팅되어야 한다. 만일 Windows 플랫폼에서 java.exe 가 C:\lang\j2sdk1.4.0\bin 에 존재한다면, JAVA_HOME 은 다음과 같이 'C:\lang\j2sdk1.4.0' 으로 세팅한다.

```
C:\>set JAVA_HOME=C:\lang\j2sdk1.4.0
C:\>set JAVA_HOME
JAVA_HOME=C:\lang\j2sdk1.4.0
```

2. 'jeus' 명령을 실행하려면 'JEUS_HOME' 이 세팅되어야 한다. 만일 Windows 플랫폼에서 c:\jeus50\bin 에 jeus.bat 가 있다면, JEUS_HOME 은 다음과 같이 'C:\jeus50' 으로 세팅한다.

```
C:\>set JEUS_HOME=C:\jeus50
C:\>set JEUS_HOME
JEUS_HOME=C:\jeus50
```

3. 'JEUS_BASEPORT', 'EJB_HOME', 'CLIENT_HOME', 'WEBTODIR' 같은 환경변수는 위에서 설명한 방법과 같이 세팅해야 한다. JEUS Server 안내서에서 'JEUS Server 환경 변수' 절을 참조한다.

   참조: JEUS 를 정상적으로 설치했다면, JEUS_HOME\bin 디렉토리에 있는 jeus.properties 에 위 내용이 모두 적용되어 있다. 그러므로 위 단계를 실행하지 않아도 무방하다.

4. 'JEUSMain.xml' 내의 'jmx-manager' 태그에 다음의 XML 태그와 같이 "snmp-adaptor-port" 를 포함시킨다.

<<JEUSMain.xml>>

```
<node>
    <engine-container>
```

```
    ...
    </engine-container>
    ...
    <jmx-manager>
        <jmx-connector>
          <jmxmp-connector/>
        </jmx-connector>
        <html-adaptor-port>7070</html-adaptor-port>
        <snmp-adaptor>
           <snmp-adaptor-port>9999</snmp-adaptor-port>
           <snmp-version>2</snmp-version>
           <snmp-max-packet-size>4096</snmp-max-packet-size>
           <snmp-security>false</snmp-security>
        </snmp-adaptor>
    </jmx-manager>
    ...
</node>
```

참조: 'JEUSMain.xml' 파일은 'JEUS_HOME\config\<node name>'라는 디렉토리에 위치한다. JEUS Sever 안내서의 JEUS Server 안내서의 3.9 절 "JEUS 의 디렉토리구조"를 참조한다.

5. JEUS_HOME\bin 디렉토리에서 'jeus' 명령 스크립트를 실행한다.

```
C:\jeus50\bin>jeus
[2005.01.11 16:21:39][0] [johan -10] [MGR-0411]
virtual host name of this manager : johan
[2005.01.11 16:21:46][0] [johan -10] [SNMP-0004] SNMP
AGENT is ready on port: 9999
[2005.01.11 16:21:46][0] [johan -10] [MGR-0241]
JeusServer is Ready
```

6. 터미널/콘솔 창을 열고서 'jeusadmin'을 입력하고 그 뒤에 노드네임을 입력한다(예 'jeusadmin johan'). 원도우즈에서는 'hostname' 명령, 유닉스에서는 'uname –a' 명령을 통해서 컴퓨터 이름을 확인할 수 있다. 프롬프트에 JEUS 사용자 이름과 패스워드를 입력한다

```
C:\jeus50\bin>hostname
johan
C:\jeus50\bin>jeusadmin johan
Login name>jeus
```

```
Password>
JEUS 5.0 Jeus Manager Controller
johan >
```

주의: 이 매뉴얼에서는 JEUS 노드 이름을 'johan'으로 가정한다.

7. jeusadmin 프롬프트가 나타나면 boot 를 입력하고 엔터키를 친다. 다음의 예제를 참조한다.

```
JEUS 5.0 Jeus Manager Controller
johan > boot
johan boot done
johan_container1
johan_container2
```

8. 잠시후에 프롬프트가 콘솔 창에 다시 나타난다. 이것은 JEUS 서버가 현재 부트가 되어서 다른 명령을 받아들일 준비가 되어 있다는 것을 의미한다. 아래와 같이 'allenglist' 명령을 실행해서 엔진이 부트가 되었는지를 확인한다.

```
JEUS 5.0 Jeus Manager Controller
johan > boot
johan boot done
johan_container1
johan_container2
johan > allenglist
=======================================
engines in the container johan_container1
johan_ejb_engine1
johan_servlet_engine1
=======================================
=======================================
engines in the container johan_container2
johan_ejb_engine2
=======================================
johan >
```

9. 아래와 같이 'mbeanlist' 명령을 실행해서 현재 생성되어 있는 MBean list 를 확인 할 수 있다.

```
JEUS 5.0 Jeus Manager Controller
```

```
johan>mbeanlist
JEUS:name=johan,j2eeType=JeusService,jeusType=JeusManage
r,JMXManager=johan
JEUS:JNDIResourceService=johan,name=JNSServer,j2eeType=J
eusService,JeusManager=johan,jeus
e=ThreadPool,JMXManager=johan
JEUS:J2EEServer=null,name=johan,j2eeType=JeusService,Jeu
sManager=johan,jeusType=Scheduler
vice,JMXManager=johan
JEUS:name=jeus.server.ControlThread,j2eeType=JeusService
,JeusManager=johan,jeusType=Thre
ool,JMXManager=johan,NodeControllerService=johan
JEUS:name=johan,j2eeType=JeusService,jeusType=JEUSMPConn
ector,JMXManager=johan
JEUS:J2EEServer=null,name=johan,j2eeType=JeusService,Jeu
sManager=johan,jeusType=JMXExport
vice,JMXManager=johan
…
```

다음 장에서는 JEUS JMX Manager 에 대해 살펴보겠다.

3 JEUS JMX Manager 설정

3.1 소개

JMX Remote API 스펙 1.0 을 따르는 클라이언트 어플리케이션에게 JEUS JMX 는 JEUS 의 구성과 실시간 정보를 제공해준다.

JEUS JMX Manager 는 JMX Remote API 커넥터, HTML 어댑터, SNMP 어댑터와 같은 관리 객체를 가지고 있다. 이것은 JEUS 모니터링 정보에 액세스하는 방법으로 세 가지를 제공한다는 것이다.

이번 장에서는 위에 언급한 것에 대한 개념들과 설정들에 대해서 설명한다. 그리고, 이번 장은 JEUS JMX 어플리케이션의 관리 안내서의 역할도 한다.

JMX Manager 의 설정은 JEUS_HOME\config\<node name> 디렉토리 안의 JEUSMain.xml 에서 한다. 설정 태그인 <jmx-manager> 태그는 <node> 태그나 <engine-container> 태그 안에서 설정할 수 있다.

설정 예:

<<*JEUSMain.xml*>>

```
<?xml version="1.0"?>
<jeus-system xmlns="http://www.tmax.co.kr/xml/ns/jeus">
<!-- The above XML header will not be repeated in the examples below
-->

<node>
    <engine-container>
        ...
        <jmx-manager>
            ...
        </jmx-manager>
    <engine-container>
    ...
    <jmx-manager>
        ...
```

```
    </jmx-manager>
<node>
```

3.2 JMX Connector 설정

JMX Connector 는 JMX Remote API 에서 정의되어 있는, remote 에서 JMX MBeanServer 에 접근하기 위한 Connector 이다. Remote API 에 정의되어 있는 RMI Connector 와 Socket 기반의 JMXMP Connector 가 제공되고 기본적으로는 JMXMP Connector 를 사용한다. 설정은 <jmx-manager> 내의 <jmx-connector> 태그로 설정하고 하위 태그의 상세한 설명은 부록의 Schema reference 를 참고하기 바란다. 설정 예는 다음과 같다.

<<*JEUSMain.xml*>>

```
<jmx-manager>
    <jmx-connector>
      <jmxmp-connector>
            <jmxmp-connector-port>5001</jmxmp-connector-port>
      </jmxmp-connector>
    </jmx-connector>
    <html-adaptor-port>7070</html-adaptor-port>
    <snmp-adaptor>
        <snmp-adaptor-port>9090</snmp-adaptor-port>
        <snmp-version>3</snmp-version>
        <snmp-max-packet-size>4096</snmp-max-packet-size>
        <snmp-security>true</snmp-security>
    </snmp-adaptor>
</jmx-manager>
```

3.3 HTML Adaptor 설정

‘HTML 어댑터’는 HTML 을 지원하는 JMX 의 프로토콜 어댑터이다. ‘HTML 어댑터’는 JEUSMain.xml 안의 `<jmx-manager>` 태그 안에 설정한다. 기본으로 설정할 태그는 다음의 리스트에서 설명한다.

- **html-adaptor-port:** HTML 어댑터의 리스너 포트

html-adaptor-port 태그는 필수 입력 사항(mandatory)이다. HTML 어댑터 포트를 ‘-1’로 설정을 하게 되면 JMX Manager 가 HTML 프로토콜을

사용하지 않음을 의미한다. 다른 서비스가 사용하는 포트를 사용하면 안 된다는 것을 유념한다. 설정 예는 다음과 같다.

<<*JEUSMain.xml*>>

```
<jmx-manager>
    <jmx-connector>
      <jmxmp-connector>
            <jmxmp-connector-port>5001</jmxmp-connector-port>
      </jmxmp-connector>
    </jmx-connector>
    <html-adaptor-port>7070</html-adaptor-port>
    <snmp-adaptor>
        <snmp-adaptor-port>9090</snmp-adaptor-port>
        <snmp-version>3</snmp-version>
        <snmp-max-packet-size>4096</snmp-max-packet-size>
        <snmp-security>true</snmp-security>
    </snmp-adaptor>
</jmx-manager>
```

3.4 SNMP Adaptor 설정

‘SNMP 어댑터’는 JMX 가 제공하는 SNMP 프로토콜 어댑터이다. ‘SNMP 어댑터’는 JEUSMain.xml 안의 `<jmx-manager>` 태그의 안에 설정한다. 기본으로 설정할 태그는 다음과 같다.

- **snmp-adaptor-port:** SNMP 어댑터의 리스너 포트.
- **snmp-version:** SNMP 버전을 지정하며 1, 2 또는 3 을 지정할 수 있으며, 기본 값은 3 이다.
- **snmp-max-packet-size**: SNMP 패킷에 대한 최대값을 설정하며 최소 256 바이트부터 설정 할 수 있다. 기본 값은 4096 이다.
- **snmp-security**: 보안을 적용시킬 것 인지를 설정한다. 'true’ 또는 ‘false’로 지정하며, 기본 값은 ‘false’이다. 그리고, ‘true’는 SNMP 버전 3 에서만 지정이 가능 하다.

‘SNMP 어댑터’에 대한 설정은 선택 사양이며, snmp-adaptor-port 태그는 필수 입력 사항(mandatory)이다. 다른 서비스가 사용하는 포트를 사용하면 안 된다는 것을 유념한다. 설정 예는 다음과 같다.

<<JEUSMain.xml>>

```
<jmx-manager>
    <jmx-connector>
      <jmxmp-connector>
            <jmxmp-connector-port>5001</jmxmp-connector-port>
      </jmxmp-connector>
    </jmx-connector>
    <html-adaptor-port>7070</html-adaptor-port>
    <snmp-adaptor>
        <snmp-adaptor-port>9090</snmp-adaptor-port>
        <snmp-version>3</snmp-version>
        <snmp-max-packet-size>4096</snmp-max-packet-size>
        <snmp-security>true</snmp-security>
    </snmp-adaptor>
</jmx-manager>
```

3.5 결론

지금까지 JMX Manager 에 대한 설정을 어떻게 하는지를 살펴봤다.

다음 장에서는 간략하게 JMX 어플리케이션 프로그래밍에 대한 설명을 한다.

4 JMX 어플리케이션 개발

4.1 소개

이번 장에서는 JEUS JMX 클라이언트 어플리케이션을 개발하는 방법과 그것을 설치하는 방법에 대해서 설명한다. J2EE JMX Remote API 1.0 과 J2EE Management 스펙에 대한 기본 지식이 있어야 한다.

JEUS JMX 를 사용하는데는 세 가지 방법이 있는데, JEUS 유틸리티 (MEJBUtility), JNDI 그리고 JMX Remote API 가 있다. SNMP 를 사용해서 JEUS 모니터링이 가능한데, 사용 방법은 JEUS SNMP 안내서에서 확인 할 수 있다.

보안을 적용하여 JMX 를 사용하기 위해서는 MBean 서버를 얻을 때 사용자 이름과 패스워드를 설정해야 한다.

마지막으로, JEUS JMX 에서 ObjectName 을 생성하는 규칙에 대해서 설명한다.

참조: JMX Remote API 에 대한 더 많은 정보를 원한다면 SUN 에서 제공하는 J2EE JMX Remote API 1.0 스펙과 JMX Remote API 를 참조 하기 바란다.

4.2 일반적인 어플리케이션 구조

이 절에서는 필요에 따라 다양한 형태가 있겠지만, JEUS JMX 클라이언트 어플리케이션의 일반적인 구조를 제시한다.

JMX 클라이언트 어플리케이션은 다음 단계를 수행한다.

- **Step 1: 환경 세팅(옵션 사항)**
- **Step 2: MBeanServer 연결**
- **Step 3: 질의**
- **Step 4: 질의 결과 처리**

Step 1 부터는 JMX 클라이언트 어플리케이션을 위해 환경들을 설정하는 단계라고 볼 수 있다. 환경 설정을 MBeanServer 에 연결하기 전에 할 수 있다. 처리하려는 MBean 을 질의하고, 그 결과를 받아서 처리할 수 있다.

Step 2 에서는 MBeanServer 로 접속하기 위한 방법은 앞서 말한 세 가지 메소드가 있다. jeus.management.RemoteMBeanServerFactory 는 JEUS 유틸리티를 사용하는 경우 사용되고, 이 외에 JNDI 에 등록되어 있는 reference 를 사용하거나 JMX Remote API 의 javax.management. remote.JMXServiceURL 을 사용하여 javax.management.remote.JMXConnector 을 얻을 수 있다. 자세한 내용은 다음 장에서 설명한다.

4.3 JEUS 유틸리티 사용

이번 절에서는 JEUS 유틸리티를 사용해서 JEUS 를 모니터링하는 JMX 어플리케이션에 대해서 설명한다. Step 2 부분이 다른 방법과 차이가 있다.

<<*JMXClientSampleJEUS.java*>>

```
// java class import
import java.util.Set;
import java.util.Iterator;
import java.util.Hashtable;
// jmx class import
import javax.management.MBeanServerConnection;
import javax.management.ObjectName;
import javax.naming.Context;
// jeus class import
import jeus.management.RemoteMBeanServerFactory;

public class JMXClientSampleJEUS {

    public static void main(String args[]) {
        try {
            // Step 1. Setting Environments
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY,
                "jeus.jndi.JEUSContextFactory");
            env.put(Context.SECURITY_PRINCIPAL,
                "administrator");
            env.put(Context.SECURITY_CREDENTIALS,
```

```
                "jeusadmin");

            // Step 2. Getting MBeanServerConnection
            MBeanServerConnection jeusMonitorMbeanServer
                = RemoteMBeanServerFactory.getMBeanServer(env);

            // Step 3. Query
            ObjectName jeusScope = new ObjectName
                (RemoteMBeanServerFactory.SYSTEM_DOMAIN
                 + ":*");
            Set jeusSet = jeusMonitorMbeanServer
                .queryNames(jeusScope, null);

            // Step 4. Handling the Query Result
            for(Iterator i = jeusSet.iterator();i.hasNext();) {
                System.out.println(
                    "[JEUS Monitoring Service] " + i.next());
            }
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }
}
```

Step 2 에서 JEUS 가 제공하는 JEUS 유틸리티 클래스인 `jeus.management.RemoteMBeanServerFactory` 를 사용했다. 이 API 를 사용하면 간단하게 `MBeanServerConnection` 을 얻을 수 있다. 이 API 는 다음 표와 같다.

표 1. RemoteMBeanServeFactory API

속성/메소드	*설명*
static field **SYSTEM_DOMAIN**	MBean 이 생성될 때 사용되는 ObjectName 의 접두어이다. JEUS 에서는 JEUS 의 system MBean 들이 모두 ”JEUS”를 사용한다.

4.4 JNDI 사용

이번 절에서는 JNDI 를 사용해서 JEUS 를 모니터링하는 JMX 어플리케이션에 대해서 설명한다. Step 2 부분이 다른 방법과 차이가 있다.

<<*JMXClientSampleJNDI.java*>>

```
// java class import
import java.util.Set;
import java.util.Iterator;
import java.util.Hashtable;
// jmx class import
import javax.management.MBeanServerConnection;
import javax.management.ObjectName;
import javax.management.remote.rmi.RMIServer;
import javax.management.remote.rmi.RMIConnector;
import javax.naming.Context;
import javax.naming.InitialContext;
// jeus class import
import jeus.management.RemoteMBeanServerFactory;
import jeus.management.JMXConstants;

public class JMXClientSampleJNDI {

    public static void main(String args[]) {
        try {
            // Step 1. Setting Environments
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY,
                "jeus.jndi.JEUSContextFactory");
            env.put(Context.SECURITY_PRINCIPAL,
                "administrator");
            env.put(Context.SECURITY_CREDENTIALS,
                "jeusadmin");

            // Step 2. Getting MBeanServerConnection
            InitialContext ctx = new InitialContext(env);
            // targetName could be node or container name,
            // for example, "johan", "johan_container1"
            String targetName = args[0];
```

```
            JMXConnector connector = (JMXConnector)ctx.lookup(
                JMXConstants.JNDI_BINDING_PREFIX
                + targetName);
            connector.connect();
            MBeanServerConnection rmiMbeanServer
                = connector.getMBeanServerConnection();

            // Step 3. Query
            ObjectName jeusScope = new ObjectName
                (RemoteMBeanServerFactory.SYSTEM_DOMAIN
                + ":*");
            Set jeusSet = rmiMbeanServer
                .queryNames(jeusScope, null);

            // Step 4. Handling the Query Result
            for(Iterator i = jeusSet.iterator();i.hasNext();) {
                System.out.println(
                    "[JEUS Monitoring Service] " + i.next());
            }
        } catch (Throwable t) {
          t.printStackTrace();
        }
    }
}
```

JNDI 에서 lookup 할 때에는 등록되어 있는 Connector 가 RMI Connector 든 JMXMP Connector 든 관계없이 같은 방식으로 사용이 가능하다. lookup 할 때 사용하는 export name 은 여기서는 JEUS 에서 사용하는 기본 naming 방식을 사용했다. 이 방식은 mgmt/rmbs/context 아래에 Manager JVM 이나 default Engine Container JVM 의 MBeanServer 는 <node name> (ex. johan), 그외 별도로 띄워진 Engine Container 의 경우는 <node name>_<container_name> (ex. johan_container1) 의 이름이다. 그 외에 RMI Connector 의 경우에는 JMXMP Connector 가 별도로 띄워져 있는 경우에 기본 방식 이외의 이름을 reference export name 으로 지정할 수 있다. (JEUSMain.xsd 의 Schema reference 부록 참조)

참조: JNDI 의 자세한 정보에 대해서는 JEUS 서버 안내서를 참조하기 바란다. 만약에 JMX 어플리케이션이 Servlet 또는 EJB 아래에서 실행된다면 JNDI 파라미터에 대한 설정은 필요하지 않다.

Security 설정에 대한 자세한 설명은 4.6절을 참고하기 바란다.

4.5 JMX Remote API 사용

이 절에서는 JMX Remote API 를 사용해서 JEUS 를 모니터링하는 JMX 어플리케이션에 대해서 설명한다. Step 2 부분이 다른 방법과 차이가 있다.

<<*JMXClientSampleRemoteAPI.java*>>

```
// java class import
import java.util.Set;
import java.util.Iterator;
import java.util.Hashtable;
// jmx class import
import javax.management.MBeanServerConnection;
import javax.management.ObjectName;
import javax.management.remote.JMXServiceURL;
import javax.management.remote.JMXConnector;
import javax.management.remote.JMXConnectorFactory;
import javax.naming.Context;
// jeus class import
import jeus.management.RemoteMBeanServerFactory;
import jeus.management.JMXConstants;

public class JMXClientSampleRemoteAPI {

    public static void main(String args[]) {
        try {
            // Step 1. Setting Environments
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY,
                "jeus.jndi.JEUSContextFactory");
            env.put(Context.SECURITY_PRINCIPAL,
                "administrator");
            env.put(Context.SECURITY_CREDENTIALS,
                "jeusadmin");

            // Step 2. Getting MBeanServerConnection
            // targetName could be node or container name,
            // for example, "johan", "johan_container1"
```

```
            String targetName = args[0];
            JMXServiceURL url =
            new JMXServiceURL("service:jmx:rmi:///jndi/"
                + JMXConstants.JNDI_BINDING_PREFIX
                + targetName);
            JMXConnector jmxconnector
                = JMXConnectorFactory.newJMXConnector(url,env);
            jmxconnector.connect();
            MBeanServerConnection jmxMbeanServer
                = jmxconnector.getMBeanServerConnection();

            // Step 3. Query
            ObjectName jeusScope = new ObjectName
                (RemoteMBeanServerFactory.SYSTEM_DOMAIN
                + ":*");
            Set jeusSet = jmxMbeanServer
                .queryNames(jeusScope, null);

            // Step 4. Handling the Query Result
            for(Iterator i = jeusSet.iterator();i.hasNext();) {
                System.out.println(
                    "[JEUS Monitoring Service] " + i.next());
            }
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }
}
```

위의 경우는 등록되어 있는 Connector 가 RMI Connector 인 경우의 URL 을 사용했다. JEUS 에서 RMI Connector 를 위해 기본적으로 사용하는 JMXServiceURL 의 URL path 는 /jndi/mgmt/rmi/ 아래에 Manager JVM 이나 default Engine Container JVM 의 MBeanServer 는 <node name> (ex. johan), 그외 별도로 띄워진 Engine Container 의 경우는 <node name>_<container_name> (ex. johan_container1) 의 이름이다. 이 URL path 는 RMI Connector 설정에서 <export-name> 설정으로 바꿀 수 있다. (JEUSMain.xsd 의 Schema reference 부록 참조)

JMXMP Connector 의 경우에는 JMXServiceURL 의 port 는 해당 Engine Container 나 Manager 가 사용하는 base port (ex. 9736)을 사용한다. 이는

JEUS 의 통합 listen port 로 접속하기 위함이다. 대신 URL path 에 Manager JVM 이나 default Engine Container JVM 의 MBeanServer 는 <node name> (ex. johan), 그외 별도로 띄워진 Engine Container 의 경우는 <node name>_<container_name> (ex. johan_container1) 의 이름을 사용하여 실제로 사용하고자 하는 MBeanServer 를 지정한다. 또한 이 경우에는 JEUS 의 확장된 profile message 를 사용해야 하기 때문에 JMX Remote API 의 environment 설정에 따라 다음과 같은 설정을 추가해야 한다.

```
t.put("jmx.remote.profiles", "JEUS/PLAIN");
t.put(“jmx.remote.profile.provider.pkgs”,
    "jeus.management.remote.profile");
```

참조: JNDI 의 자세한 정보에 대해서는 JEUS 서버 안내서를 참조하기 바란다. 만약에 JMX 어플리케이션이 Servlet 또는 EJB 아래에서 실행된다면 JNDI 파라미터에 대한 설정은 필요하지 않다.

Security 설정에 대한 자세한 설명은 4.6절을 참고하기 바란다.

4.6 Security 설정

이 절에서는 JEUS 모니터링 서비스를 위한 보안 설정에 대해서 설명한다. 기본적으로 Security 파일인 subjects.xml 과 policies.xml 에서 사용자에게 권한을 줄 수 있고, 권한을 가진 사용자들만이 JMX 클라이언트 어플리케이션에 접근할 수 있다.

<<subjects.xml>>

```
...
<subject>
    <description>No description</description>
    <principal>
          <classname>jeus.security.resource.PrincipalImpl
          </classname>
          <name>jeus</name>
    </principal>
    <credential>
          <classname>jeus.security.resource.PasswordFactory
          </classname>
          <property>
                <name>password</name>
```

```
                <value>amV1cw==</value>
            </property>
    </credential>
</subject>
...
```

<<policies.xml>>

```
...
<policy>
    <role-permissions>
          <role-permission>
                <principal>jeus</principal>
                <role>SecurityAdministrator</role>
                <classname>jeus.security.resource.RolePermission
                </classname>
           </role-permission>
    </role-permissions>
    <resource-permissions>
          <context-id>default</context-id>
          <resource-permission>
                <role> SecurityAdministrator </role>
                <resource>jeus.*</resource>
                <actions>*</actions>
                <classname>jeus.security.resource.Resource
                 Permission </classname>
          </resource-permission>
    </resource-permissions>
</policy>
...
```

다음 예제 Step 1 에서 이에 대한 코드를 볼 수 있다.

<<JMXClientSampleSecurity.java>>

```
// java class import
import java.util.Set;
import java.util.Iterator;
import java.util.Hashtable;
// jmx class import
import javax.management.MBeanServerConnection;
```

```
import javax.management.ObjectName;
import javax.naming.Context;
// jeus class import
import jeus.management.RemoteMBeanServerFactory;

public class JMXClientSampleJEUS {

    public static void main(String args[]) {
        try {
            // Step 1. Setting Environments
            Hashtable env = new Hashtable();
            env.put(Context.INITIAL_CONTEXT_FACTORY,
                "jeus.jndi.JEUSContextFactory");
            env.put(Context.SECURITY_PRINCIPAL,
                "jeus");
            env.put(Context.SECURITY_CREDENTIALS,
                "jeus");

            // Step 2. Getting MBeanServerConnection
            MBeanServerConnection jeusMonitorMbeanServer
                = RemoteMBeanServerFactory.getMBeanServer(env);

            // Step 3. Query
            ObjectName jeusScope = new ObjectName
                (RemoteMBeanServerFactory.SYSTEM_DOMAIN
                + ":*");
            Set jeusSet = jeusMonitorMbeanServer
                .queryNames(jeusScope, null);

            // Step 4. Handling the Query Result
            for(Iterator i = jeusSet.iterator();i.hasNext();) {
                System.out.println(
                    "[JEUS Monitoring Service] " + i.next());
            }
        } catch (Throwable t) {
            t.printStackTrace();
        }
    }
}
```

이상으로 JMX 어플리케이션의 보안 설정에 대해서 알아보았다. 다음 절에서는 JEUS MBean 의 표준 JMX Object Name 구조에 대해서 알아본다.

4.7 ObjectName

"ObjectName"은 MBean 객체의 기본 JMX 객체 이름이다. "ObjectName"의 포멧은 JMX 표준 포맷이지만 특정 벤더에 따라 값이 달라질 수 있다. 여기서 다른 값들을 가질 수 있는 이유는 JMX 구현이 각각 다르기 때문이다. JMX MBean 들의 계층 구조는 벤더마다 다르고, MBean 들도 벤더에 따라 여러 가지가 있다.

JEUS ObjectName 의 문법은 다음과 같다:

<domain_name>: j2eeType=<j2eeType_value>, name=<name_value>, [<parent-j2eeType_value>], [jeusType = <jeusType_value>], [isTargetable = <isTargetable_value>], [jeusManager = <jeusManager_value>], JMXManager = <JMXManager_value> [,*]

또는

<domain_name>: *

ObjectName 은 "domain_name"으로 시작 해야 하고, 각각의 이름과 값의 짝이 순차적으로 규정되지 않는다. 예를 들어 "JEUS: j2eeType=JeusService, jeusType=JEUSManager, *" 과 "JEUS: jeusType=JeusManager, j2eeType=JeusService, *" 는 둘 다 JEUSManager MBean 의 objectname 을 얻어 온다.

표 2 . ObjectName 의 요소

이름	*설명*	*값*
domain_name	JEUS Domain name	JEUS
j2eeType	MBean 은 J2EE Type 이며, J2EE Management 스펙에	다음 값들 중 하나이다. "JeusService", "J2EEDomain", "J2EEServer", "JVM", "EJBModule",

	의해 기술 된다.	“MessageDrivenBean”, “EntityBean”, “StatefulSessionBean”, “StatelessSessionBean”, "WebModule", "Servlet", "JDBCResource", "JDBCDataSource", “JDBCDriver", "JMSResource", "JTAResource", "JMSConnectionResource", "JMSConsumerResource", "JMSProducerResource".
name	MBean 의 이름. 각각의 MBean Object 에는 유일한 값이 있다.	예를 들어 “johan_container1”이라는 Container 가 실행하는 JVM 의 이름은 “johan_container1”이다
parent-j2eeType	MBean 의 상위 j2ee 타입으로, 각 MBean 들에 계층이 규정되어 있다.	예를 들면, “JDBCDriver”의 상위 j2ee 타입은 “JDBCDataSource”이다.
jeusType	JEUS JMX 에서 정의된 MBean 들의 타입이다. “JeusService” j2eeType 만 몇 가지 jeusType 을 가질 수 있다	다음 값들 중 하나이다. "JeusManager", "JMSConnectionResource", "JMSConsumerResource", "JMSProducerResource", "JMSSessionResource", "EJBEngine", "ContextGroup", "DBConnectionPool", "ThreadPool_WEBC", "WebEngine", "WebListener", "JMSClientResource", "JMSQueueConnectionFactoryResource", "JMSTopicConnectionFactoryResource", "JMSQueueDestinationResource", "JMSTopicDestinationResource", "JMSDurableSubscriberResource".

isTargetable	사용자 AP(EJB, Servlet, JSP)가 올라가 동작하는 MBean 에서는 반드시 true 로 설정되어야 한다.	“true” 또는 “false”

5 결론

이상으로 JMX 1.2 과 JMX Remote API 1.0 호환 모니터링 어플리케이션 개발에 대해서 알아보았다. JMX 에 대한 소개와 JEUS JMX 를 설정하는 방법, 그리고 JEUS JMX 클라이언트 개발에 대해서도 알아 보았다.

JMX 는 JMX Management 스펙을 만족하면서 JEUS JMX API 와 같은 고유의 서비스도 제공한다. JEUS 에 통합되어 있으므로, JEUS JMX 를 엔터프라이즈 모니터링 어플리케이션 개발에 사용할 수 있다.

A 모니터링 정보 레퍼런스

A.1 소개

이 부록에서는 JEUS JMX 의 javadoc 을 참조하는 방법과 JEUS JMX 에 대한 API 를 참조하는 방법에 대해서 살펴본다. API 를 사용하는 방법은 4 장에 설명이 되어 있다.

A.2 JEUS JMX package 의 javadoc

Javadoc 은 "%JEUS_HOME%\docs\jmxdoc" 디렉토리에 위치한다.

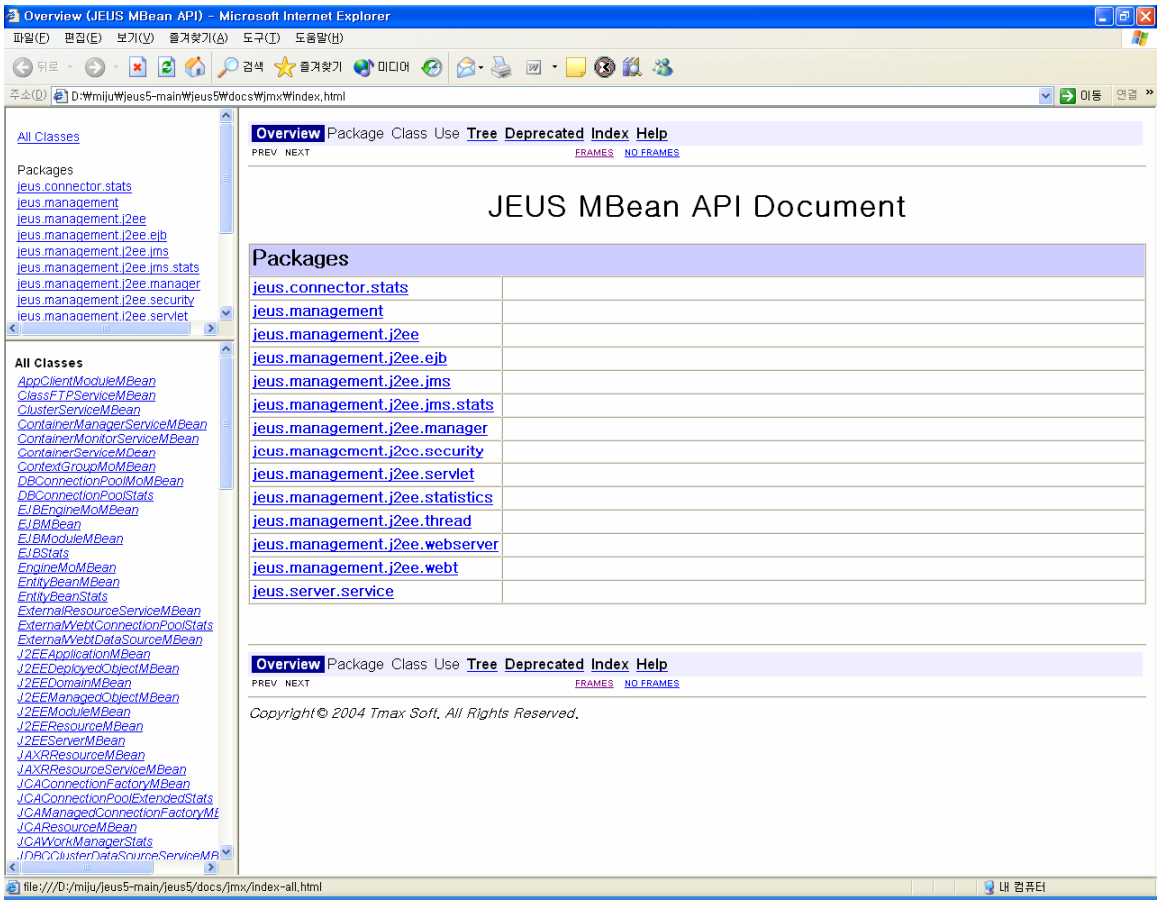

그림 1.JEUS JMX package 의 javadoc.

B JEUSMain.xml XML 설정 레퍼런스

B.1 소개

본 부록의 레퍼런스는 JEUS 의 주 설정 파일인 JEUSMain.xml 중에서 JMX 관련 태그에 대해서 설명하고 있다. 이 파일의 xsd 파일은 “JEUS_HOME\config\xsds” 디렉토리의 “jeus-main.xsd” 파일이다

본 레퍼런스는 3 부분으로 나뉘어져 있다

1. **XML Schema/XML 트리**: XML 설정 파일의 모든 태그 리스트를 정리했다. 각 노드의 형식은 다음과 갈다.
    a. 태그 레퍼런스로 빨리 찾아보기 위해서 각 태그마다 인덱스 번호(예: (11))를 붙여놓았다. 태그 레퍼런스 에서는 이 번호 순서로 설명한다.
    b. XML Schema 에서 정의한 XML 태그명을 <**tag name**> 형식으로 표시한다.
    c. XML Schema 에서 정의한 Cardinality 를 표시한다. “?” = 0 개나 1 개의 element, “+” = 1 개 이상의 element, “*” = 0 개 이상의 element, (기호가 없음) = 정확히 1 개의 element.
    d. 몇몇 태그에는 “P” 문자를 붙여 놓았는데, 해당 태그는 성능에 관계되는 태그라는 것을 뜻한다. 이 태그는 설정을 튜닝할 때 사용된다.
2. **태그 레퍼런스**: 트리에 있는 각 XML 태그를 설명한다. 이런 테이블은 다음 하위-항목들을 포함하고 있다.
    a. **Description**: 태그에 대한 간단한 설명
    b. **Value Description**: 입력하는 값과 타입
    c. **Value Type**: 값의 데이터 타입. 예) String

d. **Default Value**: 해당 XML 을 사용하지 않았을 때 기본적으로 사용되는 값

e. **Defined values**: 이미 정해져 있는 값

f. **Example**: 해당 XML 태그에 대한 예

g. **Performance Recommendation**: 성능 향상을 위해서 추천하는 값

h. **Child Elements**: 자신의 태그 안에 사용하는 태그

3. **샘플 XML 파일**: “JEUSMain.xml”에 대한 완전한 예제

B.2 XML Schema/XML Tree

```
(217) <jmx-manager>?
      (218) <jmx-connector>?
                  (219) <jmxmp-connector>?
                  (220) <jmxmp-connector-port>? P
            (221) <rmi-connector>?
                  (222) <rmi-connector-port>?
                  (223) <export-name>?
                  (224) <ref-export-name>?
      (225) <html-adaptor-port>?
      (226) <snmp-adaptor>?
            (227) <snmp-adaptor-port>
            (228) <snmp-version>? P
            (229) <snmp-max-packet-size>? P
            (230) <snmp-security>? P
            (231) <trap-demon>*
                  (232) <ip-address>
                  (233) <port>
      (234) <pooling>?
                  (235) <min>? P
                  (236) <max>? P
                  (237) <period>? P
      (238) <mlet-url>*
```

B.3 Element Reference

EngineContainer 단 설정

```
(217) <jeus-system> <node> <engine-container> <jmx-manager>
```

Description JMX Manager element 는 이 Engine Container 의 JMX 관련 모든 설정을 담고 있다.

Child Elements
```
(218)jmx-connector?
(225)html-adaptor-port?
(226)snmp-adaptor?
(238)mlet-url*
```

```
(218) <jeus-system> <node> <engine-container> <jmx-manager> <jmx-connector>
```

Description 다른 process 에서 이 Engine Container 의 JMX 를 access 할 때 사용하는 JMX Connector 를 설정한다. 기본적으로는 JEUSMP Connector 를 사용한다.

Child Elements
```
(219)jmxmp-connector?
(221)rmi-connector?
```

```
(219) <jeus-system> <node> <engine-container> <jmx-manager> <jmx-connector>
<jmxmp-connector>
```

Description JMX Connector 로 JMXMP Connector 를 사용한다.

Child Elements
```
(220)jmxmp-connector-port?
```

```
(220) <jeus-system> <node> <engine-container> <jmx-manager> <jmx-connector>
<jmxmp-connector> <jmxmp-connector-port>
```

Description 다른 process 에서 이 Engine Container 의 JMX 를 access 할 때 사용하는 JEUSMP Connector 의 listen port 를 지정한다. 만약 이 값이 0 이거나 지정하지 않으면 JEUSMP Connector 가 사용하는 listen port 를 따로 만들지 않고 jeus 의 공통 port 를 사용한다. 만약 JEUS 의 JMX RemoteAPI 를 사용하지 않고 다른 Runtime 에서 JMXMP protocol 로 접근하고자 한다면 이를 0 이 아닌 다른 값으로 지정해야 한다.

Value Type nonNegativeIntType

Value Type Description 0 이상의 int type 이다. 즉, int 범위에서 0 이상의 값들을 포함한다.

Default Value 0

Defined Value 0
jeus 의 공통 port 를 사용한다.

```
(221) <jeus-system> <node> <engine-container> <jmx-manager> <jmx-connector>
<rmi-connector>
```

Description JMX Connector 로 RMI Connector 를 사용한다. 만약 jmxmp-connector 와 같이 설정되어 있는 경우에는 JEUS system 내부적으로는 jmxmp-connector 를 기본적으로 사용하게 된다. 또한 이 경우에는 rmi-connector 의 ref-export-name 이 별도로 설정되어 있어야 한다. 이 이름이 JEUS 에서 기본적으로 사용하는 이름과 같거나 설정이 되어있지 않다면 exception 이 발생한다.

Child Elements
```
(222)rmi-connector-port?
(223)export-name?
(224)ref-export-name?
```

(222) <jeus-system> <node> <engine-container> <jmx-manager> <jmx-connector> <rmi-connector> **<rmi-connector-port>**

Description 다른 process 에서 이 Engine Container 의 JMX 를 access 할 때 사용하는 RMI Connector 의 port 를 지정한다.

Value Type nonNegativeIntType

Value Type Description 0 이상의 int type 이다. 즉, int 범위에서 0 이상의 값들을 포함한다.

Defined Value 0
random 하게 port 를 지정한다.

(223) <jeus-system> <node> <engine-container> <jmx-manager> <jmx-connector> <rmi-connector> **<export-name>**

Description JEUS 의 기본 export name 대신에 다른 export name 을 사용할때 설정한다. 이 export name 은 JMXServiceURL 의 URL path 에 들어가게 된다.

Value Type token

(224) <jeus-system> <node> <engine-container> <jmx-manager> <jmx-connector> <rmi-connector> **<ref-export-name>**

Description 이 connector 를 얻을수 있는 jndi name 을 JEUS 의 기본 jndi name 이 아닌 다른 name 으로 지정하고자 할 때 사용한다. 이 export name 으로 lookup 하면 JMXConnector 객체를 얻을수 있다.

Value Type token

(225) <jeus-system> <node> <engine-container> <jmx-manager> **<html-adaptor-port>**

Description JMX 의 adapter 중 하나인 HTML adapter 의 port 를 지정한다. 여기에 지정된 값으로 Web Browser 가 접속하게 된다.

Value Type off-intType

Value Type Description 기본적으로 non-negative int 형이지만 -1 인 경우에는 설정을 하지 않은게 된다.즉, off 된다.

(226) <jeus-system> <node> <engine-container> <jmx-manager> **<snmp-adaptor>**

Description JMX 의 adapter 중 하나인 SNMP adapter 를 설정한다.

Child Elements
(227)snmp-adaptor-port
(228)snmp-version?
(229)snmp-max-packet-size?
(230)snmp-security?
(231)trap-demon*
(234)pooling?

(227) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor> **<snmp-adaptor-port>**

Description SNMP 어댑터의 리스너 포트

Value Type snmp-adaptor-portType

(228) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor> **<snmp-version>**

Description SNMP 버전을 지정하며 1, 2 또는 3 을 지정할 수 있다

Value Type snmp-versionType

Default Value	3
Defined Value	1 2 3

(229) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor> **<snmp-max-packet-size>**

Description	SNMP 패킷에 대한 최대값을 설정하며 최소 256 바이트부터 설정 할 수 있다.
Value Type	snmp-max-packet-sizeType
Default Value	4096

(230) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor> **<snmp-security>**

Description	보안을 적용시킬 것 인지를 설정한다. 보안은 SNMP 버전 3 에서만 지정이 가능 하다.
Value Type	boolean
Default Value	false

(231) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor> **<trap-demon>**

Description	장애 상황 발생시 TRAP 메세지를 보낼 서버를 설정한다. 여러 개 설정이 가능하며 설정된 모든 ip, address 로 메세지를 보낸다.
Child Elements	(232)ip-address (233)port

(232) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor> <trap-demon> **<ip-address>**

Description	Demon 의 IP address
Value Description	a valid IP address
Value Type	token
Example	<host-name>111.111.111.1</host-name>

(233) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor> <trap-demon> **<port>**

Description	Demon 의 port
Value Description	a port number
Value Type	int
Example	<port>8888</port>

(234) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor> **<pooling>**

Description	SNMP Server 에서 요청을 처리하는 쓰레드로 구성되어 있다. 아래 element 는 이 쓰레드를 관리하는 pool 을 설정한다.

Child Elements

```
(235)min?
(236)max?
(237)period?
```

```
(235) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor>
<pooling> <min>
```

Description pooling 되는 객체의 최소값을 지정한다.

Value Type nonNegativeIntType

Value Type Description 0 이상의 int type 이다. 즉, int 범위에서 0 이상의 값들을 포함한다.

Default Value 2

```
(236) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor>
<pooling> <max>
```

Description pooling 되는 객체의 최대값을 지정한다.

Value Type nonNegativeIntType

Value Type Description 0 이상의 int type 이다. 즉, int 범위에서 0 이상의 값들을 포함한다.

Default Value 30

```
(237) <jeus-system> <node> <engine-container> <jmx-manager> <snmp-adaptor>
<pooling> <period>
```

Description pooling 되는 객체를 정리하는 시간을 지정한다.

Value Type long

Default Value 3600000

Performance Recommendation 이 값이 클수록 정리하는 주기가 길어져 server 운영에는 부하가 적게 가해지나 그만큼 메모리가 누수될 수 있으므로 적당한 값으로 지정한다.

```
(238) <jeus-system> <node> <engine-container> <jmx-manager> <mlet-url>
```

Description 이 Engine Container 의 MBeanServer 에 등록할 MLet 의 URL 을 지정한다.

Value Type token

Node 단 설정

```
(374) <jeus-system> <node> <jmx-manager>
```

Description JEUS Manager JVM 에서 사용하는 JMX 에 대한 설정이다.

Child Elements

```
(375)jmx-connector?
(382)html-adaptor-port?
(383)snmp-adaptor?
(395)mlet-url*
```

```
(375) <jeus-system> <node> <jmx-manager> <jmx-connector>
```

Description 다른 process 에서 이 Engine Container 의 JMX 를 access 할 때 사용하는 JMX Connector 를 설정한다. 기본적으로는 JEUSMP Connector 를 사용한다.

Child Elements (376)jmxmp-connector?
(378)rmi-connector?

(376) <jeus-system> <node> <jmx-manager> <jmx-connector> **<jmxmp-connector>**

Description JMX Connector 로 JMXMP Connector 를 사용한다.

Child Elements (377)jmxmp-connector-port?

(377) <jeus-system> <node> <jmx-manager> <jmx-connector> <jmxmp-connector> **<jmxmp-connector-port>**

Description 다른 process 에서 이 Engine Container 의 JMX 를 access 할 때 사용하는 JEUSMP Connector 의 listen port 를 지정한다. 만약 이 값이 0 이거나 지정하지 않으면 JEUSMP Connector 가 사용하는 listen port 를 따로 만들지 않고 jeus 의 공통 port 를 사용한다. 만약 JEUS 의 JMX RemoteAPI 를 사용하지 않고 다른 Runtime 에서 JMXMP protocol 로 접근하고자 한다면 이를 0 이 아닌 다른 값으로 지정해야 한다.

Value Type nonNegativeIntType

Value Type Description 0 이상의 int type 이다. 즉, int 범위에서 0 이상의 값들을 포함한다.

Default Value 0

Defined Value 0
jeus 의 공통 port 를 사용한다.

(378) <jeus-system> <node> <jmx-manager> <jmx-connector> **<rmi-connector>**

Description JMX Connector 로 RMI Connector 를 사용한다. 만약 jmxmp-connector 와 같이 설정되어 있는 경우에는 JEUS system 내부적으로는 jmxmp-connector 를 기본적으로 사용하게 된다. 또한 이 경우에는 rmi-connector 의 ref-export-name 이 별도로 설정되어 있어야 한다. 이 이름이 JEUS 에서 기본적으로 사용하는 이름과 같거나 설정이 되어있지 않다면 exception 이 발생한다.

Child Elements (379)rmi-connector-port?
(380)export-name?
(381)ref-export-name?

(379) <jeus-system> <node> <jmx-manager> <jmx-connector> <rmi-connector> **<rmi-connector-port>**

Description 다른 process 에서 이 Engine Container 의 JMX 를 access 할 때 사용하는 RMI Connector 의 port 를 지정한다.

Value Type nonNegativeIntType

Value Type Description 0 이상의 int type 이다. 즉, int 범위에서 0 이상의 값들을 포함한다.

Defined Value 0
random 하게 port 를 지정한다.

(380) <jeus-system> <node> <jmx-manager> <jmx-connector> <rmi-connector> **<export-name>**

Description JEUS 의 기본 export name 대신에 다른 export name 을 사용할때 설정한다. 이 export name 은 JMXServiceURL 의 URL path 에 들어가게 된다.

Value Type token

(381) <jeus-system> <node> <jmx-manager> <jmx-connector> <rmi-connector> **<ref-export-name>**

Description 이 connector 를 얻을수 있는 jndi name 을 JEUS 의 기본 jndi name 이 아닌 다른 name 으로 지정하고자 할 때 사용한다. 이 export name 으로 lookup 하면 JMXConnector 객체를 얻을수 있다.

Value Type token

(382) <jeus-system> <node> <jmx-manager> **<html-adaptor-port>**

Description JMX 의 adapter 중 하나인 HTML adapter 의 port 를 지정한다. 여기에 지정된 값으로 Web Browser 가 접속하게 된다.

Value Type off-intType

Value Type Description 기본적으로 non-negative int 형이지만 -1 인 경우에는 설정을 하지 않은게 된다.즉, off 된다.

(383) <jeus-system> <node> <jmx-manager> **<snmp-adaptor>**

Description JMX 의 adapter 중 하나인 SNMP adapter 를 설정한다.

Child Elements
```
(384)snmp-adaptor-port
(385)snmp-version?
(386)snmp-max-packet-size?
(387)snmp-security?
(388)trap-demon*
(391)pooling?
```

(384) <jeus-system> <node> <jmx-manager> <snmp-adaptor> **<snmp-adaptor-port>**

Description SNMP 어댑터의 리스너 포트

Value Type snmp-adaptor-portType

(385) <jeus-system> <node> <jmx-manager> <snmp-adaptor> **<snmp-version>**

Description SNMP 버전을 지정하며 1, 2 또는 3 을 지정할 수 있다

Value Type snmp-versionType

Default Value 3

Defined Value 1

2

3

(386) <jeus-system> <node> <jmx-manager> <snmp-adaptor> **<snmp-max-packet-size>**

Description SNMP 패킷에 대한 최대값을 설정하며 최소 256 바이트부터 설정 할 수 있다.

Value Type snmp-max-packet-sizeType

Default Value 4096

(387) <jeus-system> <node> <jmx-manager> <snmp-adaptor> **<snmp-security>**

Description 보안을 적용시킬 것 인지를 설정한다. 보안은 SNMP 버전 3 에서만 지정이 가능 하다.

Value Type boolean

Default Value	false

(388) <jeus-system> <node> <jmx-manager> <snmp-adaptor> **<trap-demon>**

Description	장애 상황 발생시 TRAP 메세지를 보낼 서버를 설정한다. 여러 개 설정이 가능하며 설정된 모든 ip, address 로 메세지를 보낸다.
Child Elements	(389)ip-address (390)port

(389) <jeus-system> <node> <jmx-manager> <snmp-adaptor> <trap-demon> **<ip-address>**

Description	Demon 의 IP address
Value Description	a valid IP address
Value Type	token
Example	<host-name>111.111.111.1</host-name>

(390) <jeus-system> <node> <jmx-manager> <snmp-adaptor> <trap-demon> **<port>**

Description	Demon 의 port
Value Description	a port number
Value Type	int
Example	<port>8888</port>

(391) <jeus-system> <node> <jmx-manager> <snmp-adaptor> **<pooling>**

Description	SNMP Server 에서 요청을 처리하는 쓰레드로 구성되어 있다. 아래 element 는 이 쓰레드를 관리하는 pool 을 설정한다.
Child Elements	(392)min? (393)max? (394)period?

(392) <jeus-system> <node> <jmx-manager> <snmp-adaptor> <pooling> **<min>**

Description	pooling 되는 객체의 최소값을 지정한다.
Value Type	nonNegativeIntType
Value Type Description	0 이상의 int type 이다. 즉, int 범위에서 0 이상의 값들을 포함한다.
Default Value	2

(393) <jeus-system> <node> <jmx-manager> <snmp-adaptor> <pooling> **<max>**

Description	pooling 되는 객체의 최대값을 지정한다.
Value Type	nonNegativeIntType
Value Type Description	0 이상의 int type 이다. 즉, int 범위에서 0 이상의 값들을 포함한다.
Default Value	30

(394) <jeus-system> <node> <jmx-manager> <snmp-adaptor> <pooling> **<period>**

Description	pooling 되는 객체를 정리하는 시간을 지정한다.
Value Type	long

Default Value	3600000
Performance Recommendation	이 값이 클수록 정리하는 주기가 길어져 server 운영에는 부하가 적게 가해지나 그만큼 메모리가 누수될 수 있으므로 적당한 값으로 지정한다.

(395) <jeus-system> <node> <jmx-manager> **<mlet-url>**

Description	이 Engine Container 의 MBeanServer 에 등록할 MLet 의 URL 을 지정한다.
Value Type	token

B.4 JEUSMain.xml 샘플 파일

<<JEUSMain.xml>>

```
<?xml version="1.0" encoding="utf-8"?>
<jeus-system xmlns="http://www.tmaxsoft.com/xml/ns/jeus">
    <node>
        <name>sky</name>
        <class-ftp>true</class-ftp>
        <sequential-start>true</sequential-start>
        <enable-webadmin>true</enable-webadmin>
        <engine-container>
            <name>container1</name>
            <sequential-start>true</sequential-start>
            <engine-command>
                <type>ws</type>
                <name>engine1</name>
            </engine-command>
            <engine-command>
                <type>jms</type>
                <name>engine1</name>
            </engine-command>
            <engine-command>
                <type>ejb</type>
                <name>engine1</name>
            </engine-command>
            <engine-command>
                <type>servlet</type>
                <name>engine1</name>
            </engine-command>
```

```
        </engine-container>

        <jmx-manager>
            <jmx-connector>
                <jmxmp-connector>
                    <jmxmp-connector-port>5001</jmxmp-connector-
port>
                </jmxmp-connector>
            </jmx-connector>
            <html-adaptor-port>-1</html-adaptor-port>
            <snmp-adaptor>
                <snmp-adaptor-port>9999</snmp-adaptor-port>
                <snmp-version>3</snmp-version>
                <snmp-max-packet-size>4096</snmp-max-packet-size>
                <snmp-security>true</snmp-security>
                <pooling>
                    <min>5</min>
                    <max>20</max>
                    <period>5</period>
                </pooling>
            </snmp-adaptor>
        </jmx-manager>
    </node>
</jeus-system>
```

색인