# 신놀이 메시지 기술 명세

0 목차



1 신놀이 메시지의 논리 구조

- 정규식 표현 -

메시지 = 메시지 식별자, 항목 그룹

항목 그룹 = (항목)*

항목 = (단일 항목 | 배열)

단일 항목 = 이름, 타입, 타입 부가 정보{0..1}, 값

타입 부가 정보 = 크기 | 문자셋

배열 = 이름, 반복 횟수, (항목 그룹)*

# 2 메시지 구현

## 2.1 메시지 운영 시나리오

### 2.1.1 메시지 정보는 XML 파일로 작성한다.

### 2.1.2 이 메시지 정보 파일을 읽어와 메시지 정보를 구축한다.

### 2.1.3 메시지 정보를 바탕으로 메시지 내용을 구성하게 한다.

### 2.1.4 네트워크 상에서 메시지 교환은 이 메시지 정보를 바탕으로 이루어 진다.

### 2.1.5 신규 메시지 정보 추가나 수정 시 XML 로 작성된 메시지 정보 파일을 읽어 와서 교체를 하면 그 즉시로 적용된다.

## 2.2 항목 이름 규칙

### 2.2.1 XML 문서 이름 규칙과 동일

XML Tag 혹은 attribute 이름과 동일하게 하여 XML 문서로 표현될 수 있도록 한다. 이는 나중 자바 변수로도 그 변환가 쉬울 것이다.

참고 주소 : http://www.w3.org/TR/xml/#NT-Name

```
[4]     NameStartChar          ::=    ":" | [A-Z] | "_" | [a-z] | [#xC0-#xD6] |
            [#xD8-#xF6] | [#xF8-#x2FF] | [#x370-#x37D] | [#x37F-#x1FFF] |
            [#x200C-#x200D] | [#x2070-#x218F] | [#x2C00-#x2FEF] |
             [#x3001-#xD7FF] | [#xF900-#xFDCF] |
             [#xFDF0-#xFFFD] | [#x10000-#xEFFFF]
[4a]    NameChar       ::=    NameStartChar | "-" | "." | [0-9] |
             #xB7 | [#x0300-#x036F] | [#x203F-#x2040]
[5]     Name    ::=    NameStartChar (NameChar)*
```

## 2.3 단일 항목 타입

네트워크 상에서 교환 되는 데이터 타입이다. 이 타입은 컴퓨터 언어의 제약 때문에 제한 될 수 있다. 예를 들면 unsigned long 은 네트워크 상에서는 가능한 타입이지만 자바에서 지원하지 않는다.

### 2.3.1 (unsigned) byte

### 2.3.2 (unsigned) short

### 2.3.3 (unsigned) int

### 2.3.4 long

### 2.3.5 `fixed length byte[]`

고정 크기를 갖는 바이트 배열, 타입 부가 정보 크기는 필수

### 2.3.6 `ub variable length byte[], us variable length byte[], si variable length byte[]`

가변 배열 크기를 갖는 바이트 배열, 접두어 ub, us, si 는 배열 크기를 말하며 si는 unsigned byte 로, us 는 unsigned short, si 는 signed int 를 뜻한다.

#### 2.3.7 ub pascal string, us pascal string, si pascal string

파스칼 문자열은 앞 부분 1byte에 문자열 크기가 명시된 문자 배열이다. 255 byte 크기로 제한 되지만 전통적인 C언어의 null 문자열 보다 매우 효율적이다. 문자열 크기를 Unsigned Byte, Unsigned Short, Signed Int 그 종류가 나뉘어 진다.

참고) 문자열 크기의 단위는 byte 이다.

#### 2.3.8 fixed length string

고정 크기 문자열, 타입 부가 정보 중 크기는 필수, 문자셋은 옵션

## 3 신놀이 메시지 구현 구성 요소

### 3.1 메시지 정보를 담고 있는 XML 파일

샘플) 2013.06.08 버전

```
<?xml version="1.0" encoding="utf-8" ?>
<!--
      item type : (unsigned) byte, (unsigned) short, (unsigned) integer, long,
                              ub pascal string, us pascal string, si pascal
string, string, byte[]
      item itemuseyn : Y 사용, N 미사용
      array counter type : reference 변수참조, direct 직접입력
-->
```

```
<sinnori_message>
<messageID>AllDataType</messageID>
<desc>고정 크기 스트림에서 모든 데이터 타입을 테스트 하기 위한 메시지</desc>
<singleitem name="byteVar1" type="byte" />
<singleitem name="byteVar2" type="byte" />
<singleitem name="byteVar3" type="byte" value="123" />
<singleitem name="unsignedByteVar1" type="unsigned byte" />
<singleitem name="unsignedByteVar2" type="unsigned byte" />
<singleitem name="unsignedByteVar3" type="unsigned byte" />
<singleitem name="shortVar1" type="short" />
<singleitem name="shortVar2" type="short" />
<singleitem name="shortVar3" type="short" />
<singleitem name="unsignedShortVar1" type="unsigned short" />
<singleitem name="unsignedShortVar2" type="unsigned short" />
<singleitem name="unsignedShortVar3" type="unsigned short" />
<singleitem name="intVar1" type="integer" />
<singleitem name="intVar2" type="integer" />
<singleitem name="intVar3" type="integer" />
<singleitem name="unsignedIntVar1" type="unsigned integer" />
<singleitem name="unsignedIntVar2" type="unsigned integer" />
<singleitem name="unsignedIntVar3" type="unsigned integer" />
<singleitem name="longVar1" type="long" />
```

```
<singleitem name="longVar2" type="long" />
<singleitem name="longVar3" type="long" />
<singleitem name="strVar1" type="ub pascal string" />
<singleitem name="strVar2" type="us pascal string" />
<singleitem name="strVar3" type="si pascal string" />
<singleitem name="bytesVar1" type="fixed length byte[]" size="2" />
<singleitem name="bytesVar2" type="variable length byte[]" />
<singleitem name="cnt" type="integer" />
<array name="memberList" cnttype="reference" cntvalue="cnt">
      <singleitem name="memberID" type="fixed length string" size="30"
value="king" />
      <singleitem name="memberName" type="fixed length string" size="30" />
      <singleitem name="cnt" type="integer" value="10" />
      <array name="itemList" cnttype="reference" cntvalue="cnt">
            <singleitem name="itemID" type="fixed length string" size="30" />
            <singleitem name="itemName" type="fixed length string" size="30" />
            <singleitem name="itemCnt" type="integer" />
      </array>
</array>
</sinnori_message>
```

### 3.2 메시지 정보를 담고 있는 XML의 구조를 정의한 XSD

메시지 정보를 담고 있는 XML 파일의 구조를 신놀이 메시지 규격에 맞도록 강제하기 위한 XML Schema 파일.  참고) 신놀이 메시지 규격은 정규식으로 표현되는 메시지 논리 구조를 뼈대로 기타 요건을 가지고 있다. 부족한 부분은 파싱 과정에서

샘플) 2013.07.17 버전

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

	<!-- 항목 그룹 -->
	<xs:group name="itemgroup">
		<xs:choice>
			<!-- 단일 항목 -->
			<xs:element name="singleitem">
				<xs:complexType>
					<xs:sequence>
						<xs:element name="desc" type="xs:string"
minOccurs="0" maxOccurs="1" />
					</xs:sequence>
					<!-- 이름 -->
					<xs:attribute name="name" use="required">
						<xs:simpleType>
							<xs:restriction base="xs:string">
								<xs:minLength value="1" />
```

```
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:attribute>

                                <!-- 항목 타입, 항목 타입은 크게 2가지로 나뉘는데 숫자형과 문
자형이 있다. 숫자형 같은 경우 정수만 지원하며 부호
                                        있음과 부호 없음으로 나뉘며 비 부호(= 부호 없음)만 앞에
표시한다. 단 특정 언어의 경우 예를 들면 자바의 경우 부호 없음을 지원하지
                                        않으므로 이를 소프트웨어로 구현한다. 소프트웨어 구현에
는 한계가 있다 예를 들면 unsigned long 같은 경우 자바로 구현할려고
                                        하면 불가능에 가까운 매우 힘든 일이다. 따라서 반듯이 신
놀이를 구현하는 언어 특성으로 기인하는 타입 제한을 숙지해야 한다. 타입
                                        제한을 극복 하는 방법으로 문자열 그 자체로 보내고 클라이
언트 혹은 비지니스 측에서 이를 적절하게 변환하여 사용하는것을 추천한다.
                                        실수형 데이터의 경우 이렇게 해결하기를 바란다. 예제)
unsigned byte, 배열은 byte 만 지원한다. 예제) byte[]
                                        숫자형 타입 목록 : byte, short, integer, long 문
자형 타입 목록 : string -->
                                <xs:attribute name="type" use="required">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                            <xs:enumeration value="byte" />
```

```
                                                <xs:enumeration value="unsigned
byte" />
                                                <xs:enumeration
value="short" />
                                                <xs:enumeration value="unsigned
short" />
                                                <xs:enumeration value="integer"
/>
                                                <xs:enumeration value="unsigned
integer" />
                                                <xs:enumeration value="long" />
                                                <xs:enumeration value="ub
pascal string" />
                                                <xs:enumeration value="us
pascal string" />
                                                <xs:enumeration value="si
pascal string" />
                                                <xs:enumeration value="fixed
length string" />
                                                <xs:enumeration value="ub
variable length byte[]" />
                                                <xs:enumeration value="us
```

```
variable length byte[]" />
                                                <xs:enumeration value="si
variable length byte[]" />
                                                <xs:enumeration value="fixed
length byte[]" />
                                          </xs:restriction>
                                    </xs:simpleType>
                              </xs:attribute>

                              <!-- 타입 부가 정보인 크기는 2가지 타입에서만 유용하다. (1)
고정 크기 바이트 배열(fixed length byte[])
                                    (2) 고정 크기 문자열(fixed length string) -->
                              <xs:attribute name="size" use="optional">
                                    <xs:simpleType>
                                          <xs:restriction base="xs:string">
                                                <xs:minLength value="1" />
                                          </xs:restriction>
                                    </xs:simpleType>
                              </xs:attribute>

                              <!-- 타입 부가 정보인인 문자셋은 오직 고정 크기 문자열(fixed
length string)에서만 유효하다. -->
```

```
                        <xs:attribute name="charset" use="optional">
                              <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                          <xs:minLength value="1" />
                                    </xs:restriction>
                              </xs:simpleType>
                        </xs:attribute>

                        <!-- 값 -->
                        <xs:attribute name="value" use="optional">
                              <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                          <xs:minLength value="1" />
                                    </xs:restriction>
                              </xs:simpleType>
                        </xs:attribute>
                  </xs:complexType>
            </xs:element>

            <!-- 배열 -->
            <xs:element name="array">
                  <xs:complexType>
```

```
                        <!-- 항목 그룹 -->
                        <xs:sequence>
                              <xs:group minOccurs="0"
maxOccurs="unbounded" ref="itemgroup" />
                        </xs:sequence>

                        <!-- 이름 -->
                        <xs:attribute name="name" use="required">
                              <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                          <xs:minLength value="1" />
                                    </xs:restriction>
                              </xs:simpleType>
                        </xs:attribute>
                        <!-- 배열의 반복 횟수 지정 방식(cnttype)은 2가지가 있다.
(1) 직접(direct) : 고정 크기 지정방식으로
                              배열 반복 횟수에는 배열의 반복 횟수 값이 저장되며, (2)
참조(reference) : 가변 크기 지정방식으로 배열 반복 횟수는
                              참조하는 항목의 값이다. -->
                        <xs:attribute name="cnttype" use="required">
                              <xs:simpleType>
                                    <xs:restriction base="xs:string">
```

```
                                        <xs:enumeration
value="reference" />
                                        <xs:enumeration
value="direct" />
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                            <!-- 배열의 반복 횟수(cntvalue) "배열의 반복 횟수 지정 방
식"이 직접(direct) 이면 배열 반복 횟수를
                                    반환하며, 참조(reference)일 경우에는 참조하는 항목
이름을 반환한다. 참조하는 항목은 숫자형으로 배열과 같은 단계로 반듯이
                                    앞에 나와야 한다. 이렇게 앞에 나와야 하는 이유는 배열 정
보를 읽어와서 배열 정보를 저장하기 전에 참조 변수가 같은 레벨에서 존재하며
                                    숫자형인지 판단을 하기 위해서이다. 메시지 정보 파일을 순
차적으로 읽기 때문에 배열 뒤에 위치하면 알 수가 없다. -->
                            <xs:attribute name="cntvalue" use="required">
                                <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                        <xs:minLength value="1" />
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
```

```
                        </xs:complexType>
                  </xs:element>
            </xs:choice>
      </xs:group>

      <!-- 메시지 -->
      <xs:element name="sinnori_message">
            <xs:complexType>
                  <xs:sequence>
                        <!-- 메시지 식별자 -->
                        <xs:element name="messageID" minOccurs="1"
maxOccurs="1">
                              <xs:simpleType>
                                    <xs:restriction base="xs:string">
                                          <xs:pattern value="[a-zA-Z]
[a-zA-Z1-9]+" />
                                    </xs:restriction>
                              </xs:simpleType>
                        </xs:element>
                        <!-- 항목 그룹 -->
                        <xs:element name="desc" type="xs:string" minOccurs="0"
                              maxOccurs="1" />
```

```
                        <xs:group minOccurs="0" maxOccurs="unbounded"
ref="itemgroup" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
</xs:schema>
```

### 3.3 메시지 정보 파일 SAX 파싱기

메시지 정보를 담고 있는 XML의 구조를 정의한 XSD를 가지고 XML로 작성된 메시지 정보 파일을 SAX 방식으로 파싱하여 메시지 정보를 작성하여 반환한다.

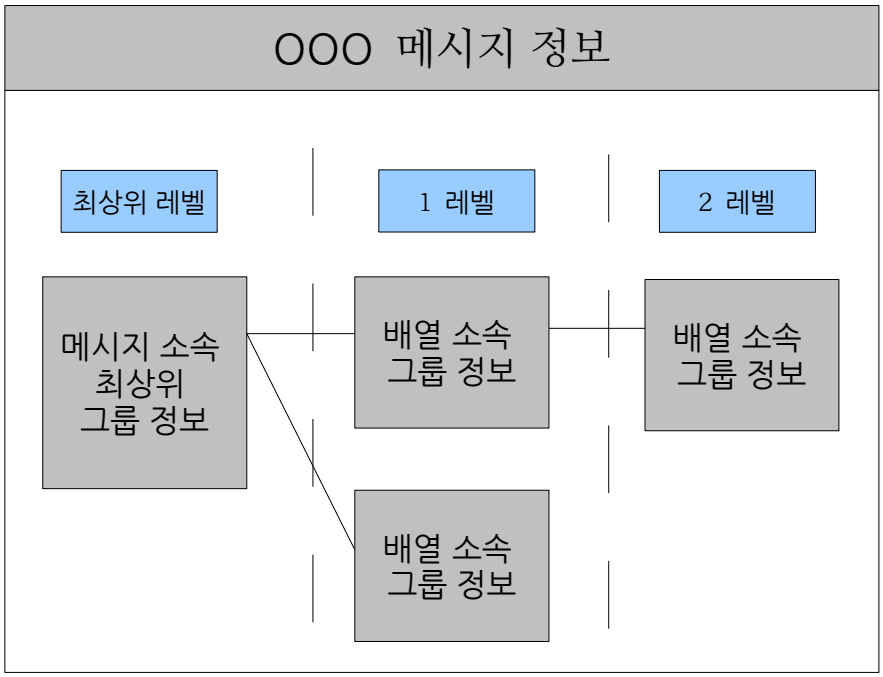
### 3.4 XML 관리기

메시지 정보를 담고 있는 XML 파일들을 메시지 정보 파일 SAX 파싱기를 통해서  키: 메시지 식별자별, 값:메시지 정보를 갖는 해쉬에 저장한다. 외부에서 원하는 메시지 식별자를  요청하면 해당 메시지 식별자에 해당하는 메시지 정보를 반환해 준다.

### 3.5 메시지 정보

메시지는 항목들로 구성되는데  이 항목은 이름, 타입, 크기, 값으로 구성된다. 이중 이름, 타입, 크기를 항목 정보라 한다. 이렇게 항목 정보 만을 모아 놓은 것 을 메시지 정보라 한다.  참고) 메시지 항목들은 순서를 갖는다.

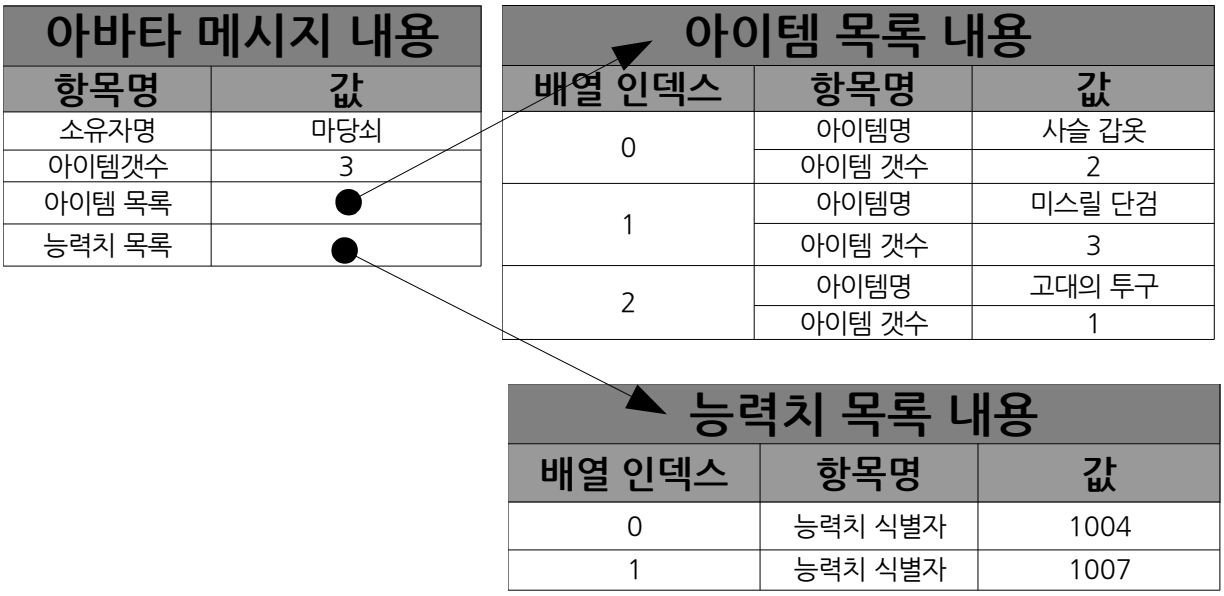
### 3.5.1 개념도




### 3.5.2 물리 구조 관점의 개념도

- 메시지의 최상의 그룹 시작 -
항목 1
항목 2
- 항목 3 : 배열 시작 -
항목 3 배열 의 항목 1
항목 3 배열의 항목 2
- 항목 3 배열의 항목 3 : 배열 시작 -
항목 3 배열의 항목 3 배열 항목 1
항목 3 배열의 항목 3 배열 항목 2
- 항목 3 배열의 항목 3 : 배열 종료 -
항목 3 배열의 항목 4
- 항목 3 : 배열 종료 -
- 항목 4 : 배열 시작 -
항목 4 배열의 항목 1
항목 4 배열의 항목 2
- 항목 4 : 배열 종료 -
항목 5
- 메시지의 최상의 그룹 종료 -

### 3.5.3 구현 관점의 개념도




**아이템 메시지 정보**

| 순번 | 항목 구분 | 항목 정보 | | | | |
|---|---|---|---|---|---|---|
| 1 | 단일항목 | 항목명 | 타입 | 크기 | 문자셋 | 디폴트값 |
| | | 소유자명 | ub pascal string | | | |
| 2 | 단일항목 | 항목명 | 타입 | 크기 | 문자셋 | 디폴트값 |
| | | 아이템갯수 | integer | | | 5 |
| 3 | 배열 | 항목명 | 반복 횟수 지정 방식 | 반복 횟수 | 소속 항목 그룹 정보 | |
| | | 아이템 목록 | 아이템 갯수 참조 | | | |
| 4 | 배열 | 항목명 | 반복 횟수 지정 방식 | 반복 횟수 | 소속 항목 그룹 정보 | |
| | | 능력치 목록 | 고정 | 2 | | |

**아이템 목록 소속 항목 그룹 정보**

| 순번 | 항목 구분 | 항목 정보 | | | | |
|---|---|---|---|---|---|---|
| 1 | 단일항목 | 항목명 | 타입 | 크기 | 문자셋 | 디폴트값 |
| | | 아이템명 | ub pascal string | | | |
| 2 | 단일항목 | 항목명 | 타입 | 크기 | 문자셋 | 디폴트값 |
| | | 아이템갯수 | integer | | | 5 |

**능력치 목록 소속 항목 그룹 정보**

| 순번 | 항목 구분 | 항목 정보 | | | | |
|---|---|---|---|---|---|---|
| 1 | 단일항목 | 항목명 | 타입 | 크기 | 문자셋 | 디폴트값 |
| | | 능력치 식별자 | integer | | | |

## 3.6 특수 메시지

### 3.6.1 예외 처리를 클라이언트에서 수행하기 위한 예외(=SelfExn)

SelfExn 메시지의 목적은 예외 상황을 클라이언트에 알려주는것을 목적으로 한다. 최종 메시지를 받은 수신단에서 SelfExn 메시지의 내용에 맞는 예외를 발생 시킨다. SelfExn 메시지의 내용은 서버/클라이언트 구분, 예외 구분 (B:BodyFormatException, N:NoMoreDataPacketBufferException, D:DynamicClassCallException, M:MessageInfoNotFoundException), 예외가 발생한 메시지 식별자 , 구체적인 에러 내용으로 구성된다.

**가** 서버

① BodyFormatException : 구분 B

(1) 입력 메시지를 바디 부분을 읽는 과정에서 발생한 오류 혹은 (2) 출력 메시지를 랩 메시지로 담는 과정에서 발생한 오류

② NoMoreDataPacketBufferException : 구분 N

(1) 래퍼 입력 메시지를 읽을때 바디 버퍼 확보 실패 혹은 (2) 원본 메시지를 래퍼 메시지 그룹에 담을때 바디 버퍼 확보 실패시 발생

③ DynamicClassCallException : : 구분 D

입력 메시지 1:1 대응 비지니스 로직 클래스 로딩 실패

④ MessageInfoNotFoundException : : 구분 M

(1) 입력 메시지의 메시지 정보 파일이 없을때 혹은 (2) 출력 메시지의 메시지 정보 파일이 없을때 발생

**나** 클라이언트

① BodyFormatException : : 구분 B

출력 메시지를 바디 부분을 읽는 과정에서 발생한 오류

② NoMoreDataPacketBufferException : 구분 N

래퍼 출력 메시지를 읽을때 바디 버퍼 확보 실패

③ MessageInfoNotFoundException : 구분 M

출력 메시지의 메시지 정보 파일이 없을때 발생

# 4 제한 사항, 문제점 혹은 향후 개선 방향

## 4.1 단계가 있는 항목에 직접 접근 불가

메시지는 트리 구조를 갖는데 최상의 단계를 제외한 항목에 대해서 직접적으로 접근 불가능하다. 최상위 루트 단계에서 부터 한 단계씩 내려 오면서 최종 항목에 접근 해야 한다.

## 4.2 메시지 정보 갱신시 문제점 돌파 전략

메시지 정보를 갱신하는 시점에서 바로 적용 되기 때문에 (1) 서버 비지니스 로직은 기존 메시지에 맞추어져 있다면 해당 비지니스 로직 수행시 틀린 부분에서 문제가 발생할 것이다. (2) 서버나 클라이언트 둘중 하나는 최신 메시지와 구 메시지를 갖게 되어 구와 신 메시지 차이로 인한 문제점을 격게 될 것이다.

### 4.2.1 무시

기존 메시지 에러 나는 대로 그냥 냅두자는 전략이다. 사이드 이팩트에 대해서 무력하다. **현재 신놀이 프레임 워크 기본 정책이다.**

### 4.2.2 안전한 메시지 정보 갱신 방법 사용

#### 가 절차

① 메시지 신규 생성을 잠시 보류하고

② 기존 메시지에 대한 처리가 종료되게 한 후

③ 시스템 메시지 관련 모든 작업을 멈춘다.

④ 수정된 메시지 정보에 맞춘 비지니스 로직을 재 로딩 한다.

⑤ 메시지 정보를 갱신한다.

⑥ 멈추었던 시스템을 메시지 관련 작업을 다시 시작한다.

나 기술 검토 사항

자바 서버에서 이렇게 할려면 각 역활을 담당하는 쓰레드를 차례로 멈추는 작업이 ㉠, ㉡, ㉢ 항까지의 작업이며 ㉣ 항과 ㉤ 항을 수행후 다시 쓰레드를 역으로 재 가동시키는 작업이 ㉥항이다.