목 차

Ⅰ. 오즈 엔터프라이즈 서버 **API** **4**
Class Cache 6
Class ConnectionPool 11
Class DataBind 23
Class Log 27
Class Mail 31
Class Module 38
Class Monitor 53
Class Service 58
Class Viewer 62
Class Servlet 70

Ⅱ. 오즈 스케줄러 서버 **API** **83**
Class Program 85
Class Publisher 91
Class Scheduler 95
Class TaskHolidayInfo 145
Class TaskHolidayGroupInfo 149

Ⅲ. 오즈 리파지토리 **API** **153**
Interface Repository 154
Class RepositoryEX 204
오즈 리파지토리 구현 233

Ⅳ. 오즈 리포트 뷰어 **API** **240**
OZLauncherDll을 이용한 호출 방법 241

Ⅴ. **RDB Store DataAction** **243**
RDB Store DataAction 인터페이스 244
RDB Store DataAction 구현 245

Ⅵ. **User Security Logic** **247**
USL 설정 248
USL 구현 253

Ⅶ. **User Defined Log** **264**
오즈 엔터프라이즈 서버에서 사용 265
오즈 스케줄러 서버에서 사용 309

Ⅷ. **User Defined post Execute** **314**
UDE 인터페이스 315
UDE 구현 317

Ⅸ. **User Defined Resource** **323**
UDR 인터페이스 324
UDR 구현 328

Ⅹ. 사용자 컴포넌트 333
연동 방법 334
사용자 컴포넌트 C 구현 함수 335

Appendix 1. SchedulerCom 활용 352

Ⅰ. 오즈 엔터프라이즈 서버 API

- Class Cache
- Class ConnectionPool
- Class DataBind
- Class Log
- Class Mail
- Class Module
- Class Monitor
- Class Service
- Class Viewer
- Class Servlet

오즈 엔터프라이즈 서버에 관련된 각종 정보 조회와 실시간 환경 설정 변경 기능을 사용자 애플리케이션에서 직접 제어할 수 있도록 자바 API를 제공합니다.

다음은 오즈 엔터프라이즈 서버 API로 제공하는 주요 클래스에 대한 설명입니다.

클래스 이름	설명
Cache	오즈 서버 캐시 매니저와 관련된 처리를 수행합니다. 캐시 매니저 환경 설정 및 캐시 항목 삭제 등을 처리합니다.
Connection Pool	오즈 서버의 데이터베이스 연결 Pool에 대한 인터페이스로 해당 데이터베이스의 JDBC/ODBC 정보의 등록, 삭제 등의 관리 기능을 수행합니다.
DataBind	데이터 바인딩 모듈 관련 기능을 수행합니다.
Log	오즈 서버 로그 매니저와 관련된 처리를 수행하며 서버 운용시 발생되는 각종 메시지, 정보, 에러 등을 기록하기 위한 환경 설정을 변경합니다.
Mail	메일 전송 관련 기능을 수행합니다.
Module	오즈 데이터 모듈 관련 기능을 수행합니다.
Monitor	오즈 서버의 실시간 모니터링 관련 기능을 수행합니다.
Service	서버 정보, 상태, 처리, 메모리 수집 등의 처리 기능을 수행합니다.
Viewer	보고서 폼의 데이터 모듈, 뷰어 실행 등의 처리 기능을 수행합니다.
Servlet	오즈 서블릿 서버와 통신하여 OZD 파일 등을 생성하는 기능을 수행합니다.

오즈 엔터프라이즈 서버 API를 사용하기 위해서는 다음과 같은 라이브러리 파일을 클래스 패스에 추가해 주어야 합니다.

라이브러리 파일명	설명
ozsfw60.jar	오즈 서버 및 Scheduler server에 접속하기 위한 파일입니다.
log4.jar	Server 내부의 클래스들을 사용할 때 Log를 남기기 위한 라이브러리 파일입니다. (API를 이용한 프로그램 실행시 classpath에 "log4.jar" 파일이 위치하여야 함)

Class Cache

Constructor Summary

- **Cache(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Cache(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **SortProperties getCacheConfiguration()**
- **void setCacheConfiguration(SortProperties p)**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우**
	`public Cache(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)`
	//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우
	`public Cache(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	

Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";

	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ getCacheConfiguration

Prototype	`public SortProperties getCacheConfiguration() throws OZCPException`
Definition	오즈 서버의 캐시 설정을 가져옵니다. 가져올 수 있는 설정값은 "SortProperties" 클래스의 key를 참조하시기 바랍니다.

■ setCacheConfiguration

Prototype	`public void setCacheConfiguration(SortProperties p) throws OZCPException`	
Definition	캐시 설정을 변경합니다. 변경할 수 있는 설정값은 "SortProperties" 클래스의 key를 참조하시기 바랍니다.	
Argument	*p*	캐시 설정 속성 값

관련 Class

■ OZCPException(oz.framework.cp.OZCPException)

API를 사용할 때 생성되는 Exception입니다. 모든 API에서 OZCPException을 발생시킵니다.

- 메소드
  - getMessage

Prototype	`public String getMessage()`
Definition	에러 내용을 가져옵니다.

- getErrorCode

Prototype	`public int getErrorCode()`
Definition	에러 코드를 가져옵니다.

■ **SortProperties(oz.util.SortProperties.java)**

getCacheConfiguration(), setCacheConfiguration()의 설정값을 반환하거나 설정하는데 사용되는 클래스입니다.

- 메소드
  - getproperty / get

Prototype	`public synchronized String getProperty(String key)` `public synchronized Object get(String key)`
Definition	해당 key에 해당하는 속성 값을 가져옵니다.

  - setproperty / put

Prototype	`public synchronized Object setProperty(String key, String value)` `public synchronized Object put(Object key, Object value)`
Definition	해당 key에 해당하는 속성을 지정한 값(value)으로 설정합니다.

- Key

getProperty()와 setProperty()에서 사용되는 key는 다음 표와 같습니다.

Key	Value	설명
Active	"true" "false"	캐시키 사용 여부 ex) p.setProperty("datamodule.active", "false");
CACHE_FILE_PATH	저장 경로	캐시 저장 경로명 ex) p.setProperty("CACHE_FILE_PATH","%OZ_HOME%/cache");
DM_CACHE_FILE_PATH	저장 경로	Data Module 캐시 저장 경로 ex) p.setProperty("DM_CACHE_FILE_PATH","%OZ_HOME%/cache_dm/");

memoryCacheValidTime	시간	메모리 캐시 타임 아웃(단위:초) ex) p.setProperty("datamodule.memoryCacheValidTime", "100");
diskCacheValidTime	시간	디스크 캐시 타임 아웃(단위:초) ex) p.setProperty("datamodule.diskCacheValidTime","100");
FreeMemoryPercentage	메모리	프리 메모리 퍼센트(%) ex) p.setProperty("datamodule.freeMemoryPercentage", "20");

※ 참고사항 : 캐시키 설정 방법은 "오즈 엔터프라이즈 서버 관리자 매뉴얼의 cachemngr.properties" 부분을 참조하시기 바랍니다.

Sample : CacheSample.java

```
package sample;

import oz.framework.api.Cache;
import org.apache.log4j.*;

import oz.util.SortProperties;

public class CacheSample {
    public static void main(String[] args) {
        //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
        BasicConfigurator.configure();

        // OZServer Info.
        //*
        // Daemon
        String IP   = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
        int PORT = 8003;    //서버가 사용하는 TCP 포트
        /*/
        // Servlet
        String URL  = "http://www.oz.com/oz/server"; //Servlet에 접근 가능한 URL
        /**/
        // User Info.
        String ID   = "admin"; //default
        String PWD  = "admin"; //default

        Cache cache = null;
        try {
```

```
            //*
            // Daemon
            cache = new Cache(IP, PORT, ID, PWD, false, false);
            /*/
            // Servlet
            cache = new Cache(URL, ID, PWD, false, false);
            /**/

            SortProperties p = new SortProperties();

            /*/
            // 캐시 환경설정 정보 저장 (setCacheConfiguration)
            p.setProperty("CACHE_FILE_PATH", "%OZ_HOME%/cache");
            // 디스크 저장 및 프리 메모리를 벗어났을때 저장 경로
            p.setProperty("DM_CACHE_FILE_PATH", "%OZ_HOME%/cache_dm");
            // 캐시 대상이 DataModule 일시 저장경로
            p.setProperty("datamodule.active", "true");
            //캐시매니저 사용 여부
            p.setProperty("datamodule.memoryCacheValidTime", "1000");
            //메모리 캐시 타임 아웃 설정
            p.setProperty("datamodule.diskCacheValidTime", "1000");
            //디스크 캐시 타임 아웃 설정
            p.setProperty("datamodule.freeMemoryPercentage", "21");
            //프리 메모리 퍼센트

            cache.setCacheConfiguration(p);

            /*/
            // 캐시 환경설정 정보 가져오기(getCacheConfiguration)
            p = cache.getCacheConfiguration();
            /**/
            p.list(System.out);
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Class ConnectionPool

Constructor Summary

- **ConnectionPool(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **ConnectionPool(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **void addPool(ConnectionPoolInfo pool)**
- **void removePool(String pool)**
- **ConnectionPoolInfo[] getPoolInfoList()**
- **ConnectionPoolStatus[] getPoolStatusList()**
- **ConnectionPoolInfo getPoolInfo(String alias)**
- **void save()**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우**
	`public ConnectionPool(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)`
	//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우
	`public ConnectionPool(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	

Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";

	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ addPool

Prototype	`public void addPool(ConnectionPoolInfo pool) throws OZCPException`
Definition	ConnectionPool을 추가합니다. 추가할 ConnectionPool은 "ConnectionPoolInfo" 클래스에서 선언되어진 멤버 변수에 값을 설정하여 추가합니다.
Argument	*pool* 추가할 ConnectionPool의 정보인 ConnectionPoolInfo 객체

■ removePool

Prototype	`public void removePool(String pool) throws OZCPException`
Definition	해당 이름의 ConnectionPool을 삭제합니다.
Argument	*pool* ConnectionPool 이름

■ getPoolInfoList

Prototype	`public ConnectionPoolInfo[] getPoolInfoList() throws OZCPException`
Definition	모든 ConnectionPool에 대한 ConnectionPoolInfo 리스트를 가져옵니다.

■ getPoolStatusList

Prototype	`public ConnectionPoolStatus[] getPoolStatusList() throws OZCPException`

Definition	모든 ConnectionPool에 대한 상태 정보를 가져옵니다.

■ **getPoolInfo**

Prototype	`public ConnectionPoolInfo getPoolInfo(String alias) throws OZCPException`
Definition	해당 ConnectionPool에 대한 ConnectionPoolInfo 리스트를 가져옵니다.
Argument	*alias* ConnectionPool의 앨리어스 이름

■ **save**

Prototype	`public void save() throws OZCPException`
Definition	서버에 모든 ConnectionPool의 정보를 저장합니다.

관련 Class

■ **ConnectionPoolInfo(oz.framework.db.ConnectionPoolInfo.class)**

데이터베이스 연결에 대한 각종 정보들을 가지고 있는 클래스입니다.

- 메소드

▪ getAlias

Prototype	`public String getAlias()`
Definition	해당 연결 Pool 정보의 alias명을 가져옵니다.

▪ setAlias

Prototype	`public void setAlias(String _alias)`
Definition	연결 Pool 정보의 alias를 설정합니다.
Argument	*_alias* 설정할 Alias명

▪ getVendor

Prototype	`public String getVendor()`
Definition	해당 연결 Pool 정보의 Vendor명을 가져옵니다.

▪ setVendor

Prototype	`public void setVendor(String vendorName)`

Definition	연결 Pool 정보의 DB Vendor명을 설정합니다.
Argument	*vendorName* 설정할 DB Vendor명

- getDriver

Prototype	`public String getDriver()`
Definition	해당 연결 Pool 정보의 Driver명을 가져옵니다. 리턴되는 값 예) oracle.jdbc.driver.OracleDriver

- setDriver

Prototype	`public void setDriver(String _driver)`
Definition	연결 Pool 정보의 Driver명을 설정합니다.
Argument	*_driver* 설정할 Driver명

- getURL

Prototype	`public String getURL()`
Definition	해당 연결 Pool 정보의 URL을 가져옵니다.

- setURL

Prototype	`public void setURL(String _url)`
Definition	연결 Pool 정보의 URL를 설정합니다.
Argument	*_url* 설정할 URL

- getInitConns

Prototype	`public int getInitConns()`
Definition	해당 연결 Pool 초기 연결 개수를 가져옵니다.

- setInitConns

Prototype	`public void setInitConns(int _initConns)`
Definition	연결 Pool 초기 연결 개수를 설정합니다
Argument	*_initConns* 설정할 초기 연결 개수

- getMaxConns

Prototype	`public int getMaxConns()`

Definition	해당 연결 Pool 최대 연결 개수를 가져옵니다.

- setMaxConns

Prototype	`public void setMaxConns(int _maxConns)`
Definition	연결 Pool 최대 연결 개수를 설정합니다
Argument	*_maxConns* 설정할 최대 연결 개수

- getTimeout

Prototype	`public int getTimeout()`
Definition	해당 연결을 가져올때 timeout 시간을 가져옵니다. (단위 : sec)

- setTimeout

Prototype	`public void setTimeout(int _timeout)`
Definition	연결을 가져올 timeout시간을 설정합니다.
Argument	*_timeout* 설정할 timeout 시간

- getDoConnectionCheck

Prototype	`public boolean getDoConnectionCheck()`
Definition	연결을 가져올 당시 해당 연결이 유효한지 여부를 체크합니다

- setDoConnectionCheck

Prototype	`public void setDoConnectionCheck(boolean _check)`
Definition	연결을 가져올 당시 해당 연결의 유효 여부를 설정합니다.
Argument	*_check* 체크 여부

- getTestQueryString

Prototype	`public String getTestQueryString()`
Definition	연결이 유효한지 여부를 체크하기 위한 쿼리문을 가져옵니다.

- setTestQueryString

Prototype	`public void setTestQueryString(String _testQuery)`
Definition	연결이 유효한지 여부를 체크하기 위한 쿼리문을 설정합니다.
Argument	*_testQuery* 유효 여부 테스트 쿼리문

- getInitSqls

Prototype	`public String getInitSqls()`
Definition	연결 Pool에서 사용할 연결을 가져온 뒤 바로 실행하는 쿼리문을 가져옵니다.

- setInitSqls

Prototype	`public void setInitSqls(String _initQuery)`
Definition	연결 Pool에서 사용할 연결을 가져온 뒤 바로 실행하는 쿼리문을 설정합니다.
Argument	*_initQuery* Init 쿼리문

- getCloseSqls

Prototype	`public String getCloseSqls()`
Definition	연결이 close 또는 free되기 전 바로 실행하는 쿼리문을 가져옵니다.

- setCloseSqls

Prototype	`public void setCloseSqls(String _closeQuery)`
Definition	연결이 close 또는 free되기 전 바로 실행하는 쿼리문을 설정합니다.
Argument	*_closeQuery* Close 쿼리문

- getSessionQuery

Prototype	`public String getSessionQuery()`
Definition	연결의 session 쿼리문을 가져옵니다.

- setSessionQuery

Prototype	`public void setSessionQuery(String _sessionQuery)`
Definition	연결의 session 쿼리문을 설정합니다.
Argument	*_sessionQuery* Session 쿼리문

- getAutoCommit

Prototype	`public String getAutoCommit()`
Definition	연결의 AutoCommit를 설정된 값을 가져옵니다.

- setAutoCommit

Prototype	`public void setAutoCommit(String _commit)`
Definition	연결의 AutoCommit을 설정합니다.
Argument	*_commit* AutoCommit 여부 : "true", "false"로 설정합니다.

- getProperties

Prototype	`public Properties getProperties()`
Definition	연결을 가져올 때 사용하는 Properties의 값을 가져옵니다.

- setProperties

Prototype	`public void setProperties(Properties _props)`
Definition	연결을 가져올때 Propeties 값을 설정합니다. 내부적으로 사용되는 예) ConnectionPoolInfo cpi = new ConnectionPoolInfo(); ... Driver driver = loadDriver(cpi.getDriver()); Connection conn = driver.connect(cpi.getURL(), cpi.getProperties());
Argument	*_props* properties

- getKey

Prototype	`public String getKey()`
Definition	연결 Pool의 unique한 Key값을 가져옵니다. connection pool마다 해당 key값이 unique합니다. 기본적으로 dbconfig.xml의 Vendor별 Items Key의 조합의 값으로 구성되어 있습니다.

- setKey

Prototype	`public void setKey(String _key)`
Definition	연결의 unique한 Key값을 설정합니다. 기본적으로 dbconfig.xml의 Vendor별 Items Key의 조합의 값으로 구성되어 처리되므로 설정하는것을 권장하지 않습니다.
Argument	*_key* key값

- getItems

Prototype	`public HashMap getItems()`
Definition	dbconfig.xml 파일의 Vendor마다 있는 Items의 key, Value값을 가져옵니다.

- setItems

Prototype	`public void setItems(String _map)`

Definition	dbconfig.xml 파일의 Vendor마다 있는 Items의 key값에 해당되는 value들을 설정합니다.
	예)
	HashMap infos = new HashMap();
	infos.put("serverAddress", "127.0.0.1");
	infos.put("portNo", "1433");
	infos.put("dbName", "master");
	infos.put("user", "sa");
	infos.put("password", "1588");
	poolInfo.setItems(infos);

Argument	*_map* key값

- getDecodeCharSet

Prototype	`public void getDecodeCharSet()`
Definition	해당 ConnectionPool에 대한 디코딩셋을 가져옵니다.

- setDecodeCharSet

Prototype	`public void setDecodeCharSet(String decode)`
Definition	해당 ConnectionPool에 대한 디코딩셋을 설정합니다.
Argument	*decode* ConnectionPool의 디코딩셋

- getEncodeCharSet

Prototype	`public String getEncodeCharSet()`
Definition	해당 ConnectionPool에 대한 인코딩셋을 가져옵니다.

- setEncodeCharSet

Prototype	`public void setEncodeCharSet(String encode)`
Definition	해당 ConnectionPool에 대한 인코딩셋을 설정합니다.
Argument	*encode* ConnectionPool의 인코딩셋

- getFetchRow

Prototype	`public int getFetchRow()`
Definition	JDBC 전송 ROW 즉, 결과 셋 반환 시 한꺼번에 가져올 수 있는 행 수를 가져옵니다.

- setFetchRow

Prototype	`public void setFetchRow(int rows)`
Definition	JDBC 전송 ROW 즉, 결과 셋 반환 시 한꺼번에 가져올 수 있는 행 수를 설정합니다.
Argument	*rows* 행 수

■ ConnectionPoolStatus(oz.framework.db.ConnectionPoolStatus.class)

ConnectionPool의 상태 정보를 멤버 변수로 가지고 있는 클래스입니다.

- 멤버 변수
  - public final static int OK = 1;

    ConnectionPool의 현재 상태를 나타냅니다. 각 상태 값에 대한 설명은 아래 표와 같습니다.

상태 값	**설명**
1	**OK** ConnectionPool 생성 성공
-1	**DRIVER_ERROR** ConnectionPool을 생성하는데 필요한 JDBC 드라이버를 불러오는데 실패
-2	**CONNECTION_ERROR** ConnectionPool을 생성 중 DBMS에서 초기 연결 획득 실패

  - public final static int DRIVER_ERROR = -1;
  - public final static int CONNECTION_ERROR = -2;

- 메소드
  - getStatusString

Prototype	`public String getStatusString()`
Definition	현재 상태 메세지를 가져옵니다. STATUS 값이 1인 경우 "OK", STATUS 값이 -1인 경우 "Fail to load or register JDBC driver" 그 외인 경우 "Fail to make initial connection"

  - getStatus

Prototype	`public int getStatus()`
Definition	현재 상태를 가져옵니다. 코드는 STATUS 값이 리턴됩니다.

- getFreeConnectionCount

Prototype	`public int getFreeConnectionCount()`
Definition	프리 상태의 연결 수를 가져옵니다.

- getCheckedOutConnectionCount

Prototype	`public int getCheckedOutConnectionCount()`
Definition	Checked out된 연결 개수를 가져옵니다.

- getErrorMessage

Prototype	`public String getErrorMessage()`
Definition	연결 Pool 생성시 발생한 Error Message를 가져옵니다.

Sample : ConnectionPoolSample.java

```
package sample;

import java.util.HashMap;
import oz.framework.api.ConnectionPool;
import oz.framework.db.ConnectionPoolInfo;
import oz.framework.db.ConnectionPoolStatus;
import org.apache.log4j.*;

public class ConnectionPoolSample {
    public static void main(String[] args) {
        //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
        BasicConfigurator.configure();

        // OZServer Info.

        //*
        // Daemon
        String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
        int PORT = 8003; //서버가 사용하는 TCP 포트

        /*/
        // Servlet
        String URL = "http://www.oz.com/oz/server"; //Servlet 에 접근 가능한 URL

        /**/
        // User Info.
```

```
        String ID = "admin";//default
        String PWD = "admin";//default
        ConnectionPool conPool = null;

        try {
             //*
             // Daemon
             conPool = new ConnectionPool(IP, PORT, ID, PWD, false, false);

             /*/
             // Servlet
             conPool = new ConnectionPool(URL, ID, PWD, false, false);

             /**/
             // 커넥션풀 추가(addPool)
             ConnectionPoolInfo poolInfo = new ConnectionPoolInfo();
             poolInfo.setAlias("forcs"); //커넥션 풀 이름
             poolInfo.setVendor("mssql"); //데이터베이스 타입(현재 MSSQL)

             // 서버의 연결정보를 저장한다.
             // 각 벤더별 map 의 키는 dbconfig.xml 에 db 연결 jdbc url 부분을 참조
             HashMap infos = new HashMap();
             infos.put("serverAddress", "127.0.0.1");
             infos.put("portNo", "1433");
             infos.put("dbName", "master");
             infos.put("user", "sa");
             infos.put("password", "1588");
             poolInfo.setItems(infos);

             poolInfo.setMaxConns(20); //데이터베이스 최대 접속
             poolInfo.setInitConns(1); //데이터베이스 초기 접속
             // 문자셋 설정
             //poolInfo.setEncodeCharSet("8859_1");
             //poolInfo.setDecodeCharSet("KSC5601");

             conPool.addPool(poolInfo);

             // 커넥션풀 제거하기(removePool)
             String conPoolName = "forcs"; //제거할 커넥션 풀 이름
             conPool.removePool(conPoolName);
             // ConnectionPoolStatus 의 정보 리스트 가져오기(getPoolStatusList)
             ConnectionPoolStatus[]
poolStatusList=conPool.getPoolStatusList();
             for (int i = 0; i < poolStatusList.length; i++) {
                ConnectionPoolStatus cps = poolStatusList[i];

                //커넥션풀의 상태
                System.out.println(i);
```

```
                System.out.println("Statusstring=" + cps. getStatusString
());

                //프리 커넥션 풀
                System.out.println("free=" + new
Integer(cps.getFreeConnectionCount()));

                //체크아웃커넥션
                System.out.println("checkedout=" + new
Integer(cps.getCheckedOutConnectionCount()));
                System.out.println();
            }
            //커넥션풀 정보 저장하기(save)
            conPool.save();
        }
        catch(Exception e) {
                e.printStackTrace();
        }
    }
}
```

Class DataBind

Constructor Summary

- **DataBind(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **DataBind(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **void setDataBindConfigration(SortProperties config)**
- **SortProperties getDataBindConfigration()**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우**
	`public DataBind(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)`
	//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우
	`public DataBind(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	

Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";

	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ setDataBindConfigration

Prototype	`public void setDataBindConfigration(SortProperties config) throws OZCPException`
Definition	DataBind 설정 즉, "databind.properties"에 설정되어 있는 값을 변경할 수 있습니다.
Argument	*config* DataBind 설정 속성

■ getDataBindConfigration

Prototype	`public SortProperties getDataBindConfigration() throws OZCPException`
Definition	DataBind의 설정값 즉, "databind.properties"에 설정되어 있는 값을 가져옵니다.

- Key

  setDataBindConfigration()와 getDataBindConfigration()에서 사용되는 key는 다음 표와 같습니다.

Key	Value	설명
ConcurrentFetch Size	버퍼 크기	FetchType이 "Concurrent"일 때 클라이언트로 데이터를 전송할때 Stream 버퍼의 크기를 설정합니다. 단위는 byte, 기본값은 4096, 최소값은 256입니다. ※ 주의사항 : 음수를 설정하거나 잘못된 형식으로 설정하면 기본값으로 설정되고, 최소값보다 작은 양수로 설정하면 최소값으로 설정됩니다.

ConcurrentFirstRow	행 개수	FetchType이 "Concurrent"일 때 클라이언트로 데이터를 전송할 때 맨 처음 전송되는 첫 데이터의 행 개수를 설정합니다. 단위는 개수, 기본값은 0입니다. ※ 주의사항 : 0보다 작은 값을 설정한 경우에는 기본값으로 설정됩니다.

Sample : DataBindSample.java

```
package sample;

import oz.framework.api.DataBind;
import org.apache.log4j.*;
import oz.util.SortProperties;

public class DataBindSample {
    public static void main(String[] args) {
        //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
        BasicConfigurator.configure();

        // OZServer Info.
        String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
        int PORT = 8003; //서버가 사용하는 TCP 포트
        // User Info.
        String ID = "admin"; //default
        String PWD = "admin"; //default

        DataBind dataBind = null;
        try {
            dataBind = new DataBind(IP, PORT, ID, PWD, false, false);
            SortProperties p = new SortProperties();

            // 데이터바인드 환경설정 정보 저장 (setDataBindConfiguration)
            //Stream 버퍼의 크기를 설정
            p.setProperty("ConcurrentFetchSize", "4096");

            //최초 row갯수 지정
            p.setProperty("ConcurrentFirstRow", "0");
            dataBind.setDataBindConfiguration(p);

            // 데이터바인드 환경설정 정보 가져오기(getDataBindConfiguration)
            p = dataBind.getDataBindConfigration();
            java.util.Vector vec = p.propertyNames();
            for(int i=0; i<vec.size(); i++) {
```

```
            String name = (String)vec.elementAt(i);
            System.out.println(name + "=" + p.getProperty(name));
         }
      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Class Log

Constructor Summary

- **Log(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Log(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **String getConfiguration()**
- **byte[] downloadLog()**
- **void downloadLog(String fileName)**
- **void setConfiguration(String logs)**
- **void setPriority(String p)**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우**
	`public Log(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)`
	//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우
	`public Log(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	

Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;

	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ getConfigure

Prototype	`public String getConfigure() throws OZCPException`
Definition	로그 설정을 서버로부터 가져옵니다.

■ downloadLog

Prototype	`public byte[] downloadLog() throws OZCPException`
Definition	서버로부터 로그 파일을 다운로드합니다.

■ downloadLog

Prototype	`public void downloadLog(String fileName) throws OZCPException, IOException`	
Definition	서버로부터 다운로드한 로그 파일을 로컬에 저장합니다.	
Argument	*fileName*	저장할 파일명

■ setConfigure

Prototype	`public void setConfigure(String logs) throws OZCPException`	
Definition	로그 환경 설정값을 변경합니다.	
Argument	*logs*	변경할 로그 환경 설정값, "key=value" 형식으로 설정 ex) String logs="Priority=DEBUG" ex) String logs="CONSOLE.Layout=%r[%t]%p%c{1}%X-%m%n

■ **setPriority**

Prototype	public void setPriority(String p) throws OZCPException
Definition	로그의 레벨을 설정합니다.(INFO, DEBUG, ERROR)
Argument	*p* 변경할 로그 레벨값

Sample : LogSample.java

```
package sample;

import oz.framework.api.Log;
import org.apache.log4j.*;

public class LogSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      // OZServer Info.
      // Daemon
      String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 8003; //서버가 사용하는 TCP 포트
      // Servlet
      String URL  = "http://www.oz.com/oz/server"; //Servlet에 접근 가능한 URL
      // User Info.
      String ID = "admin"; //default
      String PWD = "admin"; //default
      Log log = null;
      try {
         // Daemon
         log = new Log(IP, PORT, ID, PWD, false, false);
         // Servlet
         //log = new Log(URL, ID, PWD, false, false); //comment 처리
         // 로그 환경 정보 가져오기(getConfigure)
         String conf = null;
         conf = log.getConfigure();
         System.out.println(conf);
         // 로그 환경 설정(setConfigure, setProirity)
         String logs = "Priority=INFO";
         //log.setConfigure(logs);
         log.setPriority("DEBUG");
         // 로그파일 내려받기(downloadLog)
         byte b[] = log.downloadLog();
         String fileName = "server.log";
         // 내용을 파일로 저장합니다.
```

```
            log.downloadLog(fileName);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Class Mail

Constructor Summary

- **Mail(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Mail(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **void addAlias(String aliasName, SortProperties p) throws OZCPException**
- **SortProperties getAliasConfig(String aliasName) throws OZCPException**
- **String[] getMailAliasNames()**
- **void modifyAlias(String aliasName, String newAliasName, SortProperties p) throws OZCPException**
- **void removeAlias(String aliasName) throws OZCPException**
- **void send(String aliasName, String from, String fromUserName, String to, String cc, String bcc, String subject, String context, boolean isHTML, String localFileFullPath, String fileName) throws OZCPException**
- **boolean sendSync(String aliasName, String from, String fromUserName, String to, String cc, String bcc, String subject, String context, boolean isHTML, String localFileFullPath, String fileName) throws OZCPException**

Constructor Detail

Prototype

```
//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우
public Mail(String ip, int port, String id, String pw,
boolean bAutoLogin, boolean useUSL)

//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우
public Mail(String url, String id, String pw, boolean
bAutoLogin, boolean useUSL)
```

Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ addAlias

Prototype	`public void addAlias(String aliasName, SortProperties p) throws OZCPException`	
Definition	오즈 서버에 설정한 앨리어스 이름으로 메일 설정 정보를 추가합니다.	
Argument	*aliasName*	앨리어스 이름
	p	메일 설정 속성 값 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

■ getAliasConfig

Prototype	`public SortProperties getAliasConfig(String aliasName) throws OZCPException`	
Definition	해당 앨리어스의 메일 설정 정보를 가져옵니다.	
Argument	*aliasName*	앨리어스 이름

■ getMailAliasNames

Prototype	`public String[] getMailAliasNames()`
Definition	오즈 서버의 mail.properties에 설정된 메일 앨리어스를 모두 가져옵니다. ※ 참고사항 : 앨리어스의 active 여부와 관계없이 설정된 모든 앨리어스를 가져옵니다.

■ **modifyAlias**

Prototype	`public void modifyAlias(String aliasName, String newAliasName, SortProperties p) throws OZCPException`	
Definition	해당 앨리어스의 앨리어스 이름 및 메일 설정 정보를 변경합니다.	
Argument	*aliasName*	앨리어스 이름
	newAliasName	변경할 앨리어스 이름
	p	메일 설정 속성 값 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

■ **removeAlias**

Prototype	`public void removeAlias(String aliasName) throws OZCPException`	
Definition	해당 앨리어스의 메일 설정 정보를 삭제합니다.	
Argument	*aliasName*	앨리어스 이름

■ **send**

Prototype	`public void send(String aliasName, String from, String fromUserName, String to, String cc, String bcc, String subject, String context, boolean isHTML, String localFileFullPath, String fileName) throws OZCPException`	
Definition	해당 앨리어스에 설정된 메일 정보로 메일을 비동기 방식으로 전송합니다.	
Argument	*aliasName*	앨리어스 이름
	from	발신자 메일 주소
	fromUserName	발신자 이름 값을 null 또는 ""로 설정할 경우 발신자 메일 주소로 적용됨
	to	수신자 메일 주소
	cc	참조 메일 주소 값을 null 또는 ""로 설정할 경우 참조 메일 주소로 적용되며, 여러 값을 설정할 경우 "," 또는 ";"를 구분자

	로 설정
bcc	숨은 참조 메일 주소 값을 null 또는 ""로 설정할 경우 숨은 참조 메일 주소로 적용되며, 여러 값을 설정할 경우 "," 또는 ";"를 구분자로 설정
subject	메일 제목
context	메일 내용
isHTML	메시지 형식을 HTML로 설정할지 여부
localFileFullPath	파일을 첨부할 경우 첨부할 파일의 전체 경로 여러 값을 설정할 경우 "|"를 구분자로 설정
fileName	파일을 첨부할 경우 첨부 파일명 ※ 주의사항 : 첨부 파일의 구분자인 "|" 문자열은 파일명에 포함될 수 없습니다.

■ sendSync

Prototype	`public boolean sendSync(String aliasName, String from, String fromUserName, String to, String cc, String bcc, String subject, String context, boolean isHTML, String localFileFullPath, String fileName) throws OZCPException`	
Definition	해당 앨리어스에 설정된 메일 정보로 메일을 동기 방식으로 전송하고 메일 전송 성공 여부를 가져옵니다.	
Argument	*aliasName*	앨리어스 이름
	from	발신자 메일 주소
	fromUserName	발신자 이름 값을 null 또는 ""로 설정할 경우 발신자 메일 주소로 적용됨
	to	수신자 메일 주소
	cc	참조 메일 주소 값을 null 또는 ""로 설정할 경우 참조 메일 주소로 적용되며, 여러 값을 설정할 경우 "," 또는 ";"를 구분자로 설정
	bcc	숨은 참조 메일 주소 값을 null 또는 ""로 설정할 경우 숨은 참조 메일 주소로 적용되며, 여러 값을 설정할 경우 "," 또는 ";"를 구분자로 설정
	subject	메일 제목
	context	메일 내용
	isHTML	메시지 형식을 HTML로 설정할지 여부
	localFileFullPath	파일을 첨부할 경우 첨부할 파일의 전체 경로

	여러 값을 설정할 경우 "|"를 구분자로 설정
fileName	파일을 첨부할 경우 첨부 파일명 ※ 주의사항 : 첨부 파일의 구분자인 "|" 문자열은 파일명에 포함될 수 없습니다.

Option

■ 메일 설정

Key	Value	설명
active	true/false	메일 전송 기능 사용 여부 ex) setProperty("active", "true")
fromSend	발신자 메일 주소	발신자 메일 주소 ex) setProperty("fromSend", "mail@forcs.com")
toSend	수신자 메일 주소	수신자 메일 주소 ex) setProperty("toSend", "mail@forcs.com")
SMTPServer	SMTP 서버 URL	SMTP 서버 URL ex) setProperty("SMTPServer", "mail.forcs.com")
SMTPServerPort	SMTP 서버 포트 번호	SMTP 서버 포트 번호 ex) setProperty("SMTPServerPort", "25")
SMTPUserID	SMTP 사용자 ID	SMTP 인증이 필요한 메일 서버를 이용할 때 인증할 사용자 ID ex) setProperty("SMTPUserID", "UserID")
SMTPUserPassword	SMTP 사용자 ID의 패스워드	SMTP 인증이 필요한 메일 서버를 이용할 때 인증할 사용자 ID의 패스워드 ex) setProperty("SMTPUserPassword", "Password")
SMTPUserID_encrypted	암호화된 SMTP 사용자 ID	SMTP 인증이 필요한 메일 서버를 이용할 때 인증할 암호화된 사용자 ID ex) setProperty("SMTPUserID_encrypted", "ScPdRcRgFgHdEbJaJbPcFc")
SMTPUserPassword_encrypted	암호화된 SMTP 사용자 ID의 패스워드	SMTP 인증이 필요한 메일 서버를 이용할 때 인증할 암호화된 사용자 패스워드 ex) setProperty("SMTPUserPassword_encrypted", "ScPdRcRgFgEbGaDbKbFbMaIbPa")

EnableSSL	true/false	SSL 사용 여부 ex) setProperty("EnableSSL", "true")
SendRetryCount	재전송 횟수	메일 전송 실패 시 재전송 횟수 ex) setProperty("SendRetryCount", "3")
SendRetryPeriodTime	재전송 주기 (단위 : 초)	메일 전송 실패 시 재전송 주기 ex) setProperty("SendRetryPeriodTime", "10")
PrefixSubjectMessage	메일 제목 접두어	메일 제목 앞에 사용할 접두어 ex) setProperty("PrefixSubjectMessage", "[Forcs]")

Sample : MailSample.java

```
package sample;

import oz.framework.api.Mail;
import oz.util.SortProperties;

public class MailSample {

  public static void main(String[] args) throws Exception {

    Mail mail = new Mail("127.0.0.1", 8003, "admin", "admin", true, false);

    SortProperties props = new SortProperties();
    props.setProperty("active", "true");
    props.setProperty("fromSend", "forcs@forcs.com");
    props.setProperty("toSend", "oz@forcs.com");
    props.setProperty("SMTPServer", "smtp.gmail.com");
    props.setProperty("SMTPServerPort", "465");
    props.setProperty("SMTPUserID", "forcs@forcs.com");
    props.setProperty("SMTPUserPassword", "UserPassword");
    props.setProperty("EnableSSL",  "true");

    mail.addAlias("forcs", props);

    mail.sendSync("forcs", "forcs@forcs.com", "User Name", "oz@gmail.com",
null, null, "Mail Subject", "Mail Contextt", false, "d:\\parameter_test.ozd",
"test.ozd");

    props = mail.getAliasConfig("forcs");
    mail.modifyAlias("forcs", "forcs2", props);
```

```
    mail.addAlias("forcs", props);
    mail.addAlias("forcs4", props);
    mail.addAlias("forcs3", props);

    mail.removeAlias("forcs3");

    String[] names = mail.getMailAliasNames();
    for(int i = 0; i < names.length; i++) {
      SortProperties aliasProps = mail.getAliasConfig(names[i]);
      java.util.Iterator propNames = aliasProps.propertyNames().iterator();
      while (propNames.hasNext()) {
        String propName = (String) propNames.next();
        System.out.println(propName + "=" + aliasProps.getProperty(propName));
      }
    }
  }
}
```

Class Module

Constructor Summary

- **Module(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Module(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **void addApplicationParameter(String key, String value)**
- **void addODIParameter(String odiName, String key, String value)**
- **void addODIParameter(String odiName, String item, String category, Hashtable paramHash)**
- **void addParameter(String key, String value)**
- **IReportInfo addReport(String itemName, String categoryName) throws OZCPException**
- **IReportInfo addReport(String itemName, String categoryName, String displayName) throws OZCPException**
- **OZParameter[] getODIParameter(String category, String odiName) throws OZCPException**
- **InputStream getOZD() throws OZCPException**
- **Inputstring getOZD(String item, String category, String[] urls)**
- **OZODIItem[] getOZQReportParameter(String category, String ozqItemName) throws OZCPException**
- **OZReportParameter getOZReportParamter(String category, String reportName) throws OZCPException**
- **InputStream getOZU(String item, String category, String[] urls)**
- **void registODIPath(String odiName, String path)**
- **void saveOZD(String fileName) throws OZCPException**
- **void saveOZD(String fileName, String item, String category, String[] urls)**

- **void saveOZU(String fileName, String item, String category, String[] urls)**
- **void setMemoAllowed(boolean allowed)**
- **void setPassword(String password)**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우**
	`public Module(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)`
	//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우
	`public Module(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	

Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

- **addApplicationParameter**

Prototype	`public void addApplicationParameter(String key, String value)`

Definition	SDM 파일을 만들기 위해 애플리케이션의 패러미터 값을 설정합니다. 현재 애플리케이션에서 설정할 수 있는 패러미터는 ODI 패러미터만 있으므로 ODI에 설정되어 있는 ODI 패러미터 값을 설정하시기 바랍니다.	
Argument	*key*	ODI 패러미터 이름
	value	ODI 패러미터 값

■ **addODIParameter**

Prototype	`public void addODIParameter(String odiName, String key, String value)`	
Definition	SDM 파일을 만들기 위해 ODI에 지정되어 있는 ODI 패러미터 값을 설정합니다. ODI 패러미터 값을 설정하지 않을 경우에는 ODI 기본 패러미터 기본값이 사용됩니다.	
Argument	*odiName*	ODI 이름
	key	ODI 패러미터 이름
	value	ODI 패러미터 값

Prototype	`public void addODIParameter(String odiName, String item, String category, Hashtable paramHash) throws IllegalArgumentException`	
Definition	SDM 파일을 만들기 위해 ODI에 지정되어 있는 ODI 패러미터 값을 설정합니다. ODI 패러미터 값을 설정하지 않을 경우에는 ODI 기본 패러미터 기본값이 사용됩니다. ODI 패러미터 값에 따라 각각 다른 SDM 파일이 필요할 경우 각각의 패러미터 값 별로 SDM 생성 단위를 만들고자 할 때 사용합니다.	
Argument	*odiName*	디자이너에서 지정한 ODI 이름
	item	ODI 파일명
	category	ODI 파일 경로
	paramHash	패러미터 Key, Value 가 설정되어 있는 Hashtable

※ 주의사항 : OZU 생성시 paramHash 주의사항

OZU를 생성할 때 addODIParameter()의 paramHash값에 null을 넣을 경우에는 반드시 addApplicationParameter(Key,Value)에 ODI 패러미터를 설정해 주어야 합니다.

ex) addApplicationParameter를 이용한 패러미터 설정 예

```
module.addApplicationParameter("odi.odinames", "sample");
module.addApplicationParameter("odi.sample.pcount", "1");
module.addApplicationParameter("odi.sample.args1", "deptid=501");
```

■ **addParameter**

Prototype	`public void addParameter(String key, String value)`
Definition	SDM 파일을 만들기 위해 폼에 지정되어 있는 폼 패러미터 값을 설정합니다. 폼 패러미터 값을 설정하지 않을 경우에는 폼 패러미터 기본값이 사용됩니다.
Argument	*key* 폼 패러미터 이름
	value 폼 패러미터 값

■ **addReport**

Prototype	`public IReportInfo addReport(String itemName, String categoryName) throws OZCPException`
Definition	다중 보고서를 하나의 OZD 파일로 만들기 위하여 리포트를 추가합니다.
Argument	*itemName* 리포트를 추가할 아이템 이름
	categoryName 리포트를 추가할 아이템의 카테고리 이름

Prototype	`public IReportInfo addReport(String itemName, String categoryName, String displayName) throws OZCPException`
Definition	다중 보고서를 하나의 OZD 파일로 만들기 위하여 리포트를 추가합니다.
Argument	*itemName* 리포트를 추가할 아이템 이름
	categoryName 리포트를 추가할 아이템의 카테고리 이름
	displayname 뷰어의 보고서 트리에 표시할 보고서 이름

■ **getODIParameter**

Prototype	`public OZParameter[] getODIParameter(String category, String odiName) throws OZCPException`
Definition	ODI 파일에 추가된 ODI 패러미터를 가져옵니다.
Argument	*category* 카테고리 이름
	odiName ODI 이름

■ **getOZD**

Prototype	`public InputStream getOZD() throws OZCPException`
Definition	다중 보고서를 갖는 OZD를 생성한 후 입력 스트림을 가져옵니다.

Prototype	`public InputStream getOZD(String item, String category, String[] urls) throws OZCPException`

Definition	첫 번째 타입의 함수는 서버로부터 보고서 파일을 가져와 OZD 파일을 생성하여 가져오는 함수로, OZD 파일을 생성할 때 urls에 지정된 이미지 파일도 함께 저장합니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에 serverDMType을 설정하지 않는 함수를 사용하거나 또는 serverDMType을 설정하지 않으면 DM_TYPE="Memory", FetchType="Batch"로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다.	
Argument	*item*	아이템 이름 (보고서 파일인 OZR 파일 이름)
	category	카테고리 이름
	urls	OZD에 함께 저장할 이미지 파일의 URL

■ getOZQReportParameter

Prototype	`public OZODIItem[] getOZQReportParameter(String category, String ozqItemName) throws OZCPException`
Definition	OZQR 파일에 추가된 ODI 패러미터를 가져옵니다.
Argument	*category* 카테고리 이름
	ozqItemName OZQR 이름

■ getOZReportParamter

Prototype	`public OZReportParameter getOZReportParamter(String category, String reportName) throws OZCPException`
Definition	리포트의 ODI 패러미터와 폼 패러미터를 가져옵니다. ※ 주의사항 : FX 데이터에 추가된 ODI 패러미터는 가져오지 않습니다.
Argument	*category* 카테고리 이름
	reportName 리포트 이름

■ getOZU

Prototype	`public InputStream getOZU(String item, String category, String[] urls) throws OZCPException`

Definition	서버로부터 애플리케이션 파일과 SDM 파일을 가져와 OZU 파일을 생성하여 가져옵니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에는 항상 DM_TYPE="Momory", FetchType="Batch"로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다. ※ 주의사항 : "FetchUnit"은 반드시 "DM_PER_DATAMODULE"로 설정하여야 합니다.	
Argument	*item*	아이템 이름 (애플리케이션 파일인 OZA 파일 이름)
	category	카테고리 이름
	urls	OZU에 함께 저장할 이미지 파일의 URL

■ **registODIPath**

Prototype	`public void registODIPath (String odiName, String path) throws IllegalArgumentException`	
Definition	OZD 생성시 동적으로 변경할 ODI파일을 등록합니다.	
Argument	`odiName`	ODI 이름
	`path`	ODI 파일 경로

■ **saveOZD**

Prototype	`public void saveOZD(String fileName) throws OZCPException`	
Definition	다중 보고서를 갖는 OZD를 생성한 후 특정 경로에 OZD 파일을 저장합니다.	
Argument	*filename*	저장할 카테고리 경로 및 파일 이름

Prototype	`public void saveOZD(String fileName, String item, String category, String[] urls) throws OZCPException`	
Definition	보고서 파일을 OZD 파일로 저장합니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에는 항상 DM_TYPE="Momory", FetchType="Batch"으로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다.	
Argument	*fileName*	저장 경로를 포함한 OZD 파일명 설정한 저장 폴더가 없을 경우에는 자동으로 폴더 생성함
	item	보고서 파일(.ozr) 이름
	category	보고서 파일(.ozr)의 카테고리 이름
	urls	OZD에 함께 저장할 이미지 파일의 URL

■ **saveOZU**

Prototype	`public void saveOZU(String filename, String item, String category, String[] urls) throws OZCPException`	
Definition	애플리케이션 파일을 OZU 파일로 저장합니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에는 항상 DM_TYPE="Momory", FetchType="Batch"으로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다. ※ 주의사항 : "FetchUnit"은 반드시 "DM_PER_DATAMODULE"로 설정하여야 합니다.	
Argument	*fileName*	저장 경로를 포함한 OZU 파일명 설정한 저장 폴더가 없을 경우에는 자동으로 폴더 생성함
	item	애플리케이션 파일(.oza) 이름
	category	애플리케이션 파일(.oza)의 카테고리 이름
	urls	OZU에 함께 저장할 이미지 파일의 URL

■ **setMemoAllowed**

Prototype	`public void setMemoAllowed(boolean allowed)`	
Definition	메모 허용 여부를 설정합니다.	
Argument	*allowed*	허용여부

■ **setPassword**

Prototype	`public void setPassword(String password)`	
Definition	패스워드를 설정합니다.	
Argument	*password*	패스워드

관련 Class

■ **OZReportParameter(oz.framework.odm.OZReportParameter.class)**

리포트의 ODI 패러미터와 폼 패러미터 정보를 가지고 있는 클래스입니다.

- 메소드
  - getODIList

Prototype	`public OZODIItem[] getODIList()`
Definition	ODI 정보 객체를 가져옵니다.

Return	*OZODIItem* ODI 정보 객체

- getFormParameters

Prototype	`public OZParameter[] getFormParameters()`
Definition	폼 패러미터 정보 객체를 가져옵니다.
Return	*OZParameter* Form Key, Value, Type, Desc 값이 저장된 객체

■ **OZODIItem(oz.framework.odm.OZODIItem.class)**

ODI 패러미터 정보를 가지고 있는 클래스입니다.

- 메소드

- getCategory

Prototype	`public String getCategory()`
Definition	카테고리 이름을 가져옵니다.

- getODIName

Prototype	`public String getODIName()`
Definition	ODI 이름을 가져옵니다.

- getODIFileName

Prototype	`public String getODIFileName()`
Definition	ODI 파일 이름을 가져옵니다.

- getODIParameters

Prototype	`public OZParameter[] getODIParameters()`
Definition	해당 ODI 패러미터를 가져옵니다.
Return	*OZParameter* ODI Key, Value, Type, Desc 값이 저장된 객체

■ **OZParameter(oz.framework.odm.OZParameter.class)**

패러미터 정보를 가지고 있는 클래스입니다.

- 메소드

- getName

Prototype	`public String getName()`
Definition	패러미터 이름을 가져옵니다.

- getType

Prototype	`public int getType()`
Definition	패러미터 타입을 가져옵니다. 패러미터 타입은 JDBC Types과 동일합니다. 예를 들어, VARCHAR 타입인 경우 12가 리턴됩니다.

- getValue

Prototype	`public String getValue()`
Definition	패러미터 값을 가져옵니다.

- getDescription

Prototype	`public String getDescription()`
Definition	패러미터 설명을 가져옵니다.

관련 Interface

■ **IReportInfo(oz.framework.api.IReportInfo)**

다중 보고서를 갖는 OZD 생성 시 보고서에 대한 부가 정보를 설정합니다.

– **getItemName**

Prototype	`public String getItemName()`
Definition	아이템 이름을 가져옵니다.

– **getCategoryName**

Prototype	`public String getCategoryName()`
Definition	아이템의 카테고리 이름을 가져옵니다.

– **addURL**

Prototype	`public void addURL(String[] urls)`
Definition	보고서에 포함된 이미지 파일의 URL을 배열로 설정합니다.
Argument	*urls* 이미지 파일 URL

Prototype	`public void addURL(String url)`
Definition	보고서에 포함된 이미지 파일의 URL을 설정합니다.

Argument	*url*	이미지 파일 URL

- **addParameter**

Prototype	public void addParameter(String key, String value)	
Definition	패러미터 값을 설정합니다.	
Argument	*key*	패러미터 이름
	value	패러미터 값

- **addODIParameter**

Prototype	public void addODIParameter(String odiName, String key, String value)	
Definition	ODI 패러미터 값을 설정합니다.	
Argument	*odiName*	ODI 이름
	key	ODI 패러미터 이름
	value	ODI 패러미터 값

- **addProperties**

Prototype	public void addProperties(String key, String value)	
Definition	보고서 저장 옵션을 설정합니다.	
Argument	*key*	옵션 이름
	value	옵션 값 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

- **addFrameworkURL**

Prototype	public void addFrameworkURL(String fxName, String url, String charset)	
Definition	FX 데이터의 프레임웍 URL을 설정합니다.	
Argument	*fxName*	FX 데이터 이름
	url	프레임웍 URL
	charset	프레임웍 URL의 패러미터를 처리할 문자 셋 ※ 참고사항 • 문자 셋을 null 또는 공백으로 설정한 경우 시스템의 기본 문자 셋으로 적용됩니다. • 문자 셋에 base64(대소문자 구분 안 함)를 포함하여 설정한 경우 예를 들어,

base64/ksc5601 또는 base64/utf-8 등의 형태로 설정한 경우 ksc5601 또는 utf-8로 인코딩한 후 base64로 한 번 더 인코딩합니다.

Sample : ModuleSample.java

```
package sample;

import java.io.*;
import oz.framework.api.Module;
import org.apache.log4j.*;

public class ModuleSample {
   public static void main(String[] args) {
//아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      // OZServer Info.
      //*
      // Daemon
      String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 8003; //서버가 사용하는 TCP 포트
      /**/
      // User Info.
      String ID = "admin"; //default
      String PWD = "admin"; //default
      Module module = null;
      InputStream stream = null;
      try {
         //*
         // Daemon
         module = new Module(IP, PORT, ID, PWD, false, false);
         // ------------------------------------------------------------------
         // OZD 생성시 이미지 같이 저장
         // set form parameters
         module.addParameter("formparam1", "ABCD1");
         module.addParameter("formparam2", "EFGH2");
         // set odi parameters
         module.addODIParameter("parameter_test", "odiparam1", "IJKL1");
         module.addODIParameter("parameter_test", "odiparam2", "LMNO2");
         module.setPassword("1234");
         module.setMemoAllowed(true);
         String[] url = new String[0];
         module.saveOZD("C:/parameter_test.ozd", "parameter_test.ozr", "/",
url);
```

```
      }
      catch(Exception e) {
         e.printStackTrace();
      }
   }
}
```

Sample : ModuleOZUSample.java

```
package sample;

import oz.framework.api.Module;
import org.apache.log4j.*;

public class ModuleOZUSample{
  public static void main(String[] args){
//아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
   BasicConfigurator.configure();

   // OZServer Info.
   String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
   int PORT = 8003; //서버가 사용하는 TCP 포트
   // User Info.
   String ID = "admin"; //default
   String PWD = "admin"; //default
   Module module = null;
   try{
     module = new Module(IP, PORT, ID, PWD, false, /*usl*/ false);
     //ozu 를 생성하기 위해서는 반드시 설정해야 할 파라메터
     module.addApplicationParameter("odi.fetchunit", "DM_PER_DATAMODULE");
     module.addApplicationParameter("odi.odinames", "parameter_test");
     module.addApplicationParameter("odi.parameter_test.pcount", "2");
     module.addApplicationParameter("odi.parameter_test.args1",
"odiparam1=ABCD");
     module.addApplicationParameter("odi.parameter_test.args2",
"odiparam2=1234");

     String[] urls = new String[0];

     module.addODIParameter("parameter_test", "parameter_test.odi", "/",
null);

     //ozu 저장할 파일이름, oza 파일이름, 카테고리, 해당 URL
     module.saveOZU("C:/parameter_test.ozu", "parameter_test.oza", "/",
urls);
```

```
   }
   catch(Exception e){
      e.printStackTrace();
   }
  }
}
```

Sample : MultiReportSample.java

```
package sample;

import org.apache.log4j.*;
import oz.framework.api.*;

public class MultiReportSample{

   public static void main(String[] args) throws Exception{

      BasicConfigurator.configure();

      Module maker = new Module("127.0.0.1", 8003, "admin", "admin", true,
false);

      String[] urls = new String[0];

      oz.framework.api.IReportInfo reportInfo =
maker.addReport("parameter_test.ozr", "/", "Report1");
      reportInfo.addURL(urls);
      reportInfo.addODIParameter("parameter_test.odi", "odiparam1", "form
number 11");
      reportInfo.addODIParameter("parameter_test.odi", "odiparam2", "form
number 12");
      reportInfo.addParameter("formparam1", "form parameters 11");
      reportInfo.addParameter("formparam2", "form parameters 12");

      reportInfo = maker.addReport("parameter_test.ozr", "/", "Report2");
      reportInfo.addURL(urls);
      reportInfo.addODIParameter("parameter_test.odi", "odiparam1", "form
number 21");
      reportInfo.addODIParameter("parameter_test.odi", "odiparam2", "form
number 22");
      reportInfo.addParameter("formparam1", "form parameters 21");
      reportInfo.addParameter("formparam2", "form parameters 22");
```

```
      reportInfo = maker.addReport("parameter_test.ozr", "/", "Report3");
      reportInfo.addURL(urls);
      reportInfo.addODIParameter("parameter_test.odi", "odiparam1", "form
number 31");
      reportInfo.addODIParameter("parameter_test.odi", "odiparam2", "form
number 32");
      reportInfo.addParameter("formparam1", "form parameters 31");
      reportInfo.addParameter("formparam2", "form parameters 32");

      maker.saveOZD("c:\\MultiReport.ozd");
    }
}
```

UseOzdParameterSample.java

```
package sample;

import oz.framework.api.*;

public class UseOzdParameterSample {

   public static void main(String[] args) throws Exception{

      oz.framework.api.Module          module          =          new
oz.framework.api.Module("127.0.0.1",8003, "admin", "admin");

      oz.framework.api.IReportInfo             reportInfo             =
module.addReport("parameter_test.ozr", "/", "1page");
      reportInfo.addODIParameter("parameter_test", "odiparam1", "A1");
      reportInfo.addODIParameter("parameter_test", "odiparam2", "B2");

      reportInfo.addParameter("formparam1","C3");
      reportInfo.addParameter("formparam2","D4");

      reportInfo.addProperties("use_ozd_parameter", "false");

      reportInfo    =    module.addReport("parameter_test.ozr",    "/",
"2page");
      reportInfo.addODIParameter("parameter_test", "odiparam1", "E5");
      reportInfo.addODIParameter("parameter_test", "odiparam2", "F6");

      reportInfo.addParameter("formparam1","G7");
      reportInfo.addParameter("formparam2","H8");
```

```
        reportInfo.addProperties("use_ozd_parameter", "false");

        module.saveOZD("c:/daemon_api_OZD.ozd");
        System.out.println("ok");
    }
}
```

Class Monitor

Constructor Summary

- **Monitor(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Monitor (String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **Versions getVersions()**
- **MemoryStatus getMemoryInfo()**
- **byte[] downloadMonitorLog()**
- **void downloadMonitorLog(String fileName)**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우**
	`public Monitor(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)`
	//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우
	`public Monitor(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	

Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;

id	사용자 아이디 ex) String id = "admin";
pw	사용자 패스워드 ex) String pw = "admin";
bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

- **getVersions**

Prototype	`public Versions getVersions() throws OZCPException`
Definition	서버의 버전 정보와 각 모듈별 버전 정보를 가져옵니다.

- **getMemoryInfo**

Prototype	`public MemoryStatus getMemoryInfo() throws OZCPException`
Definition	현재의 메모리 사용량(전체 메모리, 사용 메모리, 비사용 메모리) 정보를 가져옵니다.

- **downloadMonitorLog**

Prototype	`public byte[] downloadMonitorLog() throws OZCPException`
Definition	서버에 저장된 모니터 로그 파일을 가져옵니다.

- **downloadMonitorLog(String fileName)**

Prototype	`public void downloadMonitorLog(String fileName) throws OZCPException, IOException`
Definition	서버에 저장된 모니터 로그 파일을 가져와서 fileName으로 지정된 위치에 로그 파일을 저장합니다.

관련 Class

■ **MemoryStatus(oz.server.monitor.MemoryStatus)**

Server가 실행되고 있는 System의 메모리 상태를 나타냅니다.

- 멤버 변수
  - public static void garbageCollect() : garbage collection 수행
  - public long getTotalMemory() : 서버 VM의 토탈 메모리
  - public long getFreememory() : 서버 VM의 Free 메모리

■ **Versions(oz.server.monitor.Versions)**

Server와 Server가 실행되고 있는 System의 버전 정보들을 나타냅니다.

- 멤버 변수
  - public String osName : Server가 실행되고 있는 OS의 이름
  - public String osVersion : Server가 실행되고 있는 OS의 버전
  - public String javaVendor : Server가 실행되고 있는 JVM의 제작 회사
  - public String javaVersion : Server가 실행되고 있는 JVM의 Version
  - public String OZServerVersion : 오즈 서버의 버전
  - public String CPRelease : OZ Common Protocol의 릴리즈 정보
  - public int CPProtocol : OZ Common Protocol의 프로토콜 버전
  - public String DMRelease : OZ Data Module의 릴리즈 정보
  - public int DMStreaming : OZ Data Module의 Streaming 프로토콜 버전

Sample : MonitorSample.java

```
package sample;

import oz.framework.api.Monitor;
import oz.server.monitor.Versions;
import oz.server.monitor.MemoryStatus;
import org.apache.log4j.*;

public class MonitorSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
       BasicConfigurator.configure();

       // OZServer Info.
       //*
       // Daemon
       String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
```

```
        int PORT = 8003; //서버가 사용하는 TCP 포트
        /*/
        // Servlet
        String URL  = "http://www.oz.com/oz/server"; //Servlet에 접근 가능한 URL
        /**/
        // User Info.
        String ID = "admin"; //default
        String PWD = "admin"; //default

        Monitor monitor = null;
        try {
            //*
            // Daemon
            monitor = new Monitor(IP, PORT, ID, PWD, false, false);
            /*/
            // Servlet
            monitor = new Monitor(URL, ID, PWD, false, false);
            /**/

            // 서버와 모듈별 버전 정보(getServerInformation)
           Versions v = monitor.getVersions();
           v._printOut();

           // 메모리 사용량 확인(getServerStatus)
           MemoryStatus ms = monitor.getMemoryInfo();

           // 사용메모리 = 전체메모리 - 비사용메모리
           long usedMemory = ms.getTotalMemory() - ms.getFreeMememoy();

           // 가져온 서버 메모리 사용량 상태값
           System.out.println("Total Memory="+ms.getTotalMemory());
           System.out.println("Used Memory="+usedMemory);
           System.out.println("Free Memory="+ms.getFreeMememoy());
           System.out.println("");

           // 모니터 로그를 내려 받습니다 .
           byte [] logBytes = monitor.downloadMonitorLog();

           // 모니터 로그를 지정하는 파일로 저장합니다.
           String logFileName = "monitor.log";
           monitor.downloadMonitorLog(logFileName);

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
```

```
}
```

Class Service

Constructor Summary

- **Service(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Service(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **void garbageCollect()**
- **void stop(boolean check)**
- **void restart()**
- **boolean ping()**
- **int getHandlerCount()**

Constructor Detail

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우**
	`public Service(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)`
	//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우
	`public Service(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)`	

Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";

	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ garbageCollect

Prototype	`public void garbageCollect() throws OZCPException`
Definition	별도의 메모리 수집 프로그램을 실행하여 JVM 상에서 사용되지 않는 메모리를 수거해 재사용 가능하게 해주는 garbage collection을 실행합니다.

■ stop

Prototype	`public void stop(boolean check) throws OZCPException`	
Definition	동작 중인 서버를 정상 종료시킵니다.	
Argument	*check*	강제 종료 여부 • true : 서버를 바로 종료 • false : 동작중인 작업이 완료된 후 서버를 종료

■ restart

Prototype	`public void restart() throws OZCPException`
Definition	동작 중인 서버를 정상 종료시키고 다시 실행합니다.

■ ping

Prototype	`public boolean ping()`
Definition	서버의 구동 여부를 반환합니다.

■ **getHandlerCount**

Prototype	public int getHandlerCount()
Definition	서버의 동시 실행수를 반환합니다.

Sample : ServiceSample.java

```
package sample;

import oz.framework.api.Service;
import org.apache.log4j.*;

public class ServiceSample {
        public static void main(String[] args) {
                // 아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지
않습니다.
                BasicConfigurator.configure();
                // OZServer Info.
                // *
                // Daemon
                String IP = "127.0.0.1"; // 서버가 설치되어 있는 호스트 컴퓨터의 IP
                int PORT = 8003; // 서버가 사용하는 TCP 포트
                String URL = "http://localhost:8080/OZServlet/server";


        /**********************************************************************
***
                 * / // Servlet String URL = "http://www.oz.com/oz/server";
//Servlet 에
                 * 접근가능한 URL /

**********************************************************************/

                // User Info.
                String ID = "admin"; // default
                String PWD = "admin"; // default
                Service service = null;
                try {
                        // *
                        // Daemon
                        service = new Service(IP, PORT, ID, PWD, false,
false);
                        //service = new Service(URL, ID, PWD, false, false);
```

```
/*****************************************************************
                        * / // Servlet service = new Service(URL, ID, PWD,
false, false); /

*****************************************************************/

                        // ping 으로 서버 동작 여부 확인
                        boolean isAlive = service.ping();

                        //서버 구동
                        if(isAlive == true) {
                                System.out.println("Server is Running");

                                // 서버의 핸들러의 갯수

        System.out.println(service.getHandlerCount());

                                // garbageCollection 실행(garbageCollects)
                                service.garbageCollect();

                                // 서버 재시작 (restart)
                                // service.restart();

                                // 서버 종료(serverStop)
                                boolean check = true;//강제종료여부결정
                                service.stop(check);

                                // 서버가 정상적으로 종료되었는지 검사 한다.
                                isAlive = true;
                                while (isAlive) {
                                        isAlive = service.ping();
                                }
                                if(isAlive == false) {
                                        System.out.println("Server is
stoped");
                                }
                        }
                        //서버 구동 중지
                         else {
                                System.out.println("Server is dead");
                        }

                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

Class Viewer

Constructor Summary

- **Viewer(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL)**
- **Viewer(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)**

Method Summary

- **byte[] getByteArrayForm(String reportName, String categoryName)**
- **InputStram getForm(String reportName, String CategoryName, boolean isCompress)**
- **Parameter[] getUserParametersWithDefaultValue(String itemName, int itemType, String categoryName)**
- **HCDataModule getHCUSDM(String fileName, String categoryName)**
- **HCDataModule getDataModule(InputStream sdmInput)**
- **HCDataModule getDataModule(String odiname, String CategoryName, Parameter[] parameters, boolean doCompress, boolean forceRefresh)**
- **HCDataModule getDataModule(String odiName, String categoryName, Parameter[] parameters, boolean doCompress, boolean forceRefresh, String[] invalidDataset, MaxRowsOfSet[] maxRows)**
- **HCDataModule getDataModule(String odiName, String categoryName, Parameter[] parameters, boolean doCompress, boolean forceRefresh, boolean isPushServer)**

Constructor Detail

Prototype	//**Daemon** 타입 - 오즈 서버 타입이 **TCP Server**인 경우 public Viewer(String ip, int port, String id, String pw, boolean bAutoLogin, boolean useUSL) //**Servlet** 타입 - 오즈 서버 타입이 **HTTP Server**인 경우 public Viewer(String url, String id, String pw, boolean bAutoLogin, boolean useUSL)	
Argument	*url*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	ip	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	port	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	id	사용자 아이디 ex) String id = "admin";
	pw	사용자 패스워드 ex) String pw = "admin";
	bAutoLogin	자동 로그인 여부 ex) boolean bAutoLogin = true;
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ getByteArrayForm

Prototype	public byte[] getByteArrayForm(String reportName, String categoryName) throws OZCPException	
Definition	리포트 폼의 내용을 가져옵니다.	
Argument	*reportName*	리포트 폼 이름
	categoryName	리포트의 카테고리명

■ getForm

Prototype	public InputStream getForm(String reportName, String categoryName, boolean isCompress) throws OZCPException

Definition	리포트 폼의 내용을 압축 여부를 설정하여 가져옵니다.	
Argument	*reportName*	리포트 폼 이름
	categoryName	리포트의 카테고리명
	isCompress	압축 여부

■ getUserParametersWithDefaultValue

Prototype	`public Parameter[] getUserParametersWithDefaultValue(String itemName, int itemType, String categoryName) throws OZCPException`	
Definition	아이템에서 사용되는 패러미터와 패러미터의 기본값을 알려줍니다.	
Argument	*itemName*	아이템 이름
	itemType	아이템의 종류 (OZR : 20001 / ODI : 10000)
	categoryName	아이템의 카테고리명
	isCompress	압축 여부

■ getHCUSDM

Prototype	`public HCDataModule getHCUSDM(String fileName, String categoryName) throws OZCPException`	
Definition	서버에 저장한 SDM을 데이터로 가져옵니다.	
Argument	*fileName*	리포트 폼 이름
	categoryName	리포트의 카테고리명

■ getDataModule

Prototype	`public HCDataModule getDataModule(InputStream sdmInput) throws OZCPException`	
Definition	SDM으로부터 데이터 레코드를 가져옵니다. FETCH 형식은 배치(batch) 타입으로, DataModule 바인딩은 2.5 방식의 Normal로 고정됩니다.	
Argument	*sdmInput*	SDM 스트림

■ getDataModule

Prototype	public HCDataModule getDataModule(String odiName, String categoryName, Parameter[] parameters, boolean doCompress,boolean forceRefresh) throws OZCPException public HCDataModule getDataModule(String odiName, String categoryName, Parameter[] parameters, boolean doCompress, boolean forceRefresh, String[] invalidDataset, MaxRowsOfSet[] maxRows) throws OZCPException public HCDataModule getDataModule(String odiName, String categoryName, Parameter[] parameters, boolean doCompress, boolean forceRefresh, boolean isPushServer) throws OZCPException
Definition	선택한 데이터 레코드를 가져옵니다.
Argument	*odiName* 데이터 소스에 대한 정보 파일명
	categoryName 데이터 소스 파일의 카테고명
	parameters 데이터 레코드를 가져오기 위한 사용자 패러미터
	doCompress 압축 전송 여부
	forceRefresh 서버의 캐시 사용 여부
	isPushServer 푸쉬 서버 여부
	invalidDataset 데이터 소스 중 사용하지 않는 데이터 셋 이름
	maxRows 데이터 셋의 최대 레코드 수 정보

관련 Class

■ **Parameter(oz.dm.Parameter)**

패러미터 이름과 값에 대한 정보를 가지고 있는 클래스입니다.

- 멤버 변수
  - public String name : 패러미터 이름
  - public String value : 패러미터 값

■ **MaxRowsOfSet (oz.dm. MaxRowsOfSet)**

데이터 셋의 최대 레코드 수를 제한하는 정보를 가지고 있는 클래스입니다.

- 메소드
  - SetSetName

Prototype	public void SetSetName(String v)

Definition	최대 레코드 수를 설정할 데이터 셋 이름을 설정합니다.

- GetSetName

Prototype	`public String GetSetName()`
Definition	최대 레코드 수가 설정되어 있는 데이터 셋 이름을 가져옵니다.

- SetMaxRow

Prototype	`public void SetMaxRow(int v)`
Definition	최대 레코드 수를 설정합니다.

- GetMaxRow

Prototype	`public int GetMaxRow()`
Definition	설정되어 있는 최대 레코드 수를 가져옵니다.

■ **HCDataModule(oz.dm.hc.HCDataModule)**

HCDataSet의 모음입니다. HCDataSet은 BCDataSet이 가지고 있는 버퍼를 OZ DataModule Streaming 규약에 맞추어 파싱하여 데이터를 추출하는 역할을 합니다.

- 멤버 변수
  - public final static String PARAMETERSET_NAME = "OZParam"; : 오즈 패러미터 선언

- 메소드
  - getDataSetNames

Prototype	`public String[] getDataSetNames()`
Definition	모든 데이터 셋의 이름을 리턴합니다.

  - getDataSets

Prototype	`public HCDataSet[] getDataSets(String name)`
Definition	해당 이름의 데이터 셋을 가져옵니다. 데이터 셋 이름은 대소문자를 구분합니다. 만일 마스터 데이터 셋인 경우에는 하나를 디테일 데이터 셋인 경우에는 여러 개를 가져옵니다. 해당 이름의 데이터 셋이 없으면 길이(length)가 0인 배열을 반환합니다.
Argument	*name* 데이터 셋 이름

- PrintInfo

Prototype	`public void PrintInfo(PrintStream out)`
Definition	데이터와 필드 정보를 PrintStream으로 출력합니다.
Argument	*out* 출력할 PrintStream 객체

■ **HCDataSet(oz.dm.hc)**

HCDataSet은 원본 데이터의 형태와 가장 유사하게 구성되어 있는 OZ DataModule Streaming 규약에 맞는 데이터 형태입니다. 따라서 원본 데이터를 직접 얻어올 수도 있고 각각의 필드와 열에 맞는 데이터의 접근도 가능합니다.

- 멤버 변수
  - final public static int BYTEARRAY_DATA_SET = 0x1;
  - final public static int UTF_DATA_SET = 0x2;
  - final public static String BYTEARRAY_PROTOCOL_NAME = "ByteArraySet";
  - final public static String UTF_PROTOCOL_NAME = "UTFSet";

- 메소드
  - getResultSet

Prototype	`public ResultSet getResultSet()`
Definition	원본 데이터를 결과셋 형태로 가져옵니다.

Sample : ViewerSample.java

```
package sample;

import oz.framework.api.Viewer;
import oz.dm.Parameter;
import oz.dm.hc.HCDataModule;
import oz.dm.MaxRowsOfSet;
import java.io.*;
import org.apache.log4j.*;

public class ViewerSample {
   public static void main(String[] args) {
//아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      // OZServer Info.
```

```
//*
 // Daemon
String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
int PORT = 8003; //서버가 사용하는 TCP 포트
/*/
// Servlet
String URL  = "http://www.oz.com/oz/server"; //Servlet 에 접근가능한 URL
/**/
// User Info.
String ID = "admin"; //default
String PWD = "admin"; //default

Viewer viewer = null;
try {
   //*
   // Daemon
   viewer = new Viewer(IP, PORT, ID, PWD, false, false);
   /*/
   // Servlet
   viewer = new Viewer(URL, ID, PWD, false, false);
   /**/

   int ODI_FILE_TYPE = 10000; // ODI 파일 타입
   int OZR_FILE_TYPE = 20001; // OZR 파일 타입
   String ozrName = "parameter_test.ozr";
   String odiName = "parameter_test.odi"; //ODI 이름
   String sdmName = "parameter_test.sdm";
   String categoryName = "/"; // 카테고리 이름
   Parameter[] param = new Parameter[0]; // 패러미터
   boolean force = false; // 캐시의 리프레쉬 옵션
   boolean compress = false; // DM 의 압축 전송 여부
   String[] invalidSet = new String[0]; //사용하지 않는 데이터 셋 이름
   MaxRowsOfSet[] mx = new MaxRowsOfSet[0]; //셋들의 레코드 개수제한

   // 사용자가 서버에 설정한 SDM 에서 데이터 가져오기(getHCUSDM)
   HCDataModule usdm = viewer.getHCUSDM(sdmName, categoryName);

   // 서버에서 폼 가져오기. (getByteArrayForm)
   byte[] formBytes = viewer.getByteArrayForm(ozrName, categoryName);

   // 서버에서 폼 가져오기 (getForm)
   InputStream in = viewer.getForm(ozrName, categoryName, compress);

   // 아이템에서 사용하는 패러미터와 기본값을 반환
   //(getUserParametersWithDefaultValue)
   param = viewer.getUserParametersWithDefaultValue(odiName,
      ODI_FILE_TYPE, categoryName);
```

```
            // Input Stream 으로부터 SDM 내용을 읽어 데이터를 획득(getDataModule)
            HCDataModule sdm = viewer.getDataModule(
                new FileInputStream(sdmName));

            // 서버로부터 데이터 레코드를 획득 (getDataModules)
            HCDataModule dModule = viewer.getDataModule(
                odiName, categoryName, param, compress, force);

            // 서버로부터 데이터 레코드를 획득 (getDataModules)
            HCDataModule dModule2 = viewer.getDataModule(
                odiName, categoryName, param,compress, force, invalidSet, mx);

        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Class Servlet

Method Summary

- **InputStream getOZD(String item, String category, Hashtable formparam, Hashtable odiparam, boolean memoallowed, String password, String id, String pw, HttpServletRequest request)**
- **InputStream getOZD(String item, String category, String serverDMType, Hashtable formparam, Hashtable odiparam, Hashtable odipath, boolean memoallowed, String password, String id, String pwd, HttpServletRequest request)**
- **InputStream getOZD(String[] item, String[] category, Hashtable[] formparam, Hashtable[] odiparam, String id, String pwd, HttpServletRequest request, String[] displayName, String[][] urls, Hashtable[] options) throws Exception**
- **InputStream getOZD(String[] item, String[] category, Hashtable[] formparam, Hashtable[] odiparam, boolean memoallowed, String password, String id, String pwd, HttpServletRequest request, String[] displayName, String[][] urls, Hashtable[] options) throws Exception**
- **InputStream getOZD(String[] item, String[] category, String serverDMType, Hashtable[] formparam, Hashtable[] odiparam, boolean memoallowed, String password, String id, String pwd, HttpServletRequest request, String[] displayName, String[][] urls, Hashtable[] options) throws Exception**
- **InputStream getOZD(String[] item, String[] category, String serverDMType, Hashtable[] formparam, Hashtable[] odiparam, Hashtable[] odiPaths, boolean memoallowed, String password, String id, String pwd, HttpServletRequest request, String[] displayName, String[][] urls, Hashtable[] options, Hashtable[] fxURLs) throws Exception**
- **InputStream getOZDWithSDM(String item, String category, Hashtable sdm, Hashtable formParam, boolean memoallowed, String password, String id, String pwd, HttpServletRequest request)**
- **getOZDWithSDM(String item, String category, Hashtable sdm, Hashtable**

formParam, boolean memoallowed, String password, String id, String pwd, HttpServletRequest request)

Method Detail

- **getOZD**

```
public final InputStream getOZD(String item, String
category, Hashtable formparam, Hashtable odiparam, boolean
memoallowed, String password, String id, String pwd,
HttpServletRequest request) throws Exception

public final InputStream getOZD(String item, String
category, String  serverDMType, Hashtable formparam,
Hashtable odiparam, Hashtable odipath, boolean memoallowed,
String password, String id, String pwd, HttpServletRequest
request)

public final InputStream getOZD(String[] item, String[]
category, Hashtable[] formparam, Hashtable[] odiparam,
String id, String pwd, HttpServletRequest request, String[]
displayName, String[][] urls, Hashtable[] options) throws
Exception
```

Prototype

```
public final InputStream getOZD(String[] item, String[]
category, Hashtable[] formparam, Hashtable[] odiparam,
boolean memoallowed, String password, String id, String pwd,
HttpServletRequest request, String[] displayName, String[][]
urls, Hashtable[] options) throws Exception

public final InputStream getOZD(String[] item, String[]
category, String serverDMType, Hashtable[] formparam,
Hashtable[] odiparam, boolean memoallowed, String password,
String id, String pwd, HttpServletRequest request, String[]
displayName, String[][] urls, Hashtable[] options) throws
Exception

public final InputStream getOZD(String[] item, String[]
category, String serverDMType, Hashtable[] formparam,
Hashtable[] odiparam, Hashtable[] odiPaths, boolean
memoallowed, String password, String id, String pwd,
HttpServletRequest request, String[] displayName, String[][]
```

	urls, Hashtable[] options, Hashtable[] fxURLs) throws Exception	
Definition	오즈 서블릿 서버를 상속받아 OZD를 생성하는 오즈 서블릿 서버를 생성하고, 새로 생성된 오즈 서블릿 서버와 통신하여 OZD 파일을 만듭니다. ※ 주의사항 : API 함수를 이용하여 데이터를 바인딩할 경우에 serverDMType을 설정하지 않는 함수를 사용하거나 또는 serverDMType을 설정하지 않으면 DM_TYPE="Memory", FetchType="Batch"로 바인딩됩니다. 대용량 데이터를 바인딩할 경우에는 메모리를 많이 사용하거나 결과물을 얻는데 시간이 걸릴 수 있습니다.	
Argument	*item*	아이템 이름 (보고서 파일인 OZR 파일 이름)
	category	카테고리 이름
	urls	OZD에 함께 저장할 이미지 파일의 URL
	serverDMType	오즈 서블릿 서버에서 데이터 모듈을 받을 때 받은 데이터 모듈을 Memory로 처리할지 File로 처리할지 여부를 설정 (기본값:Memory)
	formparam	폼 패러미터 ※ 주의사항 : 폼 패러미터는 Hashtable에 패러미터명(String), 패러미터 값(String등 패러미터의 데이터형) 형태로 입력하여야 합니다.
	odiparam	ODI 패러미터 ※ 주의사항 : ODI 패러미터는 Hashtable에 ODI명(String), 패러미터값(Hashtable) 형태로 입력하여야 하며, 패러미터값은 다시 Hashtable 형태로 패러미터명(String), 패러미터값(String) 형태로 입력하여야 합니다.
	odipath	ODI 파일 경로
	memoallowed	OZD 문서에 메모 허용 여부 설정
	password	OZD 문서를 열 때 적용할 패스워드
	id	오즈 서블릿 서버에 접속하기 위한 사용자 아이디
	pwd	오즈 서블릿 서버에 접속하기 위한 사용자 패스워드
	request	HttpServletRequest
	displayName	보고서 이름
	urls	이미지 파일 URL
	options	OZD 저장 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

	fxURLs	FX 데이터의 프레임웍 URL과 문자 셋
		※ 참고사항
		• 값을 설정하지 않을 경우 보고서에 설정된 URL과 문자 셋으로 설정됩니다.
		• fxURL의 fxName의 value가 String인 경우에는 url 이며, String[]인 경우 String[0]=프레임웍 URL, String[1]=문자셋으로 설정하여야 합니다.
		ex)
		Hashtable[] fxURLs = new Hashtable[2];
		fxURLs[0] = new Hashtable();
		String[] url = new String[2];
		url[0] = "http://127.0.0.1:8080/ozcar.jsp?param1=포시에스";
		url[1] = "base64/ksc5601";
		fxURLs[0].put("sample",url);

■ getOZDWithSDM

Prototype	`public final InputStream getOZDWithSDM(String item, String category, Hashtable sdm, Hashtable formParam, boolean memoallowed, String password, String id, String pwd, HttpServletRequest request) throws Exception`	
Definition	보고서 파일과 SDM 파일을 가져와 OZD 파일을 생성하여 가져옵니다. ※ 주의사항 - 해당 Servlet API를 웹에서 뷰어 패러미터로 호출하여 사용하는 경우에는 SDM 데이터를 "SDM 데이터 → gzip → Base64 Encode → Encode URI"로 변환하여 설정하여야 합니다. - 서블릿에서 SDM을 패러미터로 전송받을 경우에는 "Base64 Decode → ungzip → SDM 데이터"로 변환하여 전송받아야 합니다. - getOZDWithSDM 함수를 사용할 경우 doGet 함수에서 doPost 함수를 선언한 후 doPost 함수에서 getOZDWithSDM 함수를 호출하는 방식으로 사용하여야 합니다.	
Argument	*item*	아이템 이름 (보고서 파일인 OZR 파일 이름)
	category	카테고리 이름
	sdm	OZD 파일을 만들 SDM 정보
	formparam	폼 패러미터 ※ 주의사항 : 폼 패러미터는 Hashtable에 패러미터명(String), 패러미터 값(String등 패러미터의 데이터형) 형태로 입력하여야 합니다.
	memoallowed	OZD 문서에 메모 허용 여부 설정
	password	OZD 문서를 열 때 적용할 패스워드

	id	오즈 서블릿 서버에 접속하기 위한 사용자 아이디
	pwd	오즈 서블릿 서버에 접속하기 위한 사용자 패스워드
	request	HttpServletRequest

Sample : RequestOZDSample.java

```
package sample;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletConfig;
import oz.server.OZServlet;
import javax.servlet.ServletException;

import java.io.*;
import java.util.*;

import oz.framework.cp.io.OZDataOutputStream;

public class RequestOZDSample extends OZServlet {
   private static final int PROTOCOL_VER = 2005;
   private static final String _ROOT_PATH = "D:/";
   private byte[] _buf = new byte[1024];
   public void init(ServletConfig config) throws ServletException {
      super.init(config);
   }

   public void doGet(HttpServletRequest request,
      HttpServletResponse response) throws ServletException, IOException
   {
      doPost(request, response);
   }

   public void doPost(HttpServletRequest request,
      HttpServletResponse response) throws ServletException, IOException
   {
      try {
         System.out.println("Start to get OZD");
         long _JOB_ID = System.currentTimeMillis();
         String tempOZDFile = _ROOT_PATH + "sample" + _JOB_ID + ".ozd";
         String tempWMFile  = _ROOT_PATH + "sample" + _JOB_ID + ".mtx";
         // Make OZD
         makeOZD(tempOZDFile, request);
      }
```

```
    catch(Exception e) {
        ByteArrayOutputStream bout = new ByteArrayOutputStream();
        e.printStackTrace(new PrintStream(bout));
        bout.flush();
        byte[] b = bout.toByteArray();
        String error = new String(bout.toByteArray());
        bout.close();
        System.out.println(error);
        PrintWriter writer = response.getWriter();
        writer.println("RequestOZDSample : Error");
        writer.println();
        writer.println(error);
        writer.flush();
    }
}

private void writeFile(String filename, OZDataOutputStream out)
    throws IOException
{
    BufferedInputStream bin = null;
    try {
        ByteArrayOutputStream bout = new ByteArrayOutputStream();
        int len;
        bin = new BufferedInputStream(new FileInputStream(filename));
        while((len = bin.read(_buf)) >= 0) {
            bout.write(_buf, 0, len);
        }
        bout.flush();
        byte[] buf = bout.toByteArray();
        bout.close();
        out.writeInt(buf.length);
        out.write(buf,0,buf.length);
    }
    finally {
        if(bin != null) {
            try {
                bin.close();
            }
            catch(Exception ex) {
            }
        }
    }
}

private void makeOZD(String filename, HttpServletRequest request)
    throws Exception
{
    InputStream stream = null;
```

```
try {
   // Info.
   String   _ITEM = "parameter_test.ozr";
   String   _CATEGORY = "/";
   boolean  _MEMOALLOW = true;
   String   _PASSWORD = "1234";
   String   _UID = "admin";
   String   _PWD = "admin";

   //Form parameter
   Hashtable _FORM_PARAM = new Hashtable();
   _FORM_PARAM.put("formparam1", "FORM 1");
   _FORM_PARAM.put("formparam2", "FORM 2");
  //ODI parameter
   Hashtable _ODI_PARAM = new Hashtable();
   Hashtable _odi_param = new Hashtable();
   _odi_param.put("odiparam1", "ODI 1");
   _odi_param.put("odiparam2", "ODI 2");
   _ODI_PARAM.put("parameter_test", _odi_param);
   // call make ozd
   stream = getOZD(_ITEM, _CATEGORY,
                _FORM_PARAM, _ODI_PARAM, _MEMOALLOW, _PASSWORD,
                _UID, _PWD, request);
   //ODI parameter
   //Hashtable _ODI_PARAM = new Hashtable();
   //Hashtable _ODI_PARAM = new Hashtable();
   //Hashtable _ODI_PATH = new Hashtable();
   //_ODI _PARAM.put("odiparam1", "ODI 1");
   //_ODI _PARAM.put("odiparam2", "ODI 2");
   //_ODI_PARAM.put("parameter_test", _odi_param);
   //_ODI_PATH.put("parameter_test", "/test/parameter_test.odi");
   //call make ozd
   //stream = getOZD(_ITEM, _CATEGORY,
   //          _FORM_PARAM, _ODI_PARAM, _ODI_PATH, _MEMOALLOW,
   //          _PASSWORD, _UID, _PWD, request);
   FileOutputStream out = new FileOutputStream(filename);
   copy(stream, out);
   out.flush();
   out.close();
}
catch(Exception e) {
   e.printStackTrace();
   throw e;
}
finally {
   if(stream != null) {
      try {
         stream.close();
```

```
            }
            catch(Exception e) {}
         }
      }
   }

   // Util method
   public static int copy(InputStream is, OutputStream os) throws IOException
   {
      byte[] buf = new byte[1024];
      int rt = 0;
      int len;
      while((len = is.read(buf)) >= 0) {
         os.write(buf, 0, len);
         rt += len;
      }
      return rt;
   }
}
```

Sample : RequestMultiOZDSample.java

```
package oz.server;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class RequestMultiOZDSample extends OZServlet {

   public String getServletName() {
      return "[ RequestMultiOZDSample ]";
   }

   public void init(ServletConfig config) throws ServletException {
   }

   public void doGet(HttpServletRequest request, HttpServletResponse
response)
      throws ServletException, IOException
   {
      makeOZD(request);
   }
```

```
   public void doPost(HttpServletRequest request, HttpServletResponse
response)
       throws ServletException, IOException
   {
       makeOZD(request);
   }

   private void makeOZD(HttpServletRequest request) throws ServletException,
IOException
   {
       InputStream stream = null;
       try {

          //보고서 수 설정
int size = 2;
          String[] item = new String[size];
          String[] category = new String[size];
          Hashtable[] formparam = new Hashtable[size];
          Hashtable[] odiparams = new Hashtable[size];
          boolean memoAllowed = false;
          String ozdPassword = "";
          String serverID = "admin";
          String serverPWD = "admin";

          String[] displayName = new String[size];
          String[][] urls = {
             {"http://localhost:8080/tomcat.gif", "ozp:///REQ/b.gif"}
             ,
             {"http://localhost:8080/tomcat.gif", "ozp:///REQ/b.gif"}
          };

          Hashtable[] options = new Hashtable[size];

          for(int i=0; i<size; i++) {
             item[i] = "parameter_test.ozr";
             category[i] = "/";

             formparam[i] = new Hashtable();
             formparam[i].put("formparam1", "FORM1="+i);
             formparam[i].put("formparam2", "FORM2="+i);

             Hashtable odiparam = new Hashtable();
             odiparam.put("odiparam1", "ODIPARAM1=odi"+i);
```

```
            odiparam.put("odiparam2", "ODIPARAM2=odi"+i);

            odiparams[i] = new Hashtable();
            odiparams[i].put("parameter_test",odiparam);

            displayName[i] = (i+1)+"="+item[i];

            options[i] = new Hashtable();
            if (i == 0)
               options[i].put("use_ozd_parameter", "true");
            else
               options[i].put("use_ozd_parameter", "false");

         }

         stream = getOZD(item, category, formparam, odiparams, memoAllowed,
ozdPassword, serverID, serverPWD, request, displayName, urls, options);
         byte[] ozd = getBytes(stream);

         FileOutputStream out = new FileOutputStream("D:/MultiImage.ozd");
         out.write(ozd, 0, ozd.length);
         out.flush();
         out.close();
      }
      catch(Exception e) {
         e.printStackTrace();
         throw new ServletException(makeStackTrace(e));
      }
      finally {
         if(stream != null) {
            try {
               stream.close();
            }
            catch(Exception e) {}
         }
      }
   }

   private byte[] getBytes(InputStream is) {

      BufferedInputStream bis = new BufferedInputStream(is);

      int totalRead = 0;
      int bytesRead = 0;
```

```
        byte[] buffer = new byte[10000];
        byte[] ret = new byte[0];
        byte[] tempBuff;

        try {
           while((bytesRead = bis.read(buffer)) >=0) {
              totalRead += bytesRead;
              tempBuff = new byte[totalRead];
              if(ret.length > 0 ) {
                 System.arraycopy(ret, 0, tempBuff, 0, ret.length);
              }
              if(bytesRead > 0) {
                 System.arraycopy(buffer, 0, tempBuff, ret.length, bytesRead);
              }
              ret = tempBuff;
           }
           bis.close();
        } catch(Exception e) {
           return null;
        }

        return ret;
    }


    private String makeStackTrace(Throwable t) throws IOException {
        ByteArrayOutputStream bout = new ByteArrayOutputStream();
        t.printStackTrace(new PrintStream(bout));
        bout.flush();
        byte[] b = bout.toByteArray();
        String error = new String(bout.toByteArray());

        return getServletName() + "\n" + error;
    }

}
```

Sample : OZOZDMakerServlet.java

```
package sample;

import java.io.*;
import java.io.IOException;
```

```
import java.util.Hashtable;

import javax.servlet.*;
import javax.servlet.http.*;

import oz.server.OZServlet;

public class OZOZDMakerServlet extends OZServlet {
        private static final long serialVersionUID = 1L;

        public void init(ServletConfig config) throws ServletException{
                super.init(config);
        }

        public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
                doPost(req, res);
        }

        public void doPost(HttpServletRequest req, HttpServletResponse res)
            throws ServletException, IOException {
            //getOZDWithSDM 에서 설정할 Argument 값을 설정합니다.
                String inOutPath = "D:\\ServerRepository\\";
                String ozrName = "SDM_OZD_MAKE_TEST.ozr";
                String category = "/";
                boolean isMemo = false;
                String reportPassword = null;
                String serverID = "admin";
                String serverPwd = "admin";

                Hashtable param = new Hashtable();
                param.put("param1", "OZ PARAMETER ONE");
                param.put("param2", "OZ PARAMETER TWO");

                Hashtable sdm = new Hashtable();
                FileInputStream fis1 = new FileInputStream(inOutPath +
                                                    "SDM_OZD_MAKE_TEST.sdm");
                FileInputStream fis2 = new FileInputStream(inOutPath +
                                                    "SDM_OZD_MAKE_TEST1.sdm");
                sdm.put("SDM_OZD_MAKE_TEST", fis1);
                sdm.put("SDM_OZD_MAKE_TEST1", fis2);

                InputStream is = null;
                FileOutputStream fos = null;
                try {
                     //getOZDWithSDM 의 Argument 를 설정하여 SDM 을 설정합니다.
                        is = getOZDWithSDM(ozrName, category, sdm, param,
                                           isMemo, reportPassword, serverID,
```

```
                                                serverPwd, req);
                        fos = new FileOutputStream(inOutPath +
"RESULT.ozd");
                        byte[] buffer = new byte[1024];
                        int len = 0;
                        while((len = is.read(buffer)) > 0) {
                                fos.write(buffer, 0, len);
                        }
                } catch(Exception e) {
                        e.printStackTrace();
                } finally {
                        if(fos != null) {
                                fos.close();
                        }

                        if(is != null) {
                                is.close();
                        }
                        fis1.close();
                        fis2.close();
                }
        }

        public void detroy() {
                super.destroy();
        }
}
```

Ⅱ. 오즈 스케줄러 서버 API

- Class Program
- Class Publisher
- Class Scheduler
- Class TaskHolidayInfo
- Class TaskHolidayGroupInfo

오즈 스케줄러 서버에 관련된 각종 정보 조회와 실시간 환경 설정 변경 기능을 사용자 애플리케이션에서 직접 제어할 수 있도록 자바 API를 제공합니다.

다음은 오즈 스케줄러 서버 API로 제공하는 주요 클래스에 대한 설명입니다.

클래스 이름	설명
Program	오즈 스케줄러에서 외부 프로그램 등록 및 관리 기능을 수행합니다.
Publisher	오즈 스케줄러에 의해 생성되는 보고서 결과 파일을 관리합니다.
Scheduler	보고서 스케줄 작업을 생성하고 결과 파일 및 스케줄러 관리 기능을 수행합니다.
TaskHolidayInfo	태스크에 지정할 휴일을 정의합니다.
TaskHolidayGroupInfo	태스크에 지정할 휴일 그룹을 정의합니다.

오즈 스케줄러 서버 API를 사용하기 위해서는 다음과 같은 라이브러리 파일을 클래스 패스에 추가해 주어야 합니다.

라이브러리 파일명	설명
ozsfw60.jar	오즈 서버 및 Scheduler server에 접속하기 위한 파일입니다.
log4.jar	Server 내부의 클래스들을 사용할 때 Log를 남기기 위한 라이브러리 파일입니다. (API를 이용한 프로그램 실행시 classpath에 "log4.jar" 파일이 위치하여야 함)

Class Program

Constructor Summary

- **Program(String ip, int port)**

Method Summary

- **void createFolder(ServerInfo s, String folder)**
- **byte[] downloadFile(ServerInfo s, String file)**
- **FileInfo[] getExternalProgramList(ServerInfo s, String folder)**
- **void removeFiles(ServerInfo s, String folder, String[] files)**
- **void removeFolder(ServerInfo s, String folder, boolean isAll)**
- **void uploadFile(ServerInfo s, String file, byte[] b)**

Constructor Detail

Prototype	`public Program(String ip, int port)`	
Argument	*ip*	오즈 스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP ex) String ip = "127.0.0.1";
	port	스케줄러 포트 (기본값:9521) ex) int port = 9521;

Method Detail

- **createFolder**

Prototype	`public void createFolder(ServerInfo s, String folder)`	
Definition	스케줄러에서 외부 프로그램의 폴더를 생성합니다.	
Argument	*s*	오즈 서버 정보

	folder	생성할 폴더명

■ **downloadFile**

Prototype	public byte[] downloadFile(ServerInfo s, String file)	
Definition	등록된 외부 프로그램 파일을 다운로드합니다.	
Argument	*s*	오즈 서버 정보
	file	다운로드할 파일명

■ **getExternalProgramList**

Prototype	public FileInfo[] getExternalProgramList(ServerInfo s, String folder)	
Definition	스케줄러에서 외부 프로그램 목록을 가져옵니다.	
Argument	*s*	오즈 서버 정보
	folder	폴더명

■ **removeFiles**

Prototype	public void removeFiles(ServerInfo s, String folder, String[] files)	
Definition	등록된 외부 프로그램 파일을 삭제합니다.	
Argument	*s*	오즈 서버 정보
	folder	폴더명
	files	삭제할 프로그램 파일명

■ **removeFolder**

Prototype	public void removeFolder(ServerInfo s, String folder, boolean isAll)	
Definition	스케줄러에서 외부 프로그램의 폴더를 삭제합니다.	
Argument	*s*	오즈 서버 정보
	folder	삭제할 폴더명
	isAll	메소드 실행 여부 true : 메소드가 실행됨(외부 프로그램 폴더 삭제됨) false : 메소드가 실행안됨(외부 프로그램 폴더 삭제안됨)

■ **uploadFile**

Prototype	`public void uploadFile(ServerInfo s, String file, byte[] b)`	
Definition	스케줄러 서버에 외부 프로그램을 등록(업로드)합니다.	
Argument	*s*	오즈 서버 정보
	file	등록할 파일명
	b	등록할 파일의 바이트 값

관련 Class

■ **SchedulerException(oz.scheduler.SchedulerException)**

오즈 스케줄러 서버에서 발생하는 Exception입니다.

■ **ServerInfo(oz.scheduler.ServerInfo)**

오즈 서버에 대한 기본적인 정보를 가지고 있는 클래스입니다.

- 메소드

▪ setIsDaemon

Prototype	`public final void setIsDaemon(boolean isDaemon)`	
Definition	오즈 서버 타입을 설정합니다.	
Argument	*isDaemon*	오즈 서버가 Daemon 타입인지 여부 • true : Daemon 타입 서버 • false : Servlet 타입 서버

▪ setIP

Prototype	`public final void setIP(String ip)`	
Definition	오즈 서버 IP를 설정합니다. Server가 Daemon 타입일 경우에만 설정합니다.	
Argument	*ip*	오즈 서버 IP

▪ setPortNo

Prototype	`public final void setPortNo(int portNo)`	
Definition	오즈 서버 Port 번호를 설정합니다. Server가 Daemon 타입일 경우에만 설정합니다.	
Argument	*portNo*	오즈 서버 Port 번호

- setURL

Prototype	`public final void setURL(String url) throws IllegalArgumentException`	
Definition	오즈 서버 URL을 설정합니다. Server가 Servlet 타입일 경우에만 설정합니다.	
Argument	*url*	오즈 서버 URL URL은 대소문자로 구분하며, 반드시 'http://'로 시작해야 합니다.

- setID

Prototype	`public final void setID(String id)`	
Definition	사용자의 ID를 설정합니다.	
Argument	*id*	사용자의 ID

- setPWD

Prototype	`public final void setPWD(String pwd)`	
Definition	사용자의 패스워드를 설정합니다.	
Argument	*pwd*	사용자의 패스워드

■ **FileInfo(oz.scheduler.FileInfo.class)**

파일 또는 폴더에 대한 기본 정보값(파일/폴더, 이름, 크기, 최종 수정시각)들을 가지고 있습니다.

- 멤버 변수
  - public boolean isDirectory : 디렉토리(폴더)인지 파일인지 구분
  - public String name : 파일명 또는 디렉토리명
  - public long size : 파일 크기 또는 디렉토리 크기
  - public long lastModified : 파일 또는 디렉토리의 최종 수정 시각, 1970년 1월 1일 00:00:00 GMT를 기준으로 일시초로 시간을 계산한 값

Sample : ProgramSample.java

```
package sample;

import oz.scheduler.ServerInfo;
import oz.scheduler.FileInfo;
```

```
import oz.framework.api.Program;
import org.apache.log4j.*;

public class ProgramSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      //스케줄러 정보
      String IP = "127.0.0.1"; //스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 9521;     //스케줄러 서버가 사용하는 TCP 포트

      Program program = null;
      try {
         program = new Program(IP, PORT);

         // 스케줄러 서버의 정보
         ServerInfo serverInfo = new ServerInfo();

         serverInfo.setID("admin");    //아이디
         serverInfo.setPWD("admin");  //패스워드
         serverInfo.setIP(IP);        //OZ Server IP 설정. daemon 일 때만 사용
         serverInfo.setIsDaemon(true);
         //OZ Server 타입 daemon, servlet 불리언 값
         serverInfo.setPortNo(8003);
         /* OZ Server Port 번호 설정.daemon 일 때만
          * s.setURL(null);//OZ Server URL 을 설정합니다.
          * URL 은 대소문자 구분하며, 반드시 //'http://' 로 시작해야
          * 합니다. Server 가 Servlet Type 일 경우에만 설정합니다.*/

         // 스케줄러 외부 프로그램 폴더 만들기(createFolder)
         String folderName = "testFolder"; //생성할 폴더 이름
         program.createFolder(serverInfo, folderName);

         // 외부 프로그램 등록하기(uploadFile)
         String fileName = "testProgram.bat"; //등록할 외부 프로그램 이름
         byte [] fileBytes = fileName.getBytes();
         program.uploadFile(serverInfo, folderName+"/"+fileName, fileBytes);

         // 등록된 외부 프로그램 내려받기(downloadFile)
         String downFileName = "testProgram.bat";//다운로드할 프로그램 파일 이름
         byte bytesArray[] =
            program.downloadFile(serverInfo, folderName+"/"+downFileName);

         // 스케줄러 외부 프로그램 목록 구하기(getExternalProgramList)
         FileInfo fileInfo[] =
            program.getExternalProgramList(serverInfo, folderName);
         for(int i=0;i<fileInfo.length;i++){
```

```
            FileInfo fi = fileInfo[i];
            System.out.println(i);
            System.out.println(" isDirectory " + fi.isDirectory);
            System.out.println(" name " + fi.name);
            System.out.println(" size " + fi.size);
            System.out.println(" lastModified " + fi.lastModified);
            System.out.println();
         }

         // 외부 프로그램 제거(removeFiles)
         // 삭제할 파일(들)의 이름을 String 배열에 넣어준다.
         String file1 = "testProgram.bat";
         String file2 = "testProgram2.bat";
         String [] files = new String[]{file1 }; //삭제할 파일들의 배열.
         program.removeFiles(serverInfo, folderName, files);

         // 외부 프로그램 폴더 제거(removeFolder)
         boolean isAll = true; //삭제 여부
         String folderNameToRemove = "testFolder";
         //삭제할 폴더명 하나만 들어간다.
         program.removeFolder(serverInfo, folderNameToRemove, isAll);

      }
      catch(Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

Class Publisher

Constructor Summary

- **Publisher(String ip, int port)**

Method Summary

- **void createFolder(ServerInfo s, String folder)**
- **byte[] downloadFile(ServerInfo s, String file)**
- **FileInfo[] getPublishedFiles(ServerInfo s, String folder)**
- **void removeFiles(ServerInfo s, String folder, String[] files)**
- **void removeFolder(ServerInfo s, String folder, boolean isAll)**

Constructor Detail

Prototype	`public Publisher(String ip, int port)`	
Argument	*ip*	오즈 스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP ex) String ip = "127.0.0.1";
	port	스케줄러 포트 (기본값:9521) ex) int port = 9521;

Method Detail

- **createFolder**

Prototype	`public void createFolder(ServerInfo s, String folder)`	
Definition	스케줄러 익스포트 파일을 저장할 폴더를 생성합니다.	
Argument	*s*	오즈 서버 정보
	folder	생성할 폴더명

■ **downloadFile**

Prototype	`public byte[] downloadFile(ServerInfo s, String file)`	
Definition	등록된 스케줄러 익스포트 파일을 다운로드합니다.	
Argument	*s*	오즈 서버 정보
	file	다운로드할 파일명

■ **getPublishedFiles**

Prototype	`public FileInfo[] getPublishedFiles(ServerInfo s, String folder)`	
Definition	스케줄러에서 익스포트 파일 목록을 가져옵니다.	
Argument	*s*	오즈 서버 정보
	folder	폴더명

■ **removeFiles**

Prototype	`public void removeFiles(ServerInfo s, String folder, String[] files)`	
Definition	등록된 스케줄러 익스포트 파일을 삭제합니다.	
Argument	*s*	오즈 서버 정보
	folder	삭제할 폴더명
	fiels	삭제할 파일명

■ **removeFolder**

Prototype	`public void removeFolder(ServerInfo s, String folder, boolean isAll)`	
Definition	스케줄러 익스포트 파일이 저장되는 폴더를 삭제합니다.	
Argument	*s*	오즈 서버 정보
	folder	삭제할 폴더명
	isAll	메소드 실행 여부 true : 메소드가 실행됨(외부 프로그램 폴더 삭제됨) false : 메소드가 실행안됨(외부 프로그램 폴더 삭제안됨)

관련 Class

- **ServerInfo(oz.scheduler.ServerInfo)**

  Program class의 "관련 class" 부분을 참조하시기 바랍니다.

- **FileInfo(oz. scheduler.FileInfo.class)**

  Program class의 "관련 class" 부분을 참조하시기 바랍니다.

Sample : PublisherSample.java

```
package sample;

import oz.framework.api.Publisher;
import oz.scheduler.FileInfo;
import oz.scheduler.ServerInfo;
import org.apache.log4j.*;

public class PublisherSample {
   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      //스케줄러 정보
      String IP = "127.0.0.1";    //스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 9521;        //스케줄러 서버가 사용하는 TCP 포트

      Publisher publisher = null;
      try {
         publisher = new Publisher(IP, PORT);

         // 스케줄러 서버의 정보
         ServerInfo serverInfo = new ServerInfo();

         serverInfo.setID("admin");   //아이디
         serverInfo.setPWD("admin");  //패스워드
         serverInfo.setIP(IP);       //OZ Server IP 설정. daemon 일 때만 사용
         serverInfo.setIsDaemon(true);
         //OZ Server 타입 구분.daemon, servlet 불리언 값
         serverInfo.setPortNo(8003);
         /* OZ Server Port 번호 설정.daemon 일 때만
          * s.setURL(null);//OZ Server URL 을 설정합니다.
          * URL 은 대소문자 구분하며, 반드시 //'http://' 로 시작해야
          * 합니다. Server 가 Servlet Type 일 경우에만 설정합니다.*/
```

```
            // 스케줄러 생성 파일 저장 폴더 만들기(createFolder)
            String folderName = "excel"; //만들 폴더 이름
            //publisher.createFolder(serverInfo, folderName);

            // 등록된 스케줄러 생성파일을 내려받기(downloadFile)
            String fileName = "customer.xls"; //내려받을 프로그램 파일이름
            byte bytesArray[] = publisher.downloadFile(serverInfo, fileName);

            // 스케줄러 생성 파일 목록(getPublishedFiles)
            FileInfo fileInfoList[] =
               publisher.getPublishedFiles(serverInfo, folderName);
            for(int i=0; i<fileInfoList.length; i++) {
               FileInfo fi = fileInfoList[i];
               System.out.println(i);
               System.out.println(" isDirectory="+fi.isDirectory);
               System.out.println(" name="+fi.name);
               System.out.println(" size="+fi.size);
               System.out.println(" lastModified="+fi.lastModified);
               System.out.println();
            }
            // 스케줄러 생성 파일 삭제(removeFiles)
            //  삭제할 파일(들)의 이름을 String 배열에 넣어준다.
            //  customer.xls 와 orderInfo.xls 두개를 삭제하는 예시
            String file1 = "customer.xls";
            String file2 = "orderInfo.xls";
            String testFolderName = "excel";
            String [] filesToRemove = new String[]{file1, file2};
            //삭제할 파일들의 배열
            publisher.removeFiles(serverInfo, testFolderName, filesToRemove);

            // 스케줄러에서 생성되는 파일이 저장되는 폴더 제거
            boolean isAll = true;//삭제여부
            String folderNameToRemove = "excel";//삭제할 폴더 이름. 하나만 들어감.
            publisher.removeFolder(serverInfo, folderNameToRemove, isAll);

         }
         catch(Exception e)
         {
         }
      }
}
```

Class Scheduler

Constructor Summary

- **scheduler(String ip, int port)**

Method Summary

- **String createTask(ServerInfo s, NameValueCollection configMap, NameValueCollection exportMap)**
- **Vector getTask(ServerInfo s)**
- **TaskResult[] getTaskResult(ServerInfo s, String from, String to, String taskId)**
- **SortProperties[] getTaskProperties(ServerInfo info, String taskID)**
- **void removeTask(ServerInfo s, String task)**
- **String modifyTask(ServerInfo s, String taskid, SortProperties p, SortProperties exportMap)**
- **boolean taskPause(ServerInfo s, String task)**
- **boolean taskResume(ServerInfo s, String task)**
- **void stop(ServerInfo s, boolean waitTask)**
- **boolean export(ServerInfo s, SortProperties configMap, SortProperties exportMap)**
- **boolean makePDF(ServerInfo s, SortProperties configMap, SortProperties exportMap)**
- **boolean print(ServerInfo s, SortProperties configMap, SortProperties printMap)**
- **SortProperties getConfiguration(ServerInfo s)**
- **void modifyConfiguration(ServerInfo s, SortProperties configMap, SortProperties exportMap)**
- **boolean ping()**
- **String[] getOZSList(String path)**
- **void ozsVerConv(String oldPath, String newPath)**

- **boolean addTaskHolidayInfo(TaskHolidayInfo value)**
- **boolean modifyTaskHolidayInfo(String old_key, TaskHolidayInfo new_value)**
- **boolean deleteTaskHolidayInfo(String key)**
- **boolean deleteTaskHolidayInfo(String[] keys)**
- **boolean addTaskHolidayGroupInfo(TaskHolidayGroupInfo value)**
- **boolean modifyTaskHolidayGroupInfo(String old_key, TaskHolidayInfo new_value)**
- **boolean deleteTaskHolidayGroupInfo(String key)**
- **OZMap getTaskHolidayInfoList()**
- **OZMap getTaskHolidayGroupInfoList()**
- **void saveTaskHoliday()**
- **DirectExportResult directExport(ServerInfo s, SortProperties configMap, SortProperties exportMap) throws SchedulerException**
- **DirectPrintResult directPrint(ServerInfo s, SortProperties configMap, SortProperties printMap) throws SchedulerException**

Constructor Detail

Prototype	`public scheduler(String ip, int port)`	
Argument	*ip*	오즈 스케줄러 서버가 설치되어 있는 호스트 컴퓨터의 IP ex) String ip = "127.0.0.1";
	port	스케줄러 포트 (기본값:9521) ex) int port = 9521;

Method Detail

- **createTask**

Prototype	`public String createTask(ServerInfo s, NameValueCollection configMap, NameValueCollection exportMap)`

Definition	폼 파일을 기반으로 스케줄링 태스크를 생성한 후 생성된 태스크 ID를 가져옵니다. ※ 참고사항 : CreateTask 함수로 여러 개의 태스크 생성 시 뷰어의 대기 큐와 상관없이 실행되는 뷰어의 개수만큼 Thread와 param 파일이 생성됩니다.	
Argument	*s*	오즈 서버 정보
	configMap	태스크를 생성할 때 사용되는 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.
	exportMap	스케줄러에서 익스포트되는 파일의 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

■ getTask

Prototype	`public Vector getTask(ServerInfo s)`	
Definition	현재 생성된 스케줄링 태스크의 결과물을 가져옵니다.	
Argument	*s*	오즈 서버 정보

■ getTaskResult

Prototype	`public TaskResult[] getTaskResult(ServerInfo s, String from, String to, String taskId)`	
Definition	스케줄링 태스크 결과를 가져옵니다.	
Argument	*s*	오즈 서버 정보
	from	태스크 결과를 가져올 시작 시간
	to	태스크 결과를 가져올 종료 시간
	taskId	가져올 태스크 아이디

■ getTaskProperties

Prototype	`public SortProperties[] getTaskProperties(ServerInfo s, String taskId)`	
Definition	태스크의 속성 즉 스케줄러에서 태스크를 생성할 때 사용되는 설정 옵션(configMap)과 스케줄러에서 익스포트되는 파일의 설정 옵션(exportMap)을 가져옵니다.	
Argument	*s*	오즈 서버 정보
	taskId	속성을 가져올 태스크 아이디

■ **removeTask**

Prototype	`public void removeTask(ServerInfo s, String taskId)`	
Definition	스케줄링 태스크를 삭제합니다.	
Argument	*s*	오즈 서버 정보
	taskId	삭제할 태스크 아이디

■ **modifyTask**

Prototype	`public String modifyTask(ServerInfo s, String taskId, SortProperties configMap, SortProperties exportMap)`	
Definition	태스크 속성을 변경합니다.	
Argument	*s*	오즈 서버 정보
	taskId	속성을 변경할 태스크 아이디
	configMap	스케줄러 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.
	exportMap	익스포트 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

■ **taskPause**

Prototype	`public boolean taskPause(ServerInfo s, String taskId)`	
Definition	스케줄링 태스크를 중지시킵니다.	
Argument	*s*	오즈 서버 정보
	taskId	중지할 태스크 아이디

■ **taskResume**

Prototype	`public boolean taskResume(ServerInfo s, String taskId)`	
Definition	중지된 스케줄링 태스크를 다시 실행시킵니다.	
Argument	*s*	오즈 서버 정보
	taskId	다시 실행시킬 태스크 아이디

■ **stop**

Prototype	`public void stop(ServerInfo s, boolean waitTaskId)`

Definition	스케줄러를 중지시킵니다.	
Argument	*s*	오즈 서버 정보
	waitTaskId	스케줄러를 강제로 중지시킬지 여부

■ **export**

Prototype	`public boolean export(ServerInfo s, SortProperties configMap, SortProperties exportMap)`	
Definition	뷰어 파라미터를 그대로 사용하여 익스포트한 후 익스포트 성공 여부를 반환합니다. ※ 주의사항 : 서버에 설정된 스케줄러의 환경 설정 중 "ViewType=None"이 아닌 경우에만 사용할 수 있습니다. ※ 주의사항 : 익스포트할 수 있는 파일 형식은 *.pdf, *.ozd, *.html, *.jpg, *.xls, *.doc, *.svg, *.txt, *.ppt, *.tif, *csv 파일만 가능합니다.	
Argument	*s*	오즈 서버 정보
	configMap	스케줄러 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.
	exportMap	익스포트 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

※ 참고사항 : export 함수를 API 함수가 아니라 COM으로 매핑하여 ASP에서 사용하는 방법은 본 매뉴얼을 "Appendix 1. SchedulerCOM 활용" 부분을 참조하시기 바랍니다.

■ **makePDF**

Prototype	`public boolean makePDF(ServerInfo s, SortProperties configMap, SortProperties exportMap)`	
Definition	스케줄러의 작업 목록에 등록하지 않고 PDF 익스포트만 수행한 후 익스포트 성공 여부를 반환합니다. ※ 주의사항 : 서버에 설정된 스케줄러의 환경 설정 중 "ViewType=None"이 아닌 경우에만 사용할 수 있습니다.	
Argument	*s*	오즈 서버 정보
	configMap	익스포트를 위한 폼 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다. creatTask()와 비슷하나 PDF 관련 정보만 입력하면 됩니다.

	exportMap	PDF 익스포트 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다. creatTask()와 비슷하나 PDF 관련 정보만 입력하면 됩니다.

※ 참고사항 : makePDF 함수를 API 함수가 아니라 COM으로 매핑하여 ASP에서 사용하는 방법은 본 매뉴얼을 "Appendix 1. SchedulerCOM 활용" 부분을 참조하시기 바랍니다.

■ **print**

Prototype	public boolean print(ServerInfo s, SortProperties configMap, SortProperties printMap)	
Definition	뷰어 패러미터를 적용시켜 프린트하고 성공 여부를 반환합니다. (실제 프린트 작업의 성공 여부가 아니라 뷰어에 프린트 호출이 성공했는지 여부를 반환함) ※ 주의사항 : "task_type=viewerTag"인 경우에만 사용될 수 있으며 이외의 값이 들어오면 무시합니다. 또한 서버에 설정된 스케줄러의 환경설정 중 "ViewType=None"이 아닌 경우에만 사용할수 있습니다.	
Argument	*s*	오즈 서버 정보
	configMap	스케줄러 설정 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.
	printMap	출력 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다. 단, print.mode = silent로만 실행됩니다.

※ 참고사항

- print API 호출 시 태스크 작업 주기 옵션과 관계없이 즉시 한번만 프린트 작업을 호출하고 태스크는 생성되지 않으며, print API 호출 시 메일 보내기 등의 패러미터를 설정하여도 메일 관련 작업은 일어나지 않고 오직 프린트만 합니다.
- print 함수를 API로 구현할 경우 뷰어 패러미터에서 설정한 패러미터 중 아래의 패러미터는 항상 해당 값으로 고정되어 동작합니다.

```
viewer.allowmultiframe=true
viewer.mode=print
viewer.printcommand=true
viewer.showerrormessage=false
viewer.useprogressbar=false
print.ingnoreerror=false
```

```
print.mode=silent
export.confirmsave=false
export.format=""
information.debug=debug
```

■ **getConfiguration**

Prototype	`public SortProperties getConfiguration(ServerInfo s)`	
Definition	스케줄러 설정값을 가져옵니다.	
Argument	*s*	오즈 서버 정보

■ **modifyConfiguration**

Prototype	`public void modifyConfiguration(ServerInfo s, SortProperties configMap)`	
Definition	스케줄러의 설정값을 변경합니다.	
Argument	*s*	오즈 서버 정보
	configMap	스케줄러 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

■ **ping**

Prototype	`public boolean ping()`
Definition	서버의 구동 여부를 반환합니다.

■ **getOZSList**

Prototype	`public String[] getOZSList(String path)`	
Definition	설정한 경로에 있는 OZS 파일 목록을 가져옵니다. (path를 파일명으로 지정하면 해당 OZS 파일 정보만 가져오며, path를 폴더명으로 지정하면 해당 폴더에 있는 모든 OZS 파일 정보를 가져옵니다. 만일 해당 경로에 파일이 존재하지 않거나 존재하지 않는 폴더를 지정하면 빈 값을 리턴합니다.)	
Argument	*path*	OZS 파일 목록을 가져올 경로 ※ 주의사항 : OZS 파일 목록을 가져올 경로는 /%SCH_HOME%/[path]/로 지정됩니다. 예를 들어 path를 sample로 설정할 경우에는 /%SCH_HOME%/sample/에 있는 OZS 파일 목록을 가져옵니다.

■ **ozsVerConv**

Prototype	public void ozsVerConv(String oldPath, String newPath)	
Definition	2.5 버전의 OZS 파일을 최신 버전으로 변경하여 저장합니다.	
Argument	*oldPath*	변환할 OZS 파일명 또는 경로 ※ 참고사항 : 해당 경로는 /%SCH_HOME%/[oldPath]/입니다.
	newPath	변환된 OZS 파일 저장 경로 ※ 참고사항 : 해당 경로는 /%SCH_HOME%/[newPath]/이며 newPath는 oldPath와 다르게 설정하여야 합니다.

■ **addTaskHolidayInfo**

Prototype	public boolean addTaskHolidayInfo(TaskHolidayInfo value)	
Definition	태스크 휴일 정보를 추가합니다.	
Argument	*value*	태스크 휴일 정보

■ **modifyTaskHolidayInfo**

Prototype	public boolean modifyTaskHolidayInfo(String old_key, TaskHolidayInfo new_value)	
Definition	태스크 휴일 정보를 수정합니다.	
Argument	*old_key*	수정할 태스크 휴일 이름
	new_value	새로운 태스크 휴일 정보

■ **deleteTaskHolidayInfo**

Prototype	//한 개의 태스크 휴일 정보를 삭제할 경우
	public boolean deleteTaskHolidayInfo(String[] keys)
	
	//여러 개의 태스크 휴일 정보를 삭제할 경우
	
	public boolean deleteTaskHolidayInfo(String key)

Definition	태스크 휴일 정보를 삭제합니다.
Argument	*key* 삭제할 태스크 휴일 이름 배열

■ **addTaskHolidayGroupInfo**

Prototype	public boolean addTaskHolidayGroupInfo(TaskHolidayGroupInfo value)
Definition	태스크 휴일 그룹 정보를 추가합니다.

Argument	*value*	태스크 휴일 그룹 정보

■ **modifyTaskHolidayGroupInfo**

Prototype	public boolean modifyTaskHolidayGroupInfo(String old_key, TaskHolidayGroupInfo new_value)
Definition	태스크 휴일 그룹 정보를 수정합니다.
Argument	*old_key* 수정할 태스크 휴일 그룹 이름
	new_value 새로운 태스크 휴일 그룹 정보

■ **deleteTaskHolidayGroup**

Prototype	public boolean deleteTaskHolidayGroupInfo(String key)
Definition	태스크 휴일 그룹 정보를 삭제합니다.
Argument	*key* 삭제할 태스크 휴일 그룹 이름

■ **getTaskHolidayInfoList**

Prototype	public OZMap getTaskHolidayInfoList()
Definition	태스크 휴일 정보 목록을 가져옵니다.

■ **getTaskHolidayGroupInfoList**

Prototype	public OZMap getTaskHolidayGroupInfoList()
Definition	태스크 휴일 그룹 정보 목록을 가져옵니다.

■ **saveTaskHoliday**

Prototype	public void saveTaskHoliday()
Definition	태스크 휴일 정보를 xml 파일로 저장합니다.

■ **directExport**

Prototype	public DirectExportResult directExport(ServerInfo s, SortProperties configMap, SortProperties exportMap) throws SchedulerException

Definition	뷰어 패러미터를 적용시켜 익스포트한 후 결과 정보를 반환합니다. ※ 주의사항 : 서버에 설정된 스케줄러의 환경 설정 중 "ViewType=None"이 아닌 경우에만 사용할 수 있습니다. ※ 주의사항 : 익스포트할 수 있는 파일 형식은 *.pdf, *.ozd, *.html, *.jpg, *.xls, *.doc, *.svg, *.txt, *.ppt, *.tif, *csv 파일만 가능합니다.	
Argument	*s*	오즈 서버 정보
	configMap	스케줄러 설정 옵션 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.
	exportMap	익스포트 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.

■ **directPrint**

Prototype	`public DirectPrintResult directPrint(ServerInfo s, SortProperties configMap, SortProperties printMap) throws SchedulerException`	
Definition	뷰어 패러미터를 적용시켜 프린트하고 결과 정보를 반환합니다. ※ 주의사항 : "task_type=viewerTag"인 경우에만 사용될 수 있으며 이외의 값이 들어오면 무시합니다. 또한 서버에 설정된 스케줄러의 환경 설정 중 "ViewType=None"이 아닌 경우에만 사용할수 있습니다.	
Argument	*s*	오즈 서버 정보
	configMap	스케줄러 설정 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다.
	printMap	출력 정보 설정할 수 있는 key와 설명은 아래 "Option" 부분을 참조하시기 바랍니다. 단, print.mode = silent로만 실행됩니다.

관련 Class

■ **ServerInfo(oz.scheduler.ServerInfo)**

Program class의 "관련 class" 부분을 참조하시기 바랍니다.

■ **SortProperties(oz.util.SortProperties)**

Cache class의 "관련 class" 부분을 참조하시기 바랍니다.

- **TaskHolidayInfo(oz.scheduler.Holiday)**

  TaskHolidayInfo class의 "관련 class" 부분을 참조하시기 바랍니다.

- **TaskHolidayGroupInfo(oz.scheduler.Holiday)**

  TaskHolidayInfo class의 "관련 class" 부분을 참조하시기 바랍니다.

- **ScheduledTask(oz.scheduler.ScheduledTask)**

  생성된 태스크 정보를 나타냅니다.

  - 멤버 변수
    - public String taskID : 태스크 ID
    - public String taskName : 태스크 이름
    - public String taskGoupName : 태스크 그룹 이름
    - public String formCategoryName : 보고서 카테고리 이름
    - public String formFileName : 보고서 이름
    - public int schedulingType : 스케줄링 타입 - 0(즉시 실행), 1(한 번 실행), 2(주기적 실행) 중 하나의 값
    - public String schedulingTypeStr : 스케줄링 타입 - "Immediately"(즉시 실행), "Once at specific time"(한 번 실행), "Periodically"(주기적 실행) 중 하나의 값
    - public String nextRunTimeStr : 다음 실행 시간
    - public String lastRunTimeStr : 마지막으로 실행한 시간
    - public String status : 태스크 실행 상태 - "W"(대기), "R"(실행), "P"(중지) 중 하나의 값

- **TaskResult(oz.scheduler.TaskResult)**

  태스크 실행 결과를 나타냅니다.

  - 멤버 변수
    - public String taskID : 태스크 ID
    - public String taskName : 태스크 이름
    - public String taskGoupName : 태스크 그룹 이름
    - public String completedTime : 태스크 완료 시간
    - public int isSuccessfulCode : 태스크 성공 여부 코드 - 100001(General Failed), 200001(Exporting Failed), 300001(Mailing

Failed), 400001(Succeeded) 중 하나의 값

- public String isSuccessful : 태스크 성공 여부 - "Failed", "Exporting Failed", "Mailing Failed", "Succeeded" 중 하나의 값
- public String formCategoryName : 보고서 카테고리 이름
- public String formFileName : 보고서 이름
- public String Parameter : 패러미터
- public String schedulingType : 스케줄링 타입 - "immediately"(즉시 실행), "once"(한 번 실행), "periodically"(주기적 실행) 중 하나의 값
- public String exportFileList : 익스포트 파일 이름 리스트, 여러 개의 파일일 경우 "|"를 구분자로 함
- public String errorMsg : 에러 메시지

■ **DirectTaskResult(oz.scheduler.DirectTaskResult)**

태스크 실행 결과를 나타냅니다.

- 멤버 변수
  - public String taskID : 태스크 ID
  - public String taskName : 태스크 이름
  - public String taskGoupName : 태스크 그룹 이름
  - public String completedTime : 태스크 완료 시간
  - public String executeTime : 태스크 수행 시간 (단위 : 초)
  - public boolean isSuccessful : 태스크 성공 여부
  - public String formCategoryName : 보고서 카테고리 이름
  - public String formName : 보고서 이름

■ **DirectExportResult(oz.scheduler.DirectExportResult)**

익스포트 태스크 실행 결과를 나타냅니다.

- 멤버 변수
  - public String exportFileList : 익스포트된 파일 이름
  - public int pageCount : 보고서의 익스포트된 페이지 수

    ※ 주의사항

    ① 한 보고서를 여러 개의 포맷으로 동시에 익스포트할 경우 보고서의 익스포트된 페이지 수 합계를 나타냅니다.
    예를 들어, 페이지 수가 4 페이지인 보고서를 xls, doc 포맷으로 익스포트 시 pageCount 값으로 8이 리턴됩니다.

    ② 보고서 익스포트 시 "viewer.largebundle=true"일 경우 pageCount 값으로

1이 리턴됩니다.

■ **DirectPrintResult(oz.scheduler.DirectPrintResult)**

프린트 태스크 실행 결과를 나타냅니다.

- 멤버 변수
  - public int pageCount : 페이지 수
  - public int pageCopy : 인쇄 매수
  - public String pageRange : 인쇄 범위
  - public String printerName : 프린터 이름
  - public String printerDriverName : 프린터 드라이버 이름

관련 Interface

■ **OZSchedulerIdleTime(oz.scheduler.idletime)**

- init

Prototype	void init(org.apache.log4j.Category cat) throws OZSchedulerIdleTimeException
Definition	스케줄러 서버 구동 시 호출됩니다.
Argument	*cat* 스케줄러 서버 구동 시 로그에 남길 내용

- release

Prototype	void release(org.apache.log4j.Category cat) throws OZSchedulerIdleTimeException
Definition	스케줄러 서버 종료 시 호출됩니다.
Argument	*cat* 스케줄러 서버 종료시 그에 남길 내용

- idleTimeProcess

Prototype	void idleTimeProcess(org.apache.log4j.Category cat) throws OZSchedulerIdleTimeException
Definition	유휴 시간에 실행할 프로세스를 정의합니다.
Argument	*cat* 스케줄러 서버 구동 시 로그에 남길 내용

■ **Exception**

IdleTime에서 발생되는 Exception입니다.

- OZSchedulerIdleTimeException

Prototype	`public OZSchedulerIdleTimeException(String msg)`
Definition	IdleTime에서 발생되는 Exception입니다.
Argument	*msg* 에러 메시지

Option

※ 주의사항

- 서버에 설정된 스케줄러의 환경 설정 중 "ViewType"을 "None"으로 설정한 경우에는 익스포트시에 뷰어를 사용하지 않고, 서버 API를 이용하여 익스포트 하므로, OZD 파일로만 익스포트할 수 있습니다.

- 뷰어 패러미터 중 viewer.mode 값은 "export" 또는 "print"로만 설정할 수 있으며, 그 외 값을 사용자가 입력하였을 경우 "export"로 설정되어 동작합니다. viewer.mode 패러미터를 제외한 몇몇 패러미터는 아래 표와 같이 사용자가 설정한 값은 무시되고, 항상 고정된 값으로만 설정되어 동작됩니다.

뷰어 패러미터	값
viewer.useprogressbar	false
viewer.allowmultiframe	true
export.mode	silent
export.confirmsave	false
print.mode	silent
print.ignoreerror	false
information.debug	debug
viewer.showerrormessage	false

- 여러 개의 보고서를 설정할 경우 자식 보고서에 설정할 수 있는 옵션은 reportname, displayname, external_program_check, external_program_command, image_url_size, image_url_n, parameter_count, parameter_name_n, parameter_value_n, option_properties_size, option_property_name_n, option_property_value_n 입니다.

■ 기본 설정

- 리포트 이름 설정

Key	Value	설명
reportname	리포트명	태스크 생성할 리포트 경로 ex) setProperty("reportname", "/categpry/sample.ozr")

- 보고서 표시 이름 설정

Key	Value	설명
displayname	보고서 표시명	오즈 뷰어의 보고서 트리에 표시할 보고서명 ex) setProperty("displayname", "sample")

- 보고서 수 설정

Key	Value	설명
reportcount	보고서 수	자식 보고서 개수 (기본값 : 1) ex) setProperty("reportcount", "2")

- 서버 데이터 모듈

Key	Value	설명
dm_server_check	"check" "null"	서버 데이터 모듈 선택 ex) setProperty("dm_server_check", "check")
dm_server_name	SDM 파일명	서버 데이터 모듈 파일명 (Repository의 odi가 있는 category에 저장된 SDM 파일명) ex) setProperty("dm_server_name", "test1.sdm")
odi_name	ODI명	서버 데이터 모듈을 만들 ODI명 ex) setProperty("odi_name","testodi.odi")
odi_category_name	카테고리명	서버 데이터 모듈을 만들 ODI가 저장된 카테고리명 (루트인 경우 "/" 로 입력) ex) setProperty("odi_category_name", "/test")

- 태스크 그룹 이름 설정

Key	Value	설명
task_group	그룹명	태스크 그룹명 ex) setProperty("tast_group", "/Sales")

- 태스크 이름 설정

Key	Value	설명
task_name	태스크명	태스크명 ex) setProperty("tast_name", "SalesReportTask")

- 태스크 타입 설정

Key	Value	설명
task_type	"viewerTag" "none" "sdm"	태스크 타입 설정 viewerTag : 보고서 생성 태스크 또는 OZU 생성 태스크를 뷰어 패러미터 그대로 사용하여 설정 뷰어에서 사용하는 패러미터를 그대로 사용할 수 있도록 설정g하여, 주로 멀티 폼을 사용하는 보고서를 익스포트 할 때 사용됩니다. none : 보고서 생성 태스크 (ViewType이 None일 경우에만 사용 가능) sdm : SDM 생성 태스크 ex) setProperty("task_type ", "viewerTag")

- 태스크 타임 아웃 설정

Key	Value	설명
task_timeout	타임아웃 (단위 : 분)	태스크 실행 타임아웃 (기본값 : 0) ex) setProperty("task_timeout", "10")

- 태스크 재시도 횟수 설정

Key	Value	설명
task_retry_count	재시도 횟수	태스크 재시도 횟수 (기본값 : 0) ex) setProperty("task_retry_count", "3")

- 외부 프로그램 설정

Key	Value	설명
external_program_check	"check" "null"	패러미터 생성과 파일 경로의 동적 생성을 위한 외부 프로그램 체크 여부 ex) setProperty("external_program_check", "check")

external_program_command	외부 프로그램명	외부 프로그램명 ("SCH_HOME/External"에서의 상대 경로) ex) SetProperty("external_program_command", "notepad.bat")

- 이미지 URL 설정

Key	Value	설명
image_url_size	이미지 url 수	OZD에 저장할 이미지 url 수 ex) setProperty("image_url_size", "1")
image_url_1 **...** **image_url_n**	이미지 url	이미지 url(n : 입력 이미지 url 수) ex) setProperty("image_url_1", "http://127.0.0.1/sample.jpg")

- UDE 클래스 이름 설정

Key	Value	설명
ude.classname	클래스명	UDE를 실행할 클래스명 ex) setProperty("ude.classname", "oz.scheduler.ude.OZUDEImplement")

■ 전자 메일 발송

Key	Value	설명
mail_check	"check" "null"	전자 메일 발송 여부 ex) setProperty("mail_check", "check")
mail_notify_error_check	"check" "null"	수신자에게 에러 통보 여부 선택 ex) setProperty("mail_notify_error_check", "null")
mail_recipient_to	수신자 메일 주소	전자 메일 수신자 ex) setProperty("mail_recipient_to", "oz@forcs.com")
mail_recipient_cc	참조자 메일 주소	참조할 사람 메일 ex) setProperty("mail_recipient_cc", "oz@forcs.com")
mail_recipient_bcc	숨은 참조자 메일 주소	숨은 참조할 사람 메일 ex) setProperty("mail_recipient_bcc", "bbb@forcs.com")

mail_subject	메일 제목	전자 메일 제목 ex) setProperty("mail_subject", "메일제목")
mail_text_message	메일 내용	전자 메일 내용 ex) setProperty("mail_text_message", "메일내용")
mail_html_comment	"check" "null"	전자 메일 내용에 HTML 사용 여부 ex) setProperty("mail_html_comment", "check")
html_mail_content	"check" "null"	전자 메일 내용에 생성한 HTML 파일로 대체 ex) setProperty("html_mail_content", "check")
mail_attach_list	첨부 파일 리스트	전자 메일에 첨부할 스케줄러의 생성된 파일 리스트 (다수 선택이 가능하며 '/'로 구분) ex) setProperty("mail_attach_list", "excel/pdf/word")
mail_attach_zipcheck	"check" "null"	전자 메일에 첨부할 파일을 하나의 파일로 압축하여 전송 ex) setProperty("mail_attach_zipcheck", "check")
mail_attach_zipfilename	파일명	전자 메일에 첨부할 파일을 하나의 파일로 압축하여 전송 시 파일 이름 ex) setProperty("mail_attach_zipfilename", "sales")
mail_attach_zippsswor d	파일 암호	전자 메일에 첨부할 파일을 하나의 파일로 압축하여 전송할 때 파일 암호 ex) setProperty("mail_attach_zippssword", "forcs")

■ 스케줄러 파일 변환

Key	Value	설명
file_export_list	포맷 리스트	변환 파일 리스트 (다수 선택이 가능하며 '/'로 구분) ex) setProperty("file_export_list ", "xls/pdf/tif"

■ 리포트 패러미터

Key	Value	설명
parameter_count	패러미터 수	입력 패러미터 수 ex) setProperty("parameter_count", "1")

parameter_name_1 **...** **parameter_name_n**	패러미터명	입력 패러미터명 (n : 입력 패러미터 수) ex) setProperty("parameter_name_1", "[FORM].empNo")
parameter_value_1 **...** **parameter_value_n**	패러미터 값	입력 패러미터 값 (n : 입력 패러미터 수) ex) setProperty("parameter_value_1", " 10")

※ 참고사항

- 패러미터 수 : 패러미터의 수는 폼과 각각의 ODI에서 사용하는 패러미터 수의 합을 입력 패러미터 수로 설정하고, 그 개수만큼 패러미터 이름과 해당 값을 설정해 주어야 합니다.
- 패러미터명
  - ① 리포트 패러미터명은 "[FORM]." 으로 시작합니다. 예를 들어 empNo라는 패러미터는 "[FORM].empNo"로 입력하여야 합니다.
  - ② ODI 패러미터명은 ODI명으로 시작합니다. 예를 들어 ODI명이 "testodi"이고 패러미터명이 "id"일 경우에는 "testodi.id"로 입력하여야 합니다.

■ **ODI 패러미터**

Key	Value	설명
odi_parameter_ count	패러미터 수	입력 패러미터 수 ex) setProperty("odi_parameter_count", "1")
odi_parameter_name_1 **...** **odi_parameter_name_n**	패러미터명	입력 패러미터명 (n : 입력 패러미터 수) ex) setProperty("odi_parameter_name_1", "id")
odi_parameter_value_1 **...** **odi_parameter_value_n**	패러미터 값	입력 패러미터 값 (n : 입력 패러미터 수) ex) setProperty("odi_parameter_value_1", " 10")

※ 참고사항 : ODI의 패러미터 설정은 서버 데이터 모듈을 저장할 때만 사용되며, ODI는 서버 데이터 모듈을 저장할 때 지정한 ODI를 사용합니다.

■ 패러미터 값 적용 여부

Key	Value	설명

use_ozd_parameter	true/false	OZD 바인딩 시 OZD에 저장된 패러미터 값을 적용할지 여부 ex) setProperty("use_ozd_parameter ", "false")

■ 스케줄 시간

- 실행 타입

Key	Value	설명
launch_type	"once" "immediately" "periodically"	실행 타입 once : 한번 실행 immediately : 즉시 실행 periodically : 주기적 실행 ex) setProperty("launch_type", "once")

▪ launch_ type = once인 경우

Key	Value	설명
execution_year	년도	실행 시간 – 년 ex) setProperty("execution_year", "2005")
execution_month	월	실행 시간 – 월 ex) setProperty("execution_month", "12")
execution_day	일	실행 시간 – 일 ex) setProperty("execution_day", "30")
execution_hour	시간	실행 시간 – 시 ex) setProperty("execution_hour", "10")
execution_min	분	실행 시간 – 분 ex) setProperty("execution_minute", "30")

▪ launch_ type = periodically인 경우

Key	Value	설명
start_year	년도	시작 시간 – 년 ex) setProperty("start_year", "2002")
start_month	월	시작 시간 – 월 ex) setProperty("start_month", "11")
start_day	일	시작 시간 – 일 ex) setProperty("start_day", "16")

periodically_execution_day_type	"daily" "weekly" "monthly"	주기 타입 ex) setProperty("periodically_execution_day_ type", "daily")

▪ periodically_ execution_day_type = daily인 경우

Key	Value	설명
daily_type	weekday	평일 ex) setProperty("daily_type", "weekday")
daily_every_days	간격 일수	설정한 일 간격(며칠) ex) 2일 : setProperty ("daily_every_days", "2")

▪ periodically_ execution_day_type = weekly인 경우

Key	Value	설명
weekly_every_weeks	매주 간격	매주 몇번째 주 ex) setProperty("weekly_every_weeks", "2")
weekly_monday_check	"check" "null"	매주 월요일 ex) setProperty("weekly_monday_check", "check")
weekly_tuesday_check	"check" "null"	매주 화요일 ex) setProperty("weekly_tuesday_check", "check")
weekly_wednesday_check	"check" "null"	매주 수요일 ex) setProperty("weekly_wednesday_check", "check")
weekly_thursday_check	"check" "null"	매주 목요일 ex) setProperty("weekly_thursday_check", "check")
weekly_friday_check	"check" "null"	매주 금요일 ex) setProperty("weekly_friday_check", "check")
weekly_saturday_check	"check" "null"	매주 토요일 ex) setProperty("weekly_saturday_check", "check")
weekly_sunday_check	"check" "null"	매주 일요일 ex) setProperty("weekly_sunday_check", "check")

▪ periodically_ execution_day_type = monthly인 경우

Key	Value	설명

monthly_every_months	월 간격	설정한 달 간격(몇 달) ex) setProperty("monthly_every_months", "2")
monthly_type	"specific_day" , "day_of_week", "user_defined"	특정 일이나 주 ex) setProperty("monthly_type", "specific_day")
monthly_days	"1" ~ "31" "LAST"	1 ~ 31 : 매달 특정 일 LAST : 매달 마지막 일 ex) setProperty("monthly_days", "2")
monthly_which_week	"T1" "T2" "T3"" T4" "T5" "TL"	매달 몇 번째 주 T1 : 매달 첫 번째 주 T2 : 매달 두 번째 주 T3 : 매달 세 번째 주 T4 : 매달 네 번째 주 T5 : 매달 다섯 번째 주 TL : 매달 마지막 주 ex) setProperty("monthly_which_week", "T2")
monthly_which_week_day	"sunday" "monday" "tuesday" "wednesday" "thursday" "friday" "saturday"	매달 특정 요일 ex) 매달 월요일 : setProperty("monthly_which_week_day", "monday")
monthly_user_defined_days	설정할 일 리스트	설정할 일을 오름차순으로 콤마(,)로 연결하여 설정 ex) 매달 1,15일 : setProperty("monthly_user_defined_days", "1,15")

- 실행 횟수

Key	Value	설명
periodically_execution_time_ type	"once" "repeat" "user_defined"	실행 횟수(한번, 여러 번, 사용자 정의) ex) setProperty ("periodically_execution_time_type","once")

▪ periodically_ execution_time_ type = once인 경우

Key	Value	설명

once_hour	시	실행시킬 시간 – 시 ex) setProperty("once_hour", "01")
once_min	분	실행시킬 시간 – 분 ex) setProperty("once_min", "00")

▪ periodically_ execution_time_ type = repeat인 경우

Key	Value	설명
repeat_every_hours	시	주기적으로 실행 간격 – 시 ex) setProperty("repeat_every_hours", "01")
repeat_every_minutes	분	주기적으로 실행 간격 – 분 ex) setProperty("repeat_every_minutes", "04")
repeat_start_hour	시	주기적 실행 시작 시간 – 시 ex) setProperty("repeat_start_hour", "02")
repeat_start_minute	분	주기적 실행 시작 시간 – 분 ex) setProperty("repeat_start_minute", "05")
repeat_end_hour	시	주기적 실행 종료 시간 – 시 ex) setProperty("repeat_end_hour", 09")
repeat_end_minute	분	주기적 실행 종료 시간 – 분 ex) setProperty("repeat_end_minute", "35")

※ 참고사항 : 위 6가지 예제는 2시 5분에서 9시 35분 사이에 1시간 4분 간격으로 태스크 실행하라는 명령을 나타냅니다.

▪ periodically_ execution_time_ type = user_defined인 경우

Key	Value	설명
user_defined_time	설정할 시간 리스트	설정할 시간(시:분)을 오름차순으로 콤마(,)로 연결하여 설정 ex) setProperty("user_defined_time", "01:30,13:30")

■ 파일 저장

"*파일형식*.mailattach" 패러미터는 오즈 엔터 프라이즈 매니저에서만 사용하는 패러미터로 메일 전송 시 첨부 파일로 해당 포맷의 파일을 첨부할지 여부를 설정합니다. true로 설정할 경우에는 파일을 첨부하며, false로 설정할 경우에는 파일을 첨부하지 않습니다.

ex) setProperty("hdm.mailattach", "true")

그 밖의 파일 저장 시 설정할 수 있는 key에 대한 자세한 사항은 오즈 리포트 뷰어 매뉴얼의 "오즈 뷰어 호출 옵션"의 각 파일 형식 저장 관련 패러미터 부분을 참조하시기 바랍니다.

※ 참고사항 : 익스포트 시 Value 값을 "디렉토리이름/파일이름.파일형식"으로 설정할 경우에는 "%스케줄러디렉토리%\Repository/디렉토리이름" 안의 파일이 익스포트됩니다. 예를 들어, "FORCS"라는 디렉토리 안의 파일을 CSV 파일로 익스포트하고자 할 경우 setProperty("csv.filename", "FORCS/test.csv")으로 설정하고, "%스케줄러디렉토리%\Repository/FORCS" 안에 "test.csv" 파일로 저장됩니다.

■ **modifyConfiguration의 ConfigMap**

Key	Value	설명
SchedulerPort	포트 번호	스케줄러 포트 번호 (기본값 : "9521") ex) p.setProperty("SchedulerPort", "9521");
SchedulingInfoFilePath	저장 경로	스케줄링 정보 파일 저장 경로명 ex) p.setProperty("SchedulingInfoFilePath","%SCH_HOME%/ScheduledTask");
SMTPServer	서버 주소	SMTP 서버 주소 ex) p.setProperty("SMTPServer", "mail.forcs.com");
SMTPServerProt	서버 포트 번호	SMTP 서버 포트 번호 ex) p.setProperty("SMTPServerProt", "25");
MailFrom	메일 주소	전자 메일 발송 주소 ex) p.setProperty("MailFrom", "mail@forcs.com");
TempRepositoryFilePath	저장 경로	임시 파일 저장 경로 ex) p.setProperty("TempRepositoryFilePath", "%SCH_HOME%/Te mp Repository");
RepositoryFileRootPath	저장 경로	익스포트 파일 루트 저장 경로 설정 ex) p.setProperty("RepositoryFileRootPath", "%SCH_HOME%/Repository");
ExternalProgramFilePath	저장 경로	외부 프로그램 저장 경로 설정 ex) p.setProperty("ExternalProgramFilePath", "%SCH_HOME%/External");

ErrorNotifyToSender	"true" "false"	관리자에게 에러 통보 여부 설정 • true : 통보함 • false : 통보하지 않음 ex) p.setProperty("ErrorNotifyToSender", "false");

Sample : SchedulerSample.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.ServerInfo;
import oz.util.SortProperties;
import org.apache.log4j.*;

public class SchedulerSample {
   private static String[][] scheduleInfoProperties =
   {
      // Basic Configration
      {"launch_type","Immediately"}, // Task Execute Type
//      {"launch_type","Periodically"},
//      {"start_year","2004"},
//      {"start_month","12"},
//      {"start_day","29"},
//      {"periodically_execution_day_type","daily"},
//      {"daily_type","specific_day"},
//      {"daily_every_days","1"},

//      {"periodically_execution_time_type","repeat"},  //"once", "repeat",
"user_defined"
//      {"repeat_every_hours","00"},
//      {"repeat_every_minutes","01"},
//      {"repeat_start_hour","01"},
//      {"repeat_start_minute","01"},
//      {"repeat_end_hour","23"},
//      {"repeat_end_minute","59"},

//      {"daily_type","specific_day"},
//      {"daily_every_days","1"},
//      {"periodically_execution_time_type","user_defined"},
//      {"user_defined_time","16:20"},

//      {"daily_type","specific_day"},
//      {"daily_every_days","1"},
//      {"periodically_execution_time_type","user_defined"},
```

```
//      {"user_defined_time","16:20"},

      //task holiday
//      {"task_holiday", "common"},

//      {"dm_server_name", "testtesttest.sdm"},
      {"cfg.type","new"}, // Type of Task -> new, edit
//      {"cfg.task_id","parameter_test_060823103556234_296956235.ozs"},
      {"report_name", "parameter_test.ozr"},
      {"category_name","/"},
      {"group_name",""},

      // ODI
      {"odi_name", "parameter_test.odi"},
      {"odi_category_name", "/"},


      //image URL properties
//      {"image_url_size", "1"},
//      {"image_url_1","http://localhost:8088/tomcat.gif"},
//      {"image_url_2","http://localhost:8088/b.gif"},

//      {"option_properties_size", "1"},
//      {"option_property_name_1","use_ozd_parameter"},
//      {"option_property_value_1","true"},


      // Server Data Moudle
      {"dm_server_check", "check"},
      {"dm_server_name", "testtesttest.sdm"},

      {"parameter_count", "4"},

      {"parameter_name_1", "[FORM].formparam1"},
      {"parameter_value_1", "AAAA"},
      {"parameter_name_2", "[FORM].formparam2"},
      {"parameter_value_2", "BBBB"},
      {"parameter_name_3", "parameter_test.odiparam1"},
      {"parameter_value_3", "CCCC"},
      {"parameter_name_4", "parameter_test.odiparam2"},
      {"parameter_value_4", "DDDD"},


//      // It must input for ODI parameter sdm create.
      {"odi_parameter_count", "2"},
      {"odi_parameter_name_1", "odiparam1"},
      {"odi_parameter_value_1", "111"},
      {"odi_parameter_name_2", "odiparam2"},
```

```
      {"odi_parameter_value_2", "222"},
//      {"odi_parameter_name_1", "odiparam1"},
//      {"odi_parameter_name_1", "odiparam2"},

      // e-mail send.
//      {"mail_check", "check"},
//      {"html_mail_content", "check"},
//      {"mail_notify_error_check", "null"},
//      {"mail_recipient_to",  ""},
//      {"mail_recipient_cc", ""},
//      {"mail_recipient_bcc", ""},
//      {"mail_subject", ""},
//      {"mail_text_message", "context"},
//      {"mail_attach_list", "csv/xls/html/ozd/pdf/txt/tif/doc/ppt/jpg/svg"},
      {"file_export_list", "csv/xls/html/ozd/pdf/txt/tif/doc/ppt/jpg/svg"}

   };

   private static String[][] exportedProperties =
   {

      {"csv.filename ", "test.csv"}, // csv file name
      {"xls.filename ", "test.xls"}, // excel file name
      {"html.filename ", "test.html"}, // html file name
      {"ozd.filename ", "newimage.ozd"}, //OZD file name
      {"pdf.filename ", "test.pdf"},//, //pdf file name
      {"txt.filename ", "test.txt"}, //text file name
      {"tif.filename ", "test.tif"},//tiff file name
      {"doc.filename", "test.doc"}, //word file name
      {"ppt.filename ", "test.ppt"}, //ppt file name
      {"jpg.filename ", "test.jpg"}, //jpg file name
      {"svg.filename ", "test.svg"} //svg file name

   };

   public static void main(String[] args)
   {

      BasicConfigurator.configure();
      String IP = "127.0.0.1"; //scheduler server ip
      int PORT = 9521; //scheduler server port
      String[][] values = null;
      Scheduler scheduler = null;
      String taskID = null;

      try {
         scheduler = new Scheduler(IP, PORT);
         // scheduler Server Info
```

```
          ServerInfo serverInfo = new ServerInfo();

          serverInfo.setIP("127.0.0.1"); //OZ Server IP conf. In case use of
daemon
          serverInfo.setPortNo(8003);
          serverInfo.setIsDaemon(true); //OZ Server Type. -> daemon, servlet

          serverInfo.setID("admin"); //id
          serverInfo.setPWD("admin"); //pw

          SortProperties p = new SortProperties();
          setProperties(p, scheduleInfoProperties);
         // p.setProperty("task_type", "sdm");

          SortProperties exportMap = new SortProperties();
          setProperties(exportMap, exportedProperties);

          taskID = scheduler.createTask(serverInfo, p, exportMap);

          System.out.println("taskid=="+taskID);

       }
       catch (Exception e) {
          e.printStackTrace();
       }

   }

   private static void setProperties(SortProperties p, String[][] values)
   {
       for(int i=0; i<values.length; i++)
       {
          p.setProperty(values[i][0], values[i][1]);
       }
   }

}
```

Sample : SchedulerSampleTaskExportMakePrintExport.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.ServerInfo;
import oz.scheduler.ScheduledTask;
```

```
import oz.util.SortProperties;
import org.apache.log4j.*;
import oz.scheduler.SchedulerException;


public class SchedulerSampleTaskExportMakePrintExport {
   public SchedulerSampleTaskExportMakePrintExport() {
   }

   public static void main(String[] args) throws SchedulerException {
      BasicConfigurator.configure();

      Scheduler  scheduler = null;
      scheduler = new Scheduler("127.0.0.1", 9521);

      // 스케줄러 서버의 정보
      ServerInfo serverInfo = new ServerInfo();

      serverInfo.setID("admin");   //아이디
      serverInfo.setPWD("admin");  //패스워드
      serverInfo.setIsDaemon(true);

      serverInfo.setIP("127.0.0.1"); //OZ Server IP 설정. daemon 일 때만 사용
      serverInfo.setPortNo(8003);

      SortProperties config = new SortProperties();
      //
      config.setProperty("task_type", "viewerTag");   // 뷰어 태그 사용
      config.setProperty("cfg.type", "new");        // Task 의 타입 new, edit


      SortProperties printexport = new SortProperties();
      // 뷰어 태그
      printexport.setProperty("connection.server", "127.0.0.1");
      printexport.setProperty("connection.port", "8003");
      printexport.setProperty("connection.reportName",
"/parameter_test.ozr");
      printexport.setProperty("applet.useprogressbar", "false");
      printexport.setProperty("applet.allowmultiframe", "true");

      printexport.setProperty("viewer.mode", "print");
      printexport.setProperty("print.mode", "silent");
      printexport.setProperty("print.ignoreerror", "false");

      printexport.setProperty("connection.fetchtype", "BATCH");
      printexport.setProperty("information.debug", "debug");
      printexport.setProperty("applet.showerrormessage", "false");
```

```
        // 보고서를 여러 개 추가할 경우
        printexport.setProperty("viewer.childcount", "1");
        printexport.setProperty("child1.connection.server", "127.0.0.1");
        printexport.setProperty("child1.connection.port", "8003");
        printexport.setProperty("child1.connection.reportName",
"/parameter_test.ozr");
        printexport.setProperty("child1.viewer.applet.mode", "print");
        printexport.setProperty("child1.print.mode" ,"silent");
        printexport.setProperty("child1.print.ignoreerror" ,"false");
        printexport.setProperty("child1.print.pagerange" ,"range");
        printexport.setProperty("child1.print.pages" ,"2");

//      config.setProperty("launch_type", "immediately");// 한번
실행(즉시"immediately", 주기적"periodically")
        config.setProperty("launch_type", "periodically");
        config.setProperty("start_year", "2009");
        config.setProperty("start_month", "03");
        config.setProperty("start_day", "01");
        config.setProperty("start_hour", "06");
        config.setProperty("start_min", "42");
        config.setProperty("periodically_execution_day_type", "daily");
        config.setProperty("daily_type", "specific_day");
        config.setProperty("daily_every_days", "1");
//      config.setProperty("periodically_execution_time_type", "once");
//      config.setProperty("once_hour", "09");
//      config.setProperty("once_min", "41");

        config.setProperty("periodically_execution_time_type", "repeat");
        config.setProperty("repeat_every_hours", "00");
        config.setProperty("repeat_every_minutes", "01");
        config.setProperty("repeat_start_hour", "09");
        config.setProperty("repeat_start_minute", "01");
        config.setProperty("repeat_end_hour", "23");
        config.setProperty("repeat_end_minute", "01");

        String taskID = scheduler.createTask(serverInfo, config, printexport);
    }
}
```

Sample : SchedulerViewerTagSample.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.TaskResult;
```

```
import oz.scheduler.ServerInfo;
import oz.scheduler.ScheduledTask;
import oz.util.SortProperties;
import java.util.Vector;
import java.io.*;
import org.apache.log4j.*;
import oz.scheduler.SchedulerException;

public class SchedulerViewerTagSample {

   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      Scheduler  scheduler = null;
      scheduler = new Scheduler("127.0.0.1", 9521);

      // 스케줄러 서버의 정보
      ServerInfo serverInfo = new ServerInfo();
      serverInfo.setID("admin");   //아이디
      serverInfo.setPWD("admin");  //패스워드
      serverInfo.setIsDaemon(true);
      serverInfo.setIP("127.0.0.1"); //OZ Server IP 설정. daemon 일 때만 사용
      serverInfo.setPortNo(8003);

      SortProperties config = new SortProperties();

      // 새로운 방식으로 뷰어의 태그를 직접 사용할 수 있다.
      config.setProperty("task_type", "viewerTag");

      // Task 의 타입 new, edit
      config.setProperty("cfg.type", "new");

      // 스케쥴 시간한번실행(즉시:"immediately", 주기적:"periodically")
      config.setProperty("launch_type", "immediately");

      SortProperties export = new SortProperties();

      // 뷰어 태그
      export.setProperty("connection.server", "127.0.0.1");
      export.setProperty("connection.port", "8003");
      export.setProperty("connection.reportName", "/parameter_test.ozr");
      export.setProperty("applet.mode", "export");
      export.setProperty("applet.useprogressbar", "false");
      export.setProperty("applet.allowmultiframe", "true");
      export.setProperty("connection.pcount", "2");
      export.setProperty("connection.args1", "formparam1=대");
      export.setProperty("connection.args2", "formparam2=한");
```

```
export.setProperty("export.mode", "silent");

// 하나의 파일로 익스포트
export.setProperty("export.saveonefile", "true");
export.setProperty("connection.fetchtype", "BATCH");
export.setProperty("export.confirmsave", "false");
export.setProperty("information.debug", "debug");
export.setProperty("applet.showerrormessage", "false");
export.setProperty("odi.parameter_test.pcount", "2");
export.setProperty("odi.parameter_test.args1", "odiparam1=민");
export.setProperty("odi.parameter_test.args2", "odiparam2=국");
export.setProperty("odi.odinames", "parameter_test");
export.setProperty("export.format",
                  "csv/xls/html/ozd/pdf/txt/tif/doc/ppt/jpg/svg/mht");
export.setProperty("csv.filename", "1.csv");
export.setProperty("excel.filename", "1.xls");
export.setProperty("html.filename", "1.html");
export.setProperty("ozd.filename", "1.ozd");
export.setProperty("pdf.filename", "1.pdf");
export.setProperty("text.filename", "1.txt");
export.setProperty("tiff.filename", "1.tif");
export.setProperty("word.filename", "1.doc");
export.setProperty("ppt.filename", "1.ppt");
export.setProperty("jpg.filename", "1.jpg");
export.setProperty("svg.filename", "1.svg");
export.setProperty("mht.filename", "1.mht");
export.setProperty("hdm.filename", "1.hdm");

// 추가로 멀티폼에 대한 정보
export.setProperty("viewer.childcount", "1");
export.setProperty("child1.connection.server", "127.0.0.1");
export.setProperty("child1.connection.port", "8003");
export.setProperty("child1.connection.reportName",
                                    "/parameter_test.ozr");
export.setProperty("child1.applet.mode", "export");
export.setProperty("child1.applet.useprogressbar", "false");
export.setProperty("child1.applet.allowmultiframe", "true");
export.setProperty("child1.export.mode", "silent");
export.setProperty("child1.connection.fetchtype", "BATCH");
export.setProperty("child1.export.confirmsave", "false");
export.setProperty("child1.information.debug", "debug");
export.setProperty("child1.applet.showerrormessage", "false");

export.setProperty("child1.connection.pcount", "2");
export.setProperty("child1.connection.args1", "formparam1=대 1");
export.setProperty("child1.connection.args2", "formparam2=한 2");

export.setProperty("child1.odi.odinames", "parameter_test");
```

```
        export.setProperty("child1.odi.parameter_test.pcount", "2");
        export.setProperty("child1.odi.parameter_test.args1", "odiparam1=민 3");
        export.setProperty("child1.odi.parameter_test.args2", "odiparam2=국 4");

        // 최초 익스포트 정보와 동일하게 설정한다.
        export.setProperty("child1.export.format",
                           "csv/xls/html/ozd/pdf/txt/tif/doc/ppt/jpg/svg/mht");
        export.setProperty("child1.csv.filename", "child_1.csv");
        export.setProperty("child1.excel.filename", "child_1.xls");
        export.setProperty("child1.html.filename", "child_1.html");
        export.setProperty("child1.ozd.filename", "child_1.ozd");
        export.setProperty("child1.pdf.filename", "child_1.pdf");
        export.setProperty("child1.text.filename", "child_1.txt");
        export.setProperty("child1.tiff.filename", "child_1.tif");
        export.setProperty("child1.word.filename", "child_1.doc");
        export.setProperty("child1.ppt.filename", "child_1.ppt");
        export.setProperty("child1.jpg.filename", "child_1.jpg");
        export.setProperty("child1.svg.filename", "child_1.svg");
        export.setProperty("child1.mht.filename", "child_1.mht");
        export.setProperty("child1.hdm.filename", "child_1.hdm");

        // 이 방식은 API 로만 가능하며 EM 으로는 불가능합니다.
        String taskID = scheduler.createTask(serverInfo, config, export);

    }
}
```

Sample : SchedulerExportOZDsample.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.ServerInfo;
import oz.util.SortProperties;
import org.apache.log4j.*;

public class SchedulerExportOZDsample {
    private static String[][] scheduleInfoProperties = {
{"task_type","none"},
        {"cfg.type","new"}, // Type of Task -> new, edit
{"reportname", "/sample.ozr"},
//      {"category_name","/"},
//      {"group_name",""},
//      {"odi_name", "sample"},
//      {"odi_category_name", "/"},
```

```
       {"file_export_list", "ozd"},

       {"export.confirmsave", "false"},
       {"TempRepositoryFilePath", "%SCH_HOME%/TempRepository"}, //임시 파일 저장
경로
       {"RepositoryFileRootPath", "%SCH_HOME%/Repository"},

{"parameter_count", "2"},
       {"parameter_name_1", "[FORM].formparam"},
       {"parameter_value_1", "value1"},
       {"parameter_name_2", "sample.odiparam"},
       {"parameter_value_2", "value2"},

       {"image_url_size", "2"},
       {"image_url_1", "file://D:\\Repository\\img1.jpg"},
       {"image_url_2", "file://D:\\pictures\\img2.jpg"},

      // 외부 프로그램
      {"external_program_check", "check"},
      {"external_program_command","external.bat"},

       {"reportcount", "2"},
       {"child1.reportname", "/orders.ozr"},
     {"child1.displayname", "orders"},
       {"child1.odi_name", "/orders"},
     {"child1.odi_category_name", "/"},

       {"child1.parameter_count", "2"},
       {"child1.parameter_name_1", "[FORM].formparam"},
       {"child1.parameter_value_1", "value1"},
       {"child1.parameter_name_2", "sample.odiparam"},
       {"child1.parameter_value_2", "value2"},

       // 외부 프로그램
       {"child1.external_program_check", "check"},
       {"child1.external_program_command","external.bat"},

// 외부 프로그램
{"child1.option_properties_size", "1"},
       {"child1.option_property_name_1","use_ozd_parameter"},
       {"child1.option_property_value_1","true"},

      // 스케쥴 시간
       {"launch_type", "periodically"},
//한번실행(즉시"immediately",주기적"periodically")

       {"start_year", "2009"},
       {"start_month", "03"},
```

```
    {"start_day", "01"},
    {"start_hour", "06"},
    {"start_min", "42"},
    {"periodically_execution_day_type", "daily"},
    {"daily_type", "specific_day"},
    {"daily_every_days", "1"},
    {"periodically_execution_time_type", "repeat"},
    {"repeat_every_hours", "00"},
    {"repeat_every_minutes", "01"},
    {"repeat_start_hour", "09"},
    {"repeat_start_minute", "01"},
    {"repeat_end_hour", "23"},
    {"repeat_end_minute", "01"},

    {"mail_check", "check"},
    {"mail_notify_error_check", "null"},
    {"mail_recipient_to", "sblee@forcs.com"},
    {"mail_recipient_cc", ""},
    {"mail_recipient_bcc", ""},
    {"mail_subject", "aaa"},
    {"mail_text_message", "message"},
    {"mail_html_comment", "comment"},
    {"html_mail_content", "content"},
    {"mail_attach_list", "ozd"}
};

private static String[][] exportedProperties = {
    {"ozd.filename", "test11.ozd"}, //pdf file name
    {"ozd.mailattach", "true"}, //메일첨부여부
    {"ozd.password", "aaaa"}, //비밀번호
    {"ozd.memoallowed", "false"}, //
    {"ozd.includeimage", "false"} //
};

public static void main(String[] args) {

    BasicConfigurator.configure();
    String IP = "127.0.0.1"; //scheduler server ip
    int PORT = 9521; //scheduler server port
    Scheduler scheduler = null;
    boolean isDaemon = true;
    try {
        scheduler = new Scheduler(IP, PORT);

        // scheduler Server Info
        ServerInfo serverInfo = new ServerInfo();
        if(isDaemon){
```

```
          serverInfo.setIP("127.0.0.1"); //OZ Server IP conf. In case use of
daemon
          serverInfo.setPortNo(8003);
          serverInfo.setIsDaemon(true);//OZ Server Type. -> daemon, servlet
          }else{
              serverInfo.setIsDaemon(false);
              serverInfo.setURL("http://127.0.0.1:8080/oz60/server");
          }

          serverInfo.setID("admin"); //id
          serverInfo.setPWD("admin"); //pw

          SortProperties props = new SortProperties();
          setProperties(props, scheduleInfoProperties);

          SortProperties exportMap = new SortProperties();
          setProperties(exportMap, exportedProperties);

          scheduler.createTask(serverInfo,props,exportMap);
       }
       catch(Exception e)
       {
          e.printStackTrace();
       }
    }

    private static void setProperties(SortProperties p, String[][] values) {
       for(int i=0; i<values.length; i++) {
          p.setProperty(values[i][0], values[i][1]);
       }
    }
}
```

Sample : SchedulerTaskResult.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.TaskResult;
import oz.scheduler.ServerInfo;
import oz.scheduler.ScheduledTask;


public class SchedulerTaskResult {
    public SchedulerTaskResult() {
```

```
    }

    public static void main(String[] args)
    {

        String IP = "127.0.0.1"; //scheduler server ip
        int PORT = 9521; //scheduler server port
        Scheduler scheduler = null;

        try {
            scheduler = new Scheduler(IP, PORT);

            // scheduler Server Info
            ServerInfo serverInfo = new ServerInfo();

            serverInfo.setIP("127.0.0.1"); //OZ Server IP conf. In case use of
daemon
            serverInfo.setPortNo(8003);
            serverInfo.setIsDaemon(true);//OZ Server Type. -> daemon, servlet

            serverInfo.setID("admin"); //id
            serverInfo.setPWD("admin"); //pw

            ScheduledTask[] taskList = scheduler.getTask(serverInfo);
            for (int k = 0; k < taskList.length; k++) {
                showTask(taskList[k]);

                System.out.println("\n\n");

                TaskResult tr = new TaskResult();
                tr.taskID = taskList[k].taskID;

                String from = ""; //Task Result Start time
                String to = ""; //Task Result Endt time
                boolean isComplete = false;

                TaskResult[] trList = scheduler.getTaskResult(serverInfo,
                    from, to, tr.taskID);
                if (trList.length > 0) {
                    for (int i = 0; i < trList.length; i++) {
                        showTaskResult(trList[i]);
                    }
                    isComplete = true;
                }
            }
        }
        catch(Exception e)
        {
```

```
            e.printStackTrace();
        }
    }

    private static void showTask(ScheduledTask t)
    {
        System.out.println("TASK ID : " + t.taskID);
        System.out.println("TASK Name : " + t.taskName);
        System.out.println("TASK Group Name : " + t.taskGroupName);
        System.out.println("Report Name : " + t.reportName);
        System.out.println("Type : " + t.schedulingTypeStr);
        System.out.println("Finish Execute Time : " + t.lastRunTimeStr);
        System.out.println("Next Execute Time : " + t.nextRunTimeStr);
        System.out.println("Status : " + t.status);
        System.out.println("");
    }

    private static void showTaskResult(TaskResult tr)
    {
        System.out.println("TASK ID : " + tr.taskID);
        System.out.println("TASK Name : " + tr.taskName);
        System.out.println("TASK Group Name : " + tr.taskGroupName);
        System.out.println("TASK FINISH TIME : " + tr.completedTime);
        System.out.println("Is Sucessed Code : " + tr.isSuccessfulCode);
        System.out.println("Is Sucessed? : " + tr.isSuccessful);
        System.out.println("Report Name : " + tr.formFileName);
        System.out.println("Report Category Name : " + tr.formCategoryName);
        System.out.println("Parameter : " + tr.paramInfo);
        System.out.println("Type : " + tr.schedulingType);
        System.out.println("Export File List : " + tr.exportFileList);
        System.out.println("Error Message : " + tr.errorMsg);
        System.out.println("");
    }
}
```

Sample : SchedulerTaskModify.java

```
package sample;

import org.apache.log4j.*;
import oz.framework.api.*;
import oz.scheduler.ServerInfo;
import oz.scheduler.ScheduledTask;
import oz.util.SortProperties;
```

```
public class SchedulerTaskModify {

   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

     //스케줄러 정보
      String IP = "127.0.0.1"; // 스케줄러 IP
      int PORT = 9521; // 스케줄러 Port

      Scheduler scheduler = null;
      String taskID = null;

      try {
         scheduler = new Scheduler(IP, PORT);

         // OZ Server 정보
         ServerInfo serverInfo = new ServerInfo();

         serverInfo.setIP("127.0.0.1"); //오즈 서버 IP
         serverInfo.setPortNo(8003); // 오즈 서버 Port
         serverInfo.setIsDaemon(true);// 데몬 서버

         serverInfo.setID("admin"); // 오즈 서버 ID
         serverInfo.setPWD("admin"); // 오즈 서버 Password

         // 기존에 생성한 taskID
         // task 를 수정을 하기위해서는 기존에 생성된 태스크의 task_type 은 viewerTag
가 아니어야 한다.
         taskID = "parameter_test_061031115221500_263141500.ozs";

         // 태스크의 정보를 가지고 온다.
         SortProperties[] props = scheduler.getTaskProperties(serverInfo,
taskID);

         // 기존의 txt 파일로 Export 하는 Task 에 csv 포맷으로도 export 하도록 속성을
변경.
props[0].setProperty("user_defined_time", "17:53");
         props[0].setProperty("file_export_list", "csv/txt");
         props[1].setProperty("csv.filename", "modify_task.csv");

         // 변경된 태스크 출력

System.out.println("config.....................................");
         props[0].list(System.out);


System.out.println("export.....................................");
```

```
            props[1].list(System.out);

            // 모든 태스크 목록
            ScheduledTask[] taskList = scheduler.getTask(serverInfo);
            for (int i = 0; i < taskList.length; i++) {
               if (taskID.endsWith(taskList[i].taskID)) {
                  while(true) {
                     if (taskList[i].status != 'R') { //실행중이 아닌 경우
                        // 새로운 값으로 수정함
                        scheduler.modifyTask(serverInfo, taskID, props[0],
props[1]);
                        System.out.println("modify ok...");
                        break;
                     }
                     Thread.sleep(1000);
                  }
               }
            }
         } catch(Exception e) {
            e.printStackTrace();
         }
      }
}
```

Sample : SchedulerPrint.java

```
package sample;

import org.apache.log4j.BasicConfigurator;
import oz.framework.api.Scheduler;
import oz.scheduler.SchedulerException;
import oz.scheduler.ServerInfo;
import oz.util.SortProperties;

public class SchedulerPrint {

   public static void main(String[] args) {
      //아래 라인을 주석 처리하면 사용자가 지정하지 않은 로그가 나타나지 않습니다.
      BasicConfigurator.configure();

      //스케줄러 정보
      String IP = "127.0.0.1"; // 스케줄러 IP
      int PORT = 9521; // 스케줄러 Port

      try {
```

```
          Scheduler scheduler = new Scheduler(IP, PORT);

          // OZ Server 정보
          ServerInfo serverInfo = new ServerInfo();
          serverInfo.setIP("127.0.0.1"); //오즈 서버 IP
          serverInfo.setPortNo(8003); // 오즈 서버 Port
          serverInfo.setIsDaemon(true); // 데몬 서버
          serverInfo.setID("admin"); // 오즈 서버 ID
          serverInfo.setPWD("admin"); // 오즈 서버 Password

          SortProperties config = new SortProperties();

          // print 기능을 사용하기 위해서는 반드시 task_type 은 viewerTag 이어야 한다.
          config.setProperty("task_type", "viewerTag");
          config.setProperty("launch_type", "immediately");

          SortProperties printMap = new SortProperties();
          printMap.setProperty("connection.server", "127.0.0.1");
          printMap.setProperty("connection.port", "8003");
          printMap.setProperty("connection.reportName",
"/parameter_test.ozr");
          printMap.setProperty("repository_agent.try_license_check", "true");
          printMap.setProperty("connection.pcount", "2");
          printMap.setProperty("connection.args1", "formparam1=대");
          printMap.setProperty("connection.args2", "formparam2=한");
          printMap.setProperty("connection.fetchtype", "BATCH");
          printMap.setProperty("odi.parameter_test.pcount", "2");
          printMap.setProperty("odi.parameter_test.args1", "odiparam1=민");
          printMap.setProperty("odi.parameter_test.args2", "odiparam2=국");
          printMap.setProperty("odi.odinames", "parameter_test");

          //print 속성(자세한 설명은 리포트 뷰어 매뉴얼 참조)
          printMap.setProperty("print.adjust", "true");
          printMap.setProperty("print.alldocument", "false");
          printMap.setProperty("print.close", "true");
          printMap.setProperty("print.collate", "true");
          printMap.setProperty("print.copies", "1");
          printMap.setProperty("print.duplex", "none");
          printMap.setProperty("print.gray", "true");
          printMap.setProperty("print.ignoreerror", "true");
          printMap.setProperty("print.lockopt", "true");
          printMap.setProperty("print.once", "false");
          printMap.setProperty("print.orientation", "default");
          printMap.setProperty("print.pageorient", "horizontal");
          printMap.setProperty("print.pagerange", "all");
          printMap.setProperty("print.printbypage", "true");
          printMap.setProperty("print.printername", "default");
          printMap.setProperty("print.scaling", "100");
```

```
          printMap.setProperty("print.size", "A4");
          printMap.setProperty("print.spoolpages", "100");
          printMap.setProperty("print.style", "normal");
          printMap.setProperty("print.usedefaultpaper", "false");
          printMap.setProperty("print.spoolpages", "100");
          printMap.setProperty("print.usedialogopt", "true");

          // 다중 보고서 출력
          printMap.setProperty("viewer.childcount", "1");
          printMap.setProperty("child1.connection.server", "127.0.0.1");
          printMap.setProperty("child1.connection.port", "8003");
          printMap.setProperty("child1.connection.reportName",
"/module_sample.ozr");
          printMap.setProperty("child1.connection.fetchtype", "BATCH");

          // 이 방식은 API 로만 가능하며 EM 으로는 불가능합니다.
         boolean result = scheduler.print(serverInfo, config, printMap);
          if(result == true) {
                System.out.println("Success Print");
          } else {
                System.out.println("Failed Print");
          }
       } catch (SchedulerException se) {
          se.printStackTrace();
       }
    }

}
```

Sample : SchedulerTaskConverter.java

```
package sample;
import org.apache.log4j.BasicConfigurator;
import oz.framework.api.Scheduler;

public class SchedulerTaskConverter {
         public static void main(String[] args) {
         BasicConfigurator.configure();
       String IP = "127.0.0.1"; //scheduler server ip
       int PORT = 9521; //scheduler server port

       Scheduler scheduler = null;

       try {
          scheduler = new Scheduler(IP, PORT);
```

```
            // /%SCH_HOME%/ScheduledTask 의 하위 디렉토리의 모든 ozs 파일 목록을 가져옴
(변환할 OZS 파일 목록 가져옴)
            String[] ozsList = scheduler.getOZSList("/ScheduledTask");

            for(int i=0 ; i < ozsList.length ; i++) {
                  System.out.println(ozsList[i]);
            }

            // /%SCH_HOME%/ScheduledTask 의 하위 디렉토리의 모든 ozs 파일을
/%SCH_HOME%/ConvertedTask 에 변환하여 생성함
            scheduler.ozsVerConv("/ScheduledTask", "/ConvertTask");

            // /%SCH_HOME%/ ConvertTask 의 하위 디렉토리의 모든 ozs 파일 목록을 가져옴
(변환된 OZS 파일 목록 가져옴)
            ozsList = scheduler.getOZSList("/ConvertTask");
            for(int i=0 ; i < ozsList.length ; i++) {
                  System.out.println(ozsList[i]);
            }
      }
      catch(Exception e) {
         e.printStackTrace();
      }
        }
}
```

UseSCOzdParameterSample.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.TaskResult;
import oz.scheduler.ServerInfo;
import oz.scheduler.ScheduledTask;
import oz.util.SortProperties;
import java.util.Vector;
import java.io.*;
import org.apache.log4j.*;

public class UseSCOzdParameterSample {
   private static String[][] scheduleInfoProperties =
   {
      // Basic Configration
      {"launch_type","Immediately"}, // Task Execute Type
```

```
    {"cfg.type","new"}, // Type of Task -> new, edit
    {"report_name", "parameter_test.ozr"},
    {"category_name","/"},
    {"group_name",""},

    // ODI
    {"odi_name", "parameter_test.odi"},
    {"odi_category_name", "/"},


    {"option_properties_size", "1"},
    {"option_property_name_1","use_ozd_parameter"},
    {"option_property_value_1","true"},


     {"parameter_count", "4"},

    {"parameter_name_1", "[FORM].formparam1"},
    {"parameter_value_1", "홍길동1"},
    {"parameter_name_2", "[FORM].formparam2"},
    {"parameter_value_2", "홍길동2"},
    {"parameter_name_3", "parameter_test.odiparam1"},
    {"parameter_value_3", "홍길동3"},
    {"parameter_name_4", "parameter_test.odiparam2"},
    {"parameter_value_4", "홍길동4"},
    {"file_export_list", "ozd"}

};

private static String[][] exportedProperties =
{

    {"ozd.filename ", "newimage.ozd"}, //OZD file name
};

public static void main(String[] args)
{

    BasicConfigurator.configure();
    String IP = "127.0.0.1"; //scheduler server ip
    int PORT = 9521; //scheduler server port
    String[][] values = null;
    Scheduler scheduler = null;
```

```
      String taskID = null;

      try {
         scheduler = new Scheduler(IP, PORT);
         for (int i = 0; i < 1; i++) {
            // scheduler Server Info
            ServerInfo serverInfo = new ServerInfo();

            serverInfo.setIP("127.0.0.1");  //OZ  Server  IP  conf.  In
case use of daemon
            serverInfo.setPortNo(8003);
            serverInfo.setIsDaemon(true); //OZ Server Type. -> daemon,
servlet

            serverInfo.setID("admin"); //id
            serverInfo.setPWD("admin"); //pw

            SortProperties p = new SortProperties();
            setProperties(p, scheduleInfoProperties);

            SortProperties exportMap = new SortProperties();
            setProperties(exportMap, exportedProperties);

            taskID = scheduler.createTask(serverInfo, p, exportMap);

            System.out.println("ii=="+i+"     taskid=="+taskID);
         }
      }
      catch (Exception e) {
         e.printStackTrace();
      }

   }

   private  static  void  setProperties(SortProperties  p,  String[][]
values)
   {
      for(int i=0; i<values.length; i++)
      {
         p.setProperty(values[i][0], values[i][1]);
      }
   }
}
```

Sample : DirectExportResultSample.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.DirectExportResult;
import oz.scheduler.ServerInfo;
import oz.util.SortProperties;

public class DirectExportResultSample {
    public DirectExportResultSample() {
    }

    static final String IP = "127.0.0.1";
    static final int PORT = 8003;
    static final String URL = "http://localhost:8088/oz/server";
    static final boolean isDaemon = true;

    static final String ID = "admin";
    static final String PWD = "admin";

    static final String schedulerIP = "127.0.0.1";
    static final int schedulerPORT = 9521;

    public static void main(String[] args) {

        Scheduler scheduler = null;
        try {

            scheduler = new Scheduler(schedulerIP, schedulerPORT);

            // 스케줄러 서버의 정보
            ServerInfo serverInfo = new ServerInfo();
            serverInfo.setID(ID); //아이디
            serverInfo.setPWD(PWD); //패스워드
            serverInfo.setIsDaemon(isDaemon);

            if (serverInfo.getIsDaemon()) {
                serverInfo.setIP(IP); //OZ Server IP 설정. daemon 일 때만 사용
                serverInfo.setPortNo(PORT);
            }
            else {
                serverInfo.setURL(URL);
            }
            SortProperties props = new SortProperties();
            props.setProperty("task_type", "viewerTag");
```

```
            props.setProperty("cfg.type", "new");
            props.setProperty("launch_type", "immediately");


            SortProperties export = new SortProperties();

            export.setProperty("connection.server", "127.0.0.1");
            export.setProperty("connection.port", "8003");
            export.setProperty("connection.reportName", "/parameter_test.ozr");
            export.setProperty("viewer.mode", "export");
            export.setProperty("export.mode", "silent");
            export.setProperty("viewer.isframe", "true");
            export.setProperty("viewer.mode", "silent");
            export.setProperty("export.confirmsave", "false");
            export.setProperty("export.format", "gul");
            export.setProperty("gul.filename", "sample.gul");

            DirectExportResult result = scheduler.directExport(serverInfo,
props, export);
            showDirectExportResult(result);

         }
         catch (Exception e) {
            e.printStackTrace();
         }
      }

      private static void showDirectExportResult(DirectExportResult t) {
         System.out.println("태스크 아이디 :" + t.taskID);
         System.out.println("태스크 이름 :" + t.taskName);
         System.out.println("태스크 그룹이름 :" + t.taskGroupName);
         System.out.println("완료시간 :" + t.completedTime);
         System.out.println("수행시간 : " + t.executeTime + "sec");
         System.out.println("성공여부 : " + t.isSuccessful);
         System.out.println("보고서 이름 : " + t.formName);
         System.out.println("보고서 카테고리 이름 : " + t.formCategoryName);
         System.out.println("export 파일리스트 : " + t.exportFileList);
         System.out.println("익스포트된 페이지 수 : " + t.pageCount);

         System.out.println("");
      }
}
```

Sample : DirectPrintResultSample.java

```
package sample;

import oz.framework.api.*;
import oz.scheduler.DirectPrintResult;
import oz.scheduler.ServerInfo;
import oz.util.SortProperties;

public class DirectPrintResultSample {
   public DirectPrintResultSample() {
   }

   static final String IP = "127.0.0.1";
   static final int PORT = 8003;
   static final String URL = "http://localhost:8088/oz35/server";
   static final boolean isDaemon = true;

   static final String ID = "admin";
   static final String PWD = "admin";

   static final String schedulerIP = "127.0.0.1";
   static final int schedulerPORT = 9521;

   public static void main(String[] args) {

      Scheduler scheduler = null;
      try {

         scheduler = new Scheduler(schedulerIP, schedulerPORT);

         // 스케줄러 서버의 정보
         ServerInfo serverInfo = new ServerInfo();
         serverInfo.setID(ID); //아이디
         serverInfo.setPWD(PWD); //패스워드
         serverInfo.setIsDaemon(isDaemon);

         if (serverInfo.getIsDaemon()) {
            serverInfo.setIP(IP); //OZ Server IP 설정. daemon 일 때만 사용
            serverInfo.setPortNo(PORT);
         }
         else {
            serverInfo.setURL(URL);
         }
         SortProperties props = new SortProperties();
         props.setProperty("task_type", "viewerTag");
         props.setProperty("cfg.type", "new");
         props.setProperty("launch_type", "immediately");
```

```
          SortProperties print = new SortProperties();

          print.setProperty("connection.server","127.0.0.1");
          print.setProperty("connection.port" ,"8003");
          print.setProperty("connection.id" ,"admin");
          print.setProperty("connection.password" ,"admin");

          print.setProperty("connection.reportname" ,"/parameter_test.ozr");
          print.setProperty("viewer.configmode" ,"html");
          print.setProperty("odi.odinames" ,"parameter_test");
          print.setProperty("connection.pcount" ,"2");
          print.setProperty("connection.args1" ,"formparam2=대한민국");
          print.setProperty("connection.args2" ,"formparam1=대한이");
          print.setProperty("odi.parameter_test.pcount" ,"2");

print.setProperty("odi.parameter_test.args1" ,"odiparam1=odiValue1111111");

print.setProperty("odi.parameter_test.args2" ,"odiparam2=odiValue22222222");
//          print.setProperty("applet.isframe" ,"false");
          print.setProperty("connection.clientdmtype" ,"Memory");
          print.setProperty("connection.serverdmtype" ,"Memory");
          print.setProperty("connection.fetchtype" ,"Concurrent");
          print.setProperty("viewer.mode" ,"print");
          print.setProperty("print.mode" ,"silent");
          print.setProperty("viewer.printcommand", "true");

          DirectPrintResult result = scheduler.directPrint(serverInfo, props,
print);
          showDirectExportResult(result);

      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }

   private static void showDirectExportResult(DirectPrintResult t) {
      System.out.println("태스크 아이디 :" + t.taskID);
      System.out.println("태스크 이름 :" + t.taskName);
      System.out.println("태스크 그룹이름 :" + t.taskGroupName);

      System.out.println("완료시간 :" + t.completedTime);
      System.out.println("수행시간 : " + t.executeTime + "sec");
      System.out.println("성공여부 : " + t.isSuccessful);
      System.out.println("보고서 이름 : " + t.formName);
      System.out.println("보고서 카테고리 이름 : " + t.formCategoryName);

      System.out.println("페이지 수 : " + t.pageCount);
```

```
        System.out.println("인쇄 매수 : " + t.pageCopy);
        System.out.println("페이지 범위 : " + t.pageRange);
        System.out.println("프린터 이름 : " + t.printerName);
        System.out.println("프린터 드라이버 이름 : " + t.printerDriverName);

        System.out.println("");
    }
}
```

Class TaskHolidayInfo

Constructor Summary

- **TaskHolidayInfo(String name)**

Method Summary

- **String getName()**
- **void setbaseDay(String pattern)**
- **String getBaseDay()**
- **void setType(String type)**
- **String getType()**
- **boolean isLunar()**
- **void setStart(int start)**
- **int getStart()**
- **void setEnd(int end)**
- **void getEnd()**

Constructor Detail

Prototype	`public TaskHolidayInfo(String name)`	
Argument	*name*	이름 ※ 주의사항 : 이름에 ";" 문자가 포함될 수 없습니다.

Method Detail

- **getName**

Prototype	`public String getName()`

Definition	태스크 휴일 이름을 가져옵니다.

■ **setBaseDay**

Prototype	public void setbaseDay(String pattern)	
Definition	태스크 휴일로 지정할 기준일을 설정합니다.	
Argument	*pattern*	기준일의 패턴 ※ 참고사항 : 기준일 포맷 • yyyy-MM-dd : 항상 휴일로 설정 • yyyy-MM-01 : 매월 1일을 휴일로 설정 • yyyy-01-01 : 매년 1월 1일을 휴일로 설정 • 2007-01-01 : 2007년 1월 1일을 휴일로 설정

■ **getBaseDay**

Prototype	public String getBaseDay()
Definition	태스크 휴일로 지정한 기준일을 가져옵니다.

■ **setType**

Prototype	public void setType(String type)	
Definition	태스크 휴일 등록 시 양력, 음력 중 타입을 지정합니다.	
Argument	*type*	양력, 음력 중 타입 solar : 양력 lunar : 음력 ※ 참고사항 : solar, lunar 이외의 값을 설정할 경우에는 solar로 설정됩니다. (solar, lunar 이외의 타입이 존재할 경우 차후 추가 예정)

■ **getType**

Prototype	public String getType()
Definition	태스크 휴일 등록 시 지정한 타입(양력, 음력)을 가져옵니다.

■ **isLunar**

Prototype	public boolean isLunar()
Definition	음력여부를 가져옵니다.

■ **setStart**

Prototype	public void setStart(int start)

Definition	"baseday"를 기준으로 며칠 전부터 휴일로 지정할지를 설정합니다.
Argument	*start* 며칠 전부터 시작할지에 대한 시작 값

■ **getStart**

Prototype	public int getStart()
Definition	"baseday"를 기준으로 며칠 전부터 휴일로 지정했는지 값을 가져옵니다.

■ **setEnd**

Prototype	public void setEnd(int end)
Definition	"baseday"를 기준으로 며칠 후까지 휴일로 지정할지를 설정합니다.
Argument	*end* 며칠 후까지 할 것인지에 대한 값

■ **getEnd**

Prototype	public void getEnd()
Definition	"baseday"를 기준으로 며칠 후까지 휴일로 지정했는지 값을 가져옵니다.

Sample : SchedulerTaskHoliday.java

```
package sample;

import java.util.*;

import oz.util.OZMap;
import oz.framework.api.*;
import oz.scheduler.holiday.*;

public class SchedulerTaskHoliday {

   public static void main(String[] args)
   {

      String IP = "127.0.0.1"; //scheduler server ip
      int PORT = 9521; //scheduler server port
      Scheduler scheduler = null;

      try {
         scheduler = new Scheduler(IP, PORT);

         //TaskHolidayInfo 의 리스트를 가져온다.
```

```
            OZMap infoMap = scheduler.getTaskHolidayInfoList();

            Iterator set = infoMap.keys();
            while(set.hasNext()) {
                String key = (String)set.next();
                TaskHolidayInfo info = (TaskHolidayInfo)infoMap.get(key);
                System.out.println(info. toString());
            }

            //TaskHolidayInfo 를 추가한다.
            TaskHolidayInfo info = new TaskHolidayInfo("창립기념일");
            info.setBaseDay("yyyy-07-01");
            info.setType("solar");
            info.setStart(0);
            info.setEnd(0);
            scheduler.addTaskHolidayInfo(info);

            //TaskHolidayInfo 를 수정한다.
            info.setBaseDay("yyyy-07-02");
            scheduler.modifyTaskHolidayInfo("창립기념일",info);

            //TaskHolidayInfo 를 삭제한다.
            scheduler.deleteTaskHolidayInfo("창립기념일");

            //최종 TaskHoliday 에 대한 내용을 스케줄러에 저장한다.
            //(설정변경후 반드시 저장이 되어야 반영이 된다.)
            scheduler.saveTaskHoliday();

        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

}
```

※ 참고사항 : 태스크 휴일 정보와 태스크 휴일 그룹 정보를 동시에 설정할 경우에는 Scheduler Info Properties를 {"task_holiday", "태스크 휴일 이름;태스크 휴일 그룹 이름"} 형태로 추가하여야 합니다. (ex : {"task_holiday", "설날;Group1"})

Class TaskHolidayGroupInfo

Constructor Summary

- **TaskHolidayGroupInfo(String name)**

Method Summary

- **String getName()**
- **void addReference(String name)**
- **void addReference(String[] ref)**
- **String[] getReferences()**
- **void removeReference(String name)**
- **void removeReference(String[] name)**
- **void removeAllRef()**

Constructor Detail

Prototype	`public TaskHolidayGroupInfo(String name)`	
Argument	*name*	이름 ※ 주의사항 : 이름에 ";" 문자가 포함될 수 없습니다.

Method Detail

- **getName**

Prototype	`public String getName()`
Definition	태스크 휴일 그룹 이름을 가져옵니다.

- **addReference**

Prototype	public void addReference(String name)
Definition	태스크 휴일 그룹이 참조하는 태스크 휴일 이름을 등록합니다.
Argument	*name* 태스크 휴일 이름 (동일한 이름이 있을 경우 해당 값 무시함)

■ **addReference**

Prototype	public void addReference(String[] names)
Definition	태스크 휴일 그룹에서 참조하는 태스크 휴일 이름을 삭제합니다.
Argument	*names* 태스크 휴일 그룹에서 삭제할 태스크 휴일 이름

■ **getReferences**

Prototype	public String[] getReferences()
Definition	태스크 휴일을 참조할 Reference 이름의 배열을 가져옵니다.

■ **removeReference**

Prototype	public void removeReference(String name)
Definition	태스크 휴일 그룹에서 참조하는 태스크 휴일 이름을 삭제합니다.
Argument	*name* 태스크 휴일 그룹에서 삭제할 태스크 휴일 이름

■ **removeReference**

Prototype	public void removeReference(String[] name)
Definition	태스크 휴일 그룹에서 참조하는 태스크 휴일 이름을 삭제합니다.
Argument	*name* 태스크 휴일 그룹에서 삭제할 태스크 휴일 이름의 배열

■ **removeAllRef**

Prototype	public void removeAllRef()
Definition	태스크 휴일 그룹에서 참조하는 태스크 휴일 이름을 모두 삭제합니다.

Sample : SchedulerTaskHolidayGroup.java

```
package sample;

import java.util.*;
```

```
import oz.util.OZMap;
import oz.framework.api.*;
import oz.scheduler.holiday.*;


public class SchedulerTaskHolidayGroup {

   public static void main(String[] args)
   {

      String IP = "127.0.0.1"; //scheduler server ip
      int PORT = 9521; //scheduler server port
      Scheduler scheduler = null;

      try {
         scheduler = new Scheduler(IP, PORT);

         //TaskHolidayInfoGroup 의 리스트를 가져온다.
         OZMap gInfoMap = scheduler.getTaskHolidayGroupInfoList();
         Iterator set1 = gInfoMap.keys();
         while(set1.hasNext()) {
            String key = (String)set1.next();
            TaskHolidayGroupInfo ginfo =
(TaskHolidayGroupInfo)gInfoMap.get(key);
            System.out.println(ginfo.toString());
         }

         //TaskHolidayGroupInfo 을 추가한다.
         TaskHolidayGroupInfo gInfo = new TaskHolidayGroupInfo("group1");
         gInfo.addReference("신정");
         gInfo.addReference("설날");
         gInfo.addReference("31 절");
         scheduler.addTaskHolidayGroupInfo(gInfo);

         //TaskHolidayGroupInfo 을 수정한다.
         gInfo.addReference("근로자의날");
         scheduler.modifyTaskHolidayGroupInfo("group1", gInfo);
         //TaskHolidayGroupInfo 을 삭제한다.
         scheduler.deleteTaskHolidayGroupInfo("group1");

         //최종 TaskHoliday 에 대한 내용을 스케줄러에 저장한다.
         //(설정변경후 반드시 저장이 되어야 반영이 된다.)
         scheduler.saveTaskHoliday();

      }
      catch(Exception e)
      {
         e.printStackTrace();
```

```
        }
    }

}
```

※ 참고사항 : 태스크 휴일 정보와 태스크 휴일 그룹 정보를 동시에 설정할 경우에는 Scheduler Info Properties를 {"task_holiday", "태스크 휴일 이름;태스크 휴일 그룹 이름"} 형태로 추가하여야 합니다. (ex : {"task_holiday", "설날;Group1"})

Ⅲ. 오즈 리파지토리 API

- Interface Repository
- Class RepositoryEx
- 오즈 리파지토리 구현

Interface Repository

Interface Summary

[패키지명 : oz.framework.repositoryex]

■ 기본 리파지토리 인터페이스
- **public interface IOZRepository**

■ 아이템 관련 리파지토리 인터페이스
- **public interface IOZRepositoryCategory**
- **public interface IOZRepositoryItem**
- **public interface IOZRepositoryItemHistory**

■ 사용자 관련 리파지토리 인터페이스
- **public interface IOZRepositoryGroup**
- **public interface IOZRepositoryUser**
- **public interface IOZRepositoryMultiLoginUser**

■ **Exception** 관련 클래스
- **public class OZRepositoryException extends Exception**

[패키지명 : oz.framework.repositoryex.auth]

■ 권한 관련 인터페이스
- **public interface IOZRepositoryAuthGroupCategory**
- **public interface IOZRepositoryAuthGroupItem**
- **public interface IOZRepositoryAuthUserCategory**
- **public interface IOZRepositoryAuthUserItem**

[패키지명 : oz.framework.repositoryex.info]

■ 리파지토리 정보**(bean)** 관련 인터페이스
- **public interface IOZCategoryInfo**

- **public interface IOZItemInfo**
- **public interface IOZHistoryInfo**
- **public interface IOZGroupInfo**
- **public interface IOZUserInfo**
- **public interface IOZLoginInfo**

■ 리파지토리 정보**(bean)** 목록 관련 인터페이스

- **public interface IOZCategoryInfoList**
- **public interface IOZItemInfoList**
- **public interface IOZHistoryInfoList**
- **public interface IOZGroupInfoList**
- **public interface IOZUserInfoList**

Interface Method Summary

[IOZRepository]

- **void setProperty(SortProperties prop) throws OZRepositoryException**
- **int getRepositoryVersion()**
- **void openRepository() throws OZRepositoryException**
- **void closeRepository() throws OZRepositoryException**
- **Object checkRepositoryLogin(String userName, String password, IOZLoginInfo loginInfo, HttpServletRequest request, HttpServlet servlet) throws OZRepositoryException**
- **void updateSesssionState(Object sessionID, int type) throws OZRepositoryException**
- **boolean repositoryLogout(Object sessionID, String userName, IOZLoginInfo loginInfo, HttpServletRequest request, HttpServlet servlet) throws OZRepositoryException**
- **IOZRepositoryCategory getRepositoryCategory() throws OZRepositoryException**
- **IOZRepositoryItem getRepositoryItem() throws OZRepositoryException**
- **IOZRepositoryItemHistory getRepositoryItemHistory() throws OZRepositoryException**

- **IOZRepositoryGroup getRepositoryGroup() throws OZRepositoryException**
- **IOZRepositoryUser getRepositoryUser() throws OZRepositoryException**
- **IOZRepositoryMultiLoginUser getRepositoryMultiLoginUser() throws OZRepositoryException**
- **IOZRepositoryAuthGroupCategory getRepositoryAuthGroupCategory() throws OZRepositoryException**
- **IOZRepositoryAuthGroupItem getRepositoryAuthGroupItem() throws OZRepositoryException**
- **IOZRepositoryAuthUserCategory getRepositoryAuthUserCategory() throws OZRepositoryException**
- **IOZRepositoryAuthUserItem getRepositoryAuthUserItem() throws OZRepositoryException**

[IOZRepositoryCategory]

- **int getAccessTypeCategory()**
- **IOZRepository getOZRepository() throws OZRepositoryException**
- **String[] createItemInCategory(Object sessionID, String[] itemNames, String[] descs, String[] categoryIDs, InputStream[] item_ins, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException**
- **String[] createCategory(Object sessionID, String[] categoryNames, String[] pCategoryIDs, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException**
- **String modifyCategoryName(Object sessionID, String categoryID, String new_CategoryName, String comment) throws OZRepositoryException**
- **boolean[] deleteCategory(Object sessionID, String[] categoryIDs, boolean[] isDestroys, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException**
- **boolean[] unDeleteCategory(Object sessionID, String[] categoryIDs, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException**
- **boolean hasTheItemInCategory(Object sessionID, String itemID) throws OZRepositoryException**
- **int getItemCountInCategory(Object sessionID, String categoryID) throws**

OZRepositoryException

- IOZItemInfoList getItemListInCategory(Object sessionID, String categoryID) throws OZRepositoryException
- IOZCategoryInfoList getCategoryListInCategory(Object sessionID, String categoryID) throws OZRepositoryException
- String getCategoryIdOfItem(Object sessionID, String itemID) throws OZRepositoryException
- IOZCategoryInfo getCategoryInfo(Object sessionID, String categoryID) throws OZRepositoryException
- IOZItemInfoList getDeletedItemListInCategory(Object sessionID, String categoryID) throws OZRepositoryException
- boolean transferItem(Object sessionID, String[] itemIDs, String target_CategoryID) throws OZRepositoryException
- boolean transferCategory(Object sessionID, String categoryID, String target_CategoryID) throws OZRepositoryException

[IOZRepositoryItem]

- int getAccessTypeItem()
- IOZRepository getOZRepository() throws OZRepositoryException
- String[] createItem(Object sessionID, String[] itemNames, String[] descs, InputStream[] item_ins, String comment, int[] errorCode, String[] errorMsg) throws
- String modifyItemName(Object sessionID, String itemID, String new_itemName, String comment) throws OZRepositoryException
- boolean[] deleteItem(Object sessionID, String[] itemIDs, boolean[] isDestroys, String comment, int[] errorCode, String[] errorMsg) throws
- boolean[] unDeleteItem(Object sessionID, String[] itemIDs, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException
- boolean modifyItemDesc(Object sessionID, String itemID, String desc) throws OZRepositoryException
- IOZItemInfo getItemInfo(Object sessionID, String itemID) throws OZRepositoryException
- boolean hasTheItem(Object sessionID, String itemID) throws

OZRepositoryException

- InputStream[] getItems(Object sessionID, String itemIDs[], int[] errorCode, String[] errorMsg) throws OZRepositoryException
- InputStream[] getItems(Object sessionID, String itemIDs[], long[] modifiedTimes, int[] errorCode, String[] errorMsg) throws OZRepositoryException
- InputStream[] checkOutItem(Object sessionID, String itemIDs[], String[] localCheckOutFolders, long[] localFileTimes, String[] checkOutCmts, int[] errorCode, String[] errorMsg) throws
- boolean[] checkInItem(Object sessionID, String[] itemIDs, InputStream[] item_ins, String comment, boolean[] keepCheckOut, int[] errorCode, String[] errorMsg) throws
- InputStream[] undoCheckOutItem(Object sessionID, String[] itemIDs, boolean[] isReplaces, int[] errorCode, String[] errorMsg) throws OZRepositoryException
- boolean[] isCheckOutUser(Object sessionID, String[] itemIDs) throws OZRepositoryException

[IOZRepositoryItemHistory]

- int getAccessTypeItemHistory()
- IOZRepository getOZRepository() throws OZRepositoryException
- boolean rollBackItem(Object sessionID, String itemID, int itemVersion, String comment) throws OZRepositoryException
- InputStream getItemByVersion(Object sessionID, String itemID, int version) throws OZRepositoryException
- IOZHistoryInfoList getHistoryItemList(Object sessionID, String itemID) throws OZRepositoryException
- IOZHistoryInfoList getDeleteHistoryItemInfo(Object sessionID, String itemID) throws OZRepositoryException
- boolean removeHistoryItem(Object sessionID, String itemID, int itemVersion) throws OZRepositoryException

[IOZRepositoryGroup]

- int getAccessTypeGroup()

- **IOZRepository getOZRepository() throws OZRepositoryException**
- **String createGroup(Object sessionID, String gName, String pGroupID, String desc) throws OZRepositoryException**
- **String modifyGroupName(Object sessionID, String groupID, String gName) throws OZRepositoryException**
- **boolean modifyGroupDesc(Object sessionID, String groupID, String desc) throws OZRepositoryException**
- **boolean deleteGroup(Object sessionID, String groupID) throws OZRepositoryException**
- **String createUserInGroup(Object sessionID, String userName, String password, String groupID, String desc) throws OZRepositoryException**
- **IOZUserInfoList getUserInfoListInGroup(Object sessionID, String groupID) throws OZRepositoryException**
- **IOZGroupInfo getGroupInfo(Object sessionID, String groupID) throws OZRepositoryException**
- **IOZGroupInfoList getSubGroupInfoList(Object sessionID, String groupID) throws OZRepositoryException**
- **IOZGroupInfo getParentGroupInfo(Object sessionID, String groupID) throws OZRepositoryException**
- **IOZGroupInfo getGroupIdOfUser(Object sessionID, String userID) throws OZRepositoryException**
- **boolean addGroupAdmin(Object sessionID, String userID, String groupID) throws OZRepositoryException**
- **boolean removeGroupAdmin(Object sessionID, String userID, String groupID) throws OZRepositoryException**
- **boolean isUserGroupAdmin(Object sessionID, String userID, String groupID) throws OZRepositoryException**
- **IOZUserInfoList getUserAdminListInGroup(Object sessionID, String groupID) throws OZRepositoryException**
- **boolean transferUser(Object sessionID, String userID, String target_GroupID) throws OZRepositoryException**
- **boolean transferGroup(Object sessionID, String groupID, String target_GroupID) throws OZRepositoryException**

[IOZRepositoryUser]

- **int getAccessTypeUser()**
- **IOZRepository getOZRepository() throws OZRepositoryException**
- **String createUser(Object sessionID, String userName, String password, String desc) throws OZRepositoryException**
- **String modifyUserName(Object sessionID, String userID, String userName) throws OZRepositoryException**
- **boolean modifyUserPassword(Object sessionID, String userID, String old_password, String new_password) throws OZRepositoryException**
- **boolean modifyUserDesc(Object sessionID, String userID, String desc) throws OZRepositoryException**
- **boolean deleteUser(Object sessionID, String userID) throws OZRepositoryException**
- **IOZUserInfo getUserInfo(Object sessionID, String userID) throws OZRepositoryException**
- **boolean checkUserPassword(Object sessionID, String userID, String password) throws OZRepositoryException**
- **IOZUserInfoList getUserInfoList(Object sessionID) throws OZRepositoryException**
- **boolean isCheckAdmin(Object sessionID, String userID) throws OZRepositoryException**

[IOZRepositoryMultiLoginUser]

- **int getAccessTypeUser()**
- **IOZRepository getOZRepository() throws OZRepositoryException**
- **String createUser(Object sessionID, String userName, String password, String desc) throws OZRepositoryException**
- **String modifyUserName(Object sessionID, String userID, String userName) throws OZRepositoryException**
- **boolean modifyUserPassword(Object sessionID, String userID, String old_password, String new_password) throws OZRepositoryException**
- **boolean modifyUserDesc(Object sessionID, String userID, String desc) throws OZRepositoryException**
- **boolean deleteUser(Object sessionID, String userID) throws**

OZRepositoryException

- IOZUserInfo getUserInfo(Object sessionID, String userID) throws OZRepositoryException
- boolean checkUserPassword(Object sessionID, String userID, String password) throws OZRepositoryException
- IOZUserInfoList getUserInfoList(Object sessionID) throws OZRepositoryException
- boolean isCheckAdmin(Object sessionID, String userID) throws OZRepositoryException

[IOZRepositoryAuthGroupCategory]

- int getAccessTypeAuthGroupCategory()
- IOZRepository getOZRepository() throws OZRepositoryException
- boolean modifyCategoryGroupAuth(Object sessionID, String categoryID, String groupID, int perm) throws OZRepositoryException
- int getGroupAuthToCategory(Object sessionID, String categoryID, String groupID) throws OZRepositoryException
- IOZGroupInfoList getGroupListAuthToCategory(Object sessionID, String categoryID, int perm) throws OZRepositoryException
- IOZItemInfoList getItemListAuthToGroupInCategory(Object sessionID, String groupID, String categoryID, int perm) throws OZRepositoryException
- IOZCategoryInfoList getCategoryListAuthToGroupInCategory(Object sessionID, String groupID, String categoryID, int perm) throws OZRepositoryException

[IOZRepositoryAuthGroupItem]

- int getAccessTypeAuthGroupItem()
- IOZRepository getOZRepository() throws OZRepositoryException
- boolean modifyItemGroupAuth(Object sessionID, String groupID, String itemID, int perm) throws OZRepositoryException
- int getGroupAuthToItem(Object sessionID, String groupID, String itemID) throws OZRepositoryException
- IOZGroupInfoList getGroupListAuthToItem(Object sessionID, String itemID,

int perm) throws OZRepositoryException

- IOZItemInfoList getItemListAuthToGroup(Object sessionID, String groupID, int perm) throws OZRepositoryException

[IOZRepositoryAuthUserCategory]

- int getAccessTypeAuthUserCategory()
- IOZRepository getOZRepository() throws OZRepositoryException
- boolean modifyCategoryUserAuth(Object sessionID, String userID, String categoryID, int perm) throws OZRepositoryException
- int getUserAuthToCategory(Object sessionID, String userID, String categoryID) throws OZRepositoryException
- IOZUserInfoList getUserListAuthToCategory(Object sessionID, String categoryID, int perm) throws OZRepositoryException
- IOZItemInfoList getItemListAuthToUserInCategory(Object sessionID, String userID, String categoryID, int perm) throws OZRepositoryException
- IOZCategoryInfoList getCategoryListAuthToUserInCategory(Object sessionID, String userID, String categoryID, int perm) throws OZRepositoryException

[IOZRepositoryAuthUserItem]

- int getAccessTypeAuthUserItem()
- IOZRepository getOZRepository() throws OZRepositoryException
- boolean modifyItemUserAuth(Object sessionID, String userID, String itemID, int perm) throws OZRepositoryException
- int getUserAuthToItem(Object sessionID, String userID, String itemID) throws OZRepositoryException
- IOZUserInfoList getUserListAuthToItem(Object sessionID, String itemID, int perm) throws OZRepositoryException
- IOZItemInfoList getItemListAuthToUser(Object sessionID, String userID, int perm) throws OZRepositoryException

Interface Method Detail

[IOZRepository Interface Method]

■ setProperty

Prototype	`public void setProperty(SortProperties prop) throws OZRepositoryException`
Definition	리파지토리 설정을 변경합니다. "repository.properties"에 설정되어 있는 값을 변경할 수 있으며, 설정할 수 있는 Key와 값은 아래 표를 참조하시기 바랍니다.
Argument	*prop* 리파지토리 설정 속성

■ getRepositoryVersion

Prototype	`public int getRepositoryVersion()`
Definition	리파지토리 버전을 정의한 후 정의한 버전을 가져옵니다.

■ openRepository

Prototype	`public void openRepository() throws OZRepositoryException`
Definition	서버 구동 시 리파지토리를 오픈합니다.

■ closeRepository

Prototype	`public void closeRepository() throws OZRepositoryException`
Definition	서버 종료 시 리파지토리를 종료합니다.

■ checkRepositoryLogin

Prototype	`public Object checkRepositoryLogin(String userName, String password, IOZLoginInfo loginInfo, HttpServletRequest request, HttpServlet servlet) throws OZRepositoryException`	
Definition	클라이언트를 호출할 때 해당 메소드를 호출하여 로그인한 세션 정보를 가져옵니다.	
Argument	*userName*	접속한 사용자 이름
	password	접속한 사용자의 패스워드
	isCallServer	서버에서 호출했는지 여부
	loginInfo	로그인 정보 ※ 참고사항 : 자세한 로그인 정보는 IOZLoginInfo 인터페이스를 참조하시기 바랍니다.
	request	HTTP Servelt Request
	servlet	HTTP Servlet

updateSesssionState

Prototype	`public void updateSesssionState(Object sessionID, int type) throws OZRepositoryException`	
Definition	클라이언트를 호출할 때 해당 메소드를 호출하여 로그인한 세션 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	type	서버 상태 • 0 : DEFAULT • 1 : BARE_SERVER • 2 : FROM_SERVER

repositoryLogout

Prototype	`public boolean repositoryLogout(Object sessionID, String userName, IOZLoginInfo loginInfo, HttpServletRequest request, HttpServlet servlet) throws OZRepositoryException`	
Definition	지정한 사용자 ID에 해당하는 사용자를 로그 아웃 즉, 접속 해제시키며 로그아웃 성공 여부를 가져옵니다.	
Argument	*sessionID*	세션 ID
	userName	사용자 이름
	loginInfo	로그인 정보 (자세한 정보는 해당 인터페이스 참조)
	request	HTTP Servelt Request
	servlet	HTTP Servlet

getRepositoryCategory

Prototype	`public IOZRepositoryCategory getRepositoryCategory() throws OZRepositoryException`
Definition	카테고리 인터페이스를 구현한 경우 구현된 클래스를 가져옵니다.

getRepositoryItem

Prototype	`public IOZRepositoryItem getRepositoryItem() throws OZRepositoryException`
Definition	아이템 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

getRepositoryItemHistory

Prototype	`public IOZRepositoryItemHistory getRepositoryItemHistory() throws OZRepositoryException`
Definition	아이템 히스토리 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

■ **getRepositoryGroup**

Prototype	public IOZRepositoryGroup getRepositoryGroup() throws OZRepositoryException
Definition	그룹 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

■ **getRepositoryUser**

Prototype	public IOZRepositoryUser getRepositoryUser() throws OZRepositoryException
Definition	유저 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

■ **getRepositoryMultiLoginUser**

Prototype	public IOZRepositoryMultiLoginUser getRepositoryMultiLoginUser() throws OZRepositoryException
Definition	멀티 로그인 유저 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

■ **getRepositoryAuthGroupCategory**

Prototype	public IOZRepositoryAuthGroupCategory getRepositoryAuthGroupCategory() throws OZRepositoryException
Definition	카테고리에 대한 그룹 권한 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

■ **getRepositoryAuthGroupItem**

Prototype	public IOZRepositoryAuthGroupItem getRepositoryAuthGroupItem() throws OZRepositoryException
Definition	아이템에 대한 그룹 권한 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

■ **getRepositoryAuthUserCategory**

Prototype	public IOZRepositoryAuthUserCategory getRepositoryAuthUserCategory() throws OZRepositoryException
Definition	카테고리에 대한 사용자 권한 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

■ **getRepositoryAuthUserItem**

Prototype	public IOZRepositoryAuthUserItem getRepositoryAuthUserItem() throws OZRepositoryException

Definition	아이템에 대한 사용자 권한 인터페이스를 구현한 경우 해당 구현된 클래스를 가져옵니다.

- Key

  setRepositoryConfig()와 getRepositoryConfig()에서 사용되는 key는 다음 표와 같습니다.

Key	Value	설명
REPOSITORY_TYPE	"RDB" "BUILTIN"	리파지토리 타입 ex) prop.setProperty("REPOSITORY_TYPE","RDB");
REPOSITORY_FILE_PATH	저장 경로	리파지토리 파일 경로 ex) prop.setProperty("REPOSITORY_FILE_PATH","c:/temp_repository");
REPOSITORY_ITEM_NUMBER_PER_DIRECTORY	아이템 파일수	디렉토리내 저장 가능한 아이템 파일 개수 (기본값 : "500") ex) prop.setProperty("REPOSITORY_ITEM_NUMBER_PER_DIRECTORY","100");
REPOSITORY_HISTORY_ITEM_VALID_DAYS	일 수	아이템 유효 기간 ex) prop.setProperty("REPOSITORY_HISTORY_ITEM_VALID_DAYS","20");
REPOSITORY_ADD_COMPRESSED_ITEM	"true" "false"	아이템 압축 여부 ex) prop.setProperty("REPOSITORY_ADD_COMPRESSED_ITEM","false");

[IOZRepositoryCategory Interface Method]

■ getAccessTypeCategory

Prototype	`public int getAccessTypeCategory()`

Definition	카테고리 인터페이스에 접근 가능한 메소드를 숫자 값으로 가져옵니다.
	int ACCESS_CATEGORY_NOT = 0x00000000
	int ACCESS_CREATEITEM_IN_CATEGORY = 0x00000001
	int ACCESS_CREATE_CATEGORY = 0x00000002
	int ACCESS_MODIFY_CATEGORY_NAME = 0x00000004
	int ACCESS_DELETE_CATEGORY = 0x00000008
	int ACCESS_UN_DELETE_CATEGORY = 0x00000010
	int ACCESS_HAS_THE_ITEM_IN_CATEGORY = 0x00000020
	int ACCESS_GET_ITEMCOUNT_IN_CATEGORY = 0x00000040
	int ACCESS_GET_ITEMLIST_IN_CATEGORY = 0x00000080
	int ACCESS_GET_CATEGORYID_OF_ITEM = 0x00000100
	int ACCESS_GET_CATEGORY_INFO = 0x00000200
	int ACCESS_GET_DELETED_ITEMLIST_IN_CATEGORY = 0x00000400
	int ACCESS_GET_SEARCH_ITEMLIST_IN_CATEGORY = 0x00000800
	int ACCESS_GET_CATEGORYLIST_IN_CATEGORY = 0x00001000
	int ACCESS_TRANSFER_ITEM = 0x00002000;
	int ACCESS_TRANSFER_CATEGORY = 0x00004000;

■ **getOZRepository**

Prototype	`public IOZRepository getOZRepository() throws OZRepositoryException`
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ **createItemInCategory**

Prototype	`public String[] createItemInCategory(Object sessionID, String[] itemNames, String[] descs, String[] categoryIDs, InputStream[] item_ins, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException`	
Definition	새로운 아이템을 생성하고 생성된 아이템의 ID를 반환합니다.	
Argument	*sessionID*	세션 ID
	itemName	새로 생성할 아이템의 ID
	descs	새로 생성할 아이템의 설명 내용
	categoryIDs	새로 생성할 아이템이 카테고리 ID
	item_ins	새로 생성할 아이템의 입력 스트림
	comment	주석문
	errorCode	에러 코드
	errorMsg	에러 메시지

■ **createCategory**

Prototype	public String[] createCategory(Object sessionID, String[] categoryNames, String[] pCategoryIDs, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException	
Definition	카테고리를 새로 생성하고 생성된 카테고리의 ID를 반환합니다.	
Argument	*sessionID*	세션 ID
	categoryName	생성할 카테고리의 이름
	pCategoryIDs	생성할 카테고리의 상위 카테고리 ID
	comment	주석문
	errorCode	에러 코드
	errorMsg	에러 메시지

■ **modifyCategoryName**

Prototype	public String modifyCategoryName(Object sessionID, String categoryID, String new_CategoryName, String comment) throws OZRepositoryException	
Definition	지정한 카테고리 ID에 해당하는 카테고리의 이름을 변경합니다.	
Argument	*sessionID*	세션 ID
	categoryID	이름을 변경할 카테고리의 ID
	new_categoryName	변경할 카테고리 이름
	comment	주석문

■ **deleteCategory**

Prototype	public boolean[] deleteCategory(Object sessionID, String[] categoryIDs, boolean[] isDestroys, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException	
Definition	지정한 카테고리 ID에 해당하는 카테고리를 삭제합니다.	
Argument	*sessionID*	세션 ID
	categoryIDs	삭제할 카테고리의 ID
	isDestroys	카테고리를 영구 삭제할 지 여부
	comment	주석문
	errorCode	에러 코드
	errorMsg	에러 메시지

■ **unDeleteCategory**

Prototype	`public boolean[] unDeleteCategory(Object sessionID, String[] categoryIDs, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException`	
Definition	삭제된 카테고리를 복원하고 복원 성공 여부를 가져옵니다. ※ 참고사항 : 카테고리 삭제 시 "isDestroys=false"로 설정하였을 경우 삭제된 카테고리의 복원이 가능합니다.	
Argument	*sessionID*	세션 ID
	categoryIDs	복원할 카테고리의 ID
	comment	주석문
	errorCode	에러 코드
	errorMsg	에러 메시지

■ **hasTheItemInCategory**

Prototype	`public boolean hasTheItemInCategory(Object sessionID, String itemID) throws OZRepositoryException`	
Definition	해당 카테고리에 특정 아이템이 존재하는지 여부를 가져옵니다.	
Argument	*sessionID*	세션 ID
	itemID	아이템 ID

■ **getItemCountInCategory**

Prototype	`public int getItemCountInCategory(Object sessionID, String categoryID) throws OZRepositoryException`	
Definition	지정한 카테고리에 속해 있는 모든 아이템의 개수를 가져옵니다.	
Argument	*sessionID*	세션 ID
	categoryID	아이템의 개수를 얻을 카테고리의 ID

■ **getItemListInCategory**

Prototype	`public IOZItemInfoList getItemListInCategory(Object sessionID, String categoryID) throws OZRepositoryException`	
Definition	지정한 카테고리에 속해 있는 모든 아이템 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	categoryID	아이템 정보를 얻을 카테고리 ID

■ **getCategoryIdOfItem**

Prototype	public String getCategoryIdOfItem(Object sessionID, String itemID) throws OZRepositoryException	
Definition	지정한 아이템이 존재하는 카테고리의 ID를 반환합니다.	
Argument	*sessionID*	세션 ID
	itemID	카테고리의 ID를 얻을 아이템 ID

■ **getCategoryInfo**

Prototype	public IOZCategoryInfo getCategoryInfo(Object sessionID, String categoryID) throws OZRepositoryException	
Definition	지정한 카테고리의 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	categoryID	정보를 가져올 카테고리 ID

■ **getDeletedItemListInCategory**

Prototype	public IOZItemInfoList getDeletedItemListInCategory(Object sessionID, String categoryID) throws OZRepositoryException	
Definition	지정한 카테고리의 삭제된 아이템 정보를 가져옵니다. ※ 참고사항 : "isDestroys=false"로 설정하여 삭제한 아이템의 정보만 가져옵니다.	
Argument	*sessionID*	세션 ID
	categoryID	삭제된 아이템 정보를 가져올 카테고리 ID

■ **transferItem**

Prototype	public boolean transferItem(Object sessionID, String[] itemIDs, String target_CategoryID) throws OZRepositoryException	
Definition	지정한 아이템의 카테고리를 이동하고 카테고리 이동 성공 여부를 가져옵니다.	
Argument	*sessionID*	세션 ID
	itemIDs	카테고리를 이동할 아이템 ID
	target_CategoryID	이동할 카테고리 ID

■ **transferCategory**

Prototype	public boolean transferCategory(Object sessionID, String categoryID, String target_CategoryID) throws OZRepositoryException	
Definition	지정한 카테고리를 다른 카테고리로 이동하고 카테고리 이동 성공 여부를 가져	

	옵니다.	
	sessionID	세션 ID
Argument	*itemIDs*	카테고리를 이동할 카테고리 ID
	target_CategoryID	이동할 카테고리 ID

[IOZRepositoryItem Interface Method]

■ getAccessTypeItem

Prototype	`public int getAccessTypeCategory()`
Definition	아이템 인터페이스에 접근 가능한 메소드를 숫자 값으로 가져옵니다. int ACCESS_ITEM_NOT = 0x00000000 int ACCESS_CREATE_ITEM = 0x00000001 int ACCESS_MODIFY_ITEMNAME = 0x00000002 int ACCESS_DELETE_ITEM = 0x00000004 int ACCESS_UN_DELETE_ITEM = 0x00000008 int ACCESS_MODIFY_ITEM_DESC = 0x00000010 int ACCESS_HAS_THE_ITEM = 0x00000020 int ACCESS_GET_ITEM_INFO = 0x00000040 int ACCESS_GET_ITEMS_UNCONDITION = 0x00000080 int ACCESS_GET_ITEMS = 0x00000100 int ACCESS_CHECKOUT_ITEM = 0x00000200 int ACCESS_CHECKIN_ITEM = 0x00000400 int ACCESS_UNDO_CHECKOUT_ITEM = 0x00000800 int ACCESS_IS_CHECKOUT_USER = 0x00001000

■ getOZRepository

Prototype	`public IOZRepository getOZRepository() throws OZRepositoryException`
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ createItem

Prototype	`public String[] createItem(Object sessionID, String[] itemNames, String[] descs, InputStream[] item_ins, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException`	
Definition	아이템을 생성하고 생성한 아이템 ID를 반환합니다.	
Argument	*sessionID*	세션 ID
	itemNames	아이템 이름
	descs	아이템 설명

	item_ins	아이템 스트림
	comment	주석문
	errCodes	에러 코드
	errMsgs	에러 메시지

■ **modifyItemName**

Prototype	`public String modifyItemName(Object sessionID, String itemID, String new_itemName, String comment) throws OZRepositoryException`	
Definition	지정한 아이템 ID에 해당하는 아이템 이름을 변경하고 변경된 아이템 ID를 반환합니다.	
Argument	*sessionID*	세션 ID
	itemID	이름을 변경할 아이템 ID
	new_itemName	변경할 아이템 이름
	comment	주석문

■ **deleteItem**

Prototype	`public boolean[] deleteItem(Object sessionID, String[] itemIDs, boolean[] isDestroys, String comment, int[] errorCode, String[] errorMsg) throws`	
Definition	지정한 아이템을 리파지토리에서 삭제하고 아이템 삭제 성공 여부를 가져옵니다.	
Argument	*sessionID*	세션 ID
	itemIDs	삭제할 아이템의 ID
	isDestroys	아이템을 영구 삭제할 지 여부
	comment	주석문
	errCodes	에러 코드
	errMsgs	에러 메시지

■ **unDeleteItem**

Prototype	`public boolean[] unDeleteItem(Object sessionID, String[] itemIDs, String comment, int[] errorCode, String[] errorMsg) throws OZRepositoryException`	
Definition	삭제된 아이템을 복원하고 복원 성공 여부를 가져옵니다. ※ 참고사항 : 아이템 삭제 시 "isDestroys=false"로 설정하였을 경우 삭제된 아이템의 복원이 가능합니다.	
Argument	*sessionID*	세션 ID

itemIDs	복원할 아이템 ID
comment	주석문
errCodes	에러 코드
errMsgs	에러 메시지

■ modifyItemDesc

Prototype	`public boolean modifyItemDesc(Object sessionID, String itemID, String desc) throws OZRepositoryException`
Definition	지정한 아이템 ID에 해당하는 아이템 설명을 변경하고 변경 성공 여부를 가져옵니다.
Argument	*sessionID* 세션 ID
	itemID 변경할 아이템 ID
	desc 변경할 내용

■ getItemInfo

Prototype	`public IOZItemInfo getItemInfo(Object sessionID, String itemID) throws OZRepositoryException`
Definition	지정한 아이템 ID에 해당하는 아이템 정보를 가져옵니다.
Argument	*sessionID* 세션 ID
	itemID 아이템 정보를 가져올 아이템 ID

■ hasTheItem

Prototype	`public boolean hasTheItem(Object sessionID, String itemID) throws OZRepositoryException`
Definition	지정한 아이템의 존재 여부를 가져옵니다.
Argument	*sessionID* 세션 ID
	itemID 아이템 ID

■ getItems

Prototype	`public InputStream[] getItems(Object sessionID, String itemIDs[], int[] errorCode, String[] errorMsg) throws OZRepositoryException`
	
	`public InputStream[] getItems(Object sessionID, String itemIDs[], long[] modifiedTimes, int[] errorCode, String[] errorMsg) throws OZRepositoryException`

Definition	지정한 아이템을 가져온 후 아이템 스트림을 가져옵니다.	
Argument	*sessionID*	세션 ID
	itemIDs	가져올 아이템 ID
	modifiedTimes	클라이언트 시간
	errCodes	에러 코드
	errMsgs	에러 메시지

■ checkOutItem

Prototype	`public InputStream[] checkOutItem(Object sessionID, String itemIDs[], String[] localCheckOutFolders, long[] localFileTimes, String[] checkOutCmts, int[] errorCode, String[] errorMsg) throws OZRepositoryException`
Definition	지정한 아이템을 지정한 사용자 ID로 체크 아웃할 폴더에 체크 아웃을 하고 로컬 파일의 시간이 아이템 시간 보다 작을 경우 아이템 스트림을 가져옵니다.

Argument		
	sessionID	세션 ID
	itemIDs	체크 아웃할 아이템의 ID
	localCheckOut Folders	체크 아웃할 로컬 폴더 이름
	localFileTimes	체크 아웃할 아이템의 로컬 파일 시간
	checkOutCmts	주석
	errCodes	에러 코드
	errMsgs	에러 메시지

■ checkInItem

Prototype	`public boolean[] checkInItem(Object sessionID, String[] itemIDs, InputStream[] item_ins, String comment, boolean[] keepCheckOut, int[] errorCode, String[] errorMsg) throws OZRepositoryException`
Definition	지정한 아이템을 지정한 사용자 ID로 체크인하고 체크인 성공 여부를 가져옵니다.

Argument		
	sessionID	세션 ID
	itemIDs	체크인할 아이템 ID
	item Streams	체크인할 아이템 스트림
	comment	주석문
	keepCheckOut	체크 아웃 상태를 유지할지 여부
	errCodes	에러 코드
	errMsgs	에러 메시지

■ **undoCheckOutItem**

Prototype	public InputStream[] undoCheckOutItem(Object sessionID, String[] itemIDs, boolean[] isReplaces, int[] errorCode, String[] errorMsg) throws OZRepositoryException	
Definition	지정한 아이템을 체크 아웃 취소합니다.	
Argument	*sessionID*	세션 ID
	itemIDs	체크 아웃 취소할 아이템의 ID
	isReplaces	로컬 파일 변경 여부
	errCodes	에러 코드
	errMsgs	에러 메시지

■ **isCheckOutUser**

Prototype	public boolean[] isCheckOutUser(Object sessionID, String[] itemIDs) throws OZRepositoryException	
Definition	지정한 사용자가 지정한 아이템을 체크 아웃했는지 여부를 확인합니다	
Argument	*sessionID*	세션 ID
	itemIDs	아이템 ID

[IOZRepositoryItemHistory]

■ **getAccessTypeItemHistory**

Prototype	public int getAccessTypeItemHistory()
Definition	아이템 히스토리 인터페이스에 접근 가능한 메소드를 숫자 값으로 가져옵니다. int ACCESS_ITEM_HISTORY_NOT = 0x00000000 int ACCESS_ROLLBACK_ITEM = 0x00000001 int ACCESS_GET_ITEM_BY_VERSION = 0x00000002 int ACCESS_GET_HISTORY_ITEMLIST = 0x00000004 int ACCESS_GET_DELETE_HISTORY_ITEMLIST = 0x00000008 int ACCESS_REMOVE_HISTORY_ITEM = 0x00000010

■ **getOZRepository**

Prototype	public IOZRepository getOZRepository() throws OZRepositoryException
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ **rollBackItem**

Prototype	public boolean rollBackItem(Object sessionID, String itemID, int itemVersion, String comment) throws OZRepositoryException	
Definition	지정한 아이템을 지정한 버전으로 복원시킵니다.	
Argument	*sessionID*	세션 ID
	itemID	복원시킬 아이템의 ID
	itemVersion	복원시킬 버전
	comment	주석문

■ **getItemByVersion**

Prototype	public InputStream getItemByVersion(Object sessionID, String itemID, int version) throws OZRepositoryException	
Definition	지정한 아이템 ID에 해당하는 아이템 중 지정한 버전의 아이템을 가져옵니다.	
Argument	*sessionID*	세션 ID
	itemID	가져올 아이템의 ID
	version	가져올 아이템의 버전

■ **getHistoryItemList**

Prototype	public IOZHistoryInfoList getHistoryItemList(Object sessionID, String itemID) throws OZRepositoryException	
Definition	지정한 아이템의 히스토리 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	itemIDs	히스토리 정보를 가져올 아이템의 ID

■ **getDeleteHistoryItemInfo**

Prototype	public IOZHistoryInfoList getDeleteHistoryItemInfo(Object sessionID, String itemID) throws OZRepositoryException	
Definition	삭제된 히스토리 리스트를 가져옵니다. ※ 참고사항 : "isDestroys=false"로 설정하여 삭제한 히스토리의 리스트만 가져옵니다.	
Argument	*sessionID*	세션 ID
	itemIDs	히스토리 정보를 가져올 아이템의 ID

■ **removeHistoryItem**

Prototype	`public boolean removeHistoryItem(Object sessionID, String itemID, int itemVersion) throws OZRepositoryException`	
Definition	지정한 아이템에 대해 특정 버전의 히스토리를 삭제합니다.	
Argument	*sessionID*	세션 ID
	itemID	히스토리를 삭제할 아이템의 ID
	item version	히스토리를 삭제할 버전

[IOZRepositoryGroup]

■ getAccessTypeGroup

Prototype	`public int getAccessTypeGroup()`
Definition	그룹 인터페이스에 접근 가능한 메소드를 숫자 값으로 가져옵니다. int ACCESS_GROUP_NOT = 0x00000000 int ACCESS_CREATE_GROUP = 0x00000001 int ACCESS_MODIFY_GROUP_NAME = 0x00000002 int ACCESS_MODIFY_GROUP_DESC = 0x00000004 int ACCESS_DELETE_GROUP = 0x00000008 int ACCESS_CREATE_UESR_IN_GROUP = 0x00000010 ACCESS_TRANSFER_USER = 0x00000020 int ACCESS_GET_USERINFOLIST_IN_GROUP = 0x00000040 int ACCESS_GET_GROUPINFO = 0x00000080 int ACCESS_GET_SUB_GROUPINFO_LIST = 0x00000100 int ACCESS_GET_PARENT_GROUPINFO = 0x00000200 int ACCESS_GET_GROUPID_OF_USER = 0x00000400 int ACCESS_ADD_GROUP_ADMIN = 0x00000800 int ACCESS_REMOVE_GROUP_ADMIN = 0x00001000 int ACCESS_IS_USER_GROUP_ADMIN = 0x00002000 int ACCESS_GET_USER_ADMINLIST_IN_GROUP = 0x00004000 int ACCESS_TRANSFER_GROUP = 0x00008000

■ getOZRepository

Prototype	`public IOZRepository getOZRepository() throws OZRepositoryException`
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ createGroup

Prototype	`public String createGroup(Object sessionID, String gName, String pGroupID, String desc) throws OZRepositoryException`
Definition	새로운 그룹을 생성하고 생성된 그룹의 ID를 반환합니다.

Argument	*sessionID*	세션 ID
	gName	새로 생성할 그룹의 이름
	pGroupID	새로 생성할 그룹의 상위 그룹 ID
	desc	그룹에 대한 설명 내용

■ modifyGroupName

Prototype	`public String modifyGroupName(Object sessionID, String groupID, String gName) throws OZRepositoryException`
Definition	지정한 그룹 ID에 해당하는 그룹의 이름을 변경하고 변경된 그룹 ID를 반환합니다.

Argument	*sessionID*	세션 ID
	groupID	그룹의 이름을 변경할 그룹 ID
	gName	변경할 그룹의 이름

■ modifyGroupDesc

Prototype	`public boolean modifyGroupDesc(Object sessionID, String groupID, String desc) throws OZRepositoryException`
Definition	지정한 그룹 ID에 해당하는 그룹의 설명을 변경하고 변경 성공 여부를 반환합니다.

Argument	*sessionID*	세션 ID
	groupID	그룹의 설명을 변경할 그룹 ID
	desc	변경할 그룹의 설명

■ deleteGroup

Prototype	`public boolean deleteGroup(Object sessionID, String groupID) throws OZRepositoryException`
Definition	지정한 그룹 ID에 해당하는 그룹을 삭제하고 삭제 성공 여부를 반환합니다.

Argument	*sessionID*	세션 ID
	groupID	삭제할 그룹 ID

■ createUserInGroup

Prototype	`public String createUserInGroup(Object sessionID, String userName, String password, String groupID, String desc) throws OZRepositoryException`
Definition	지정한 그룹에 새로운 사용자 ID를 생성하고 생성된 사용자 ID를 반환합니다.

Argument	*sessionID*	세션 ID
	userName	사용자 이름
	password	패스워드
	groupID	그룹ID
	desc	사용자에 대한 설명 내용

■ getUserInfoListInGroup

Prototype	public IOZUserInfoList getUserInfoListInGroup(Object sessionID, String groupID) throws OZRepositoryException	
Definition	지정한 그룹 ID에 등록되어 있는 모든 사용자의 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	groupID	사용자의 정보를 가져올 그룹 ID

■ getGroupInfo

Prototype	public IOZGroupInfo getGroupInfo(Object sessionID, String groupID) throws OZRepositoryException	
Definition	지정한 그룹 ID에 해당하는 그룹의 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	groupID	그룹 ID

■ getSubGroupInfoList

Prototype	public IOZGroupInfoList getSubGroupInfoList(Object sessionID, String groupID) throws OZRepositoryException	
Definition	지정한 그룹의 하위 그룹 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	groupID	하위 그룹의 정보를 가져올 그룹 ID

■ getParentGroupInfo

Prototype	public IOZGroupInfo getParentGroupInfo(Object sessionID, String groupID) throws OZRepositoryException	
Definition	지정한 그룹의 상위 그룹의 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	groupID	상위 그룹의 정보를 가져올 그룹 ID

■ **getGroupIdOfUser**

Prototype	public IOZGroupInfo getGroupIdOfUser(Object sessionID, String userID) throws OZRepositoryException	
Definition	지정한 사용자가 속해 있는 그룹의 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	userID	그룹 정보를 가져올 사용자 ID

■ **addGroupAdmin**

Prototype	public boolean addGroupAdmin(Object sessionID, String userID, String groupID) throws OZRepositoryException	
Definition	지정한 그룹의 그룹 관리자를 추가하고 추가 성공 여부를 반환합니다.	
Argument	*sessionID*	세션 ID
	userID	그룹 관리자로 추가할 사용자 ID
	groupID	그룹 관리자를 추가할 그룹 ID

■ **removeGroupAdmin**

Prototype	public boolean removeGroupAdmin(Object sessionID, String userID, String groupID) throws OZRepositoryException	
Definition	지정한 그룹의 그룹 관리자를 해제하고 해제 성공 여부를 반환합니다.	
Argument	*sessionID*	세션 ID
	userID	그룹 관리자 권한을 해제할 사용자 ID
	groupID	그룹 관리자를 해제할 그룹 ID

■ **isUserGroupAdmin**

Prototype	public boolean isUserGroupAdmin(Object sessionID, String userID, String groupID) throws OZRepositoryException	
Definition	지정한 사용자 ID에 해당하는 사용자가 해당 그룹의 관리자인지 여부를 확인합니다.	
Argument	*sessionID*	세션 ID
	userID	관리자인지 체크할 사용자 ID
	groupID	그룹 ID

■ **getUserAdminListInGroup**

Prototype	public IOZUserInfoList getUserAdminListInGroup(Object sessionID, String groupID) throws OZRepositoryException	
Definition	지정한 그룹의 그룹 관리자 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	groupID	그룹 관리자 정보를 가져올 그룹 ID

■ **transferUser**

Prototype	public boolean transferUser(Object sessionID, String userID, String target_GroupID) throws OZRepositoryException	
Definition	지정한 사용자의 그룹을 이동하고 그룹 이동 성공 여부를 가져옵니다.	
Argument	*sessionID*	세션 ID
	userID	그룹을 이동할 사용자 ID
	target_groupID	이동할 그룹 ID

■ **transferGroup**

Prototype	public boolean transferGroup(Object sessionID, String groupID, String target_GroupID) throws OZRepositoryException	
Definition	지정한 그룹을 다른 그룹으로 이동하고 그룹 이동 성공 여부를 가져옵니다.	
Argument	*sessionID*	세션 ID
	userID	그룹을 이동할 그룹 ID
	target_groupID	이동할 그룹 ID

[IOZRepositoryUser]

■ **getAccessTypeUser**

Prototype	public int getAccessTypeUSer()

Definition	사용자 인터페이스에 접근 가능한 메소드를 숫자 값으로 가져옵니다. int ACCESS_USER_NOT = 0x00000000 int ACCESS_CREATE_USER = 0x00000001 int ACCESS_MODIFY_USER_NAME = 0x00000002 int ACCESS_MODIFY_USER_PASSWORD = 0x00000004 int ACCESS_MODIFY_USER_DESC = 0x00000008 int ACCESS_DELETE_USER = 0x00000010 int ACCESS_GET_USERINFO = 0x00000020 int ACCESS_CHECK_USER_PASSWORD = 0x00000040 int ACCESS_GET_USERINFO_LIST = 0x00000080 int ACCESS_GET_IS_CHECK_ADMIN = 0x00000100

■ getOZRepository

Prototype	`public IOZRepository getOZRepository() throws OZRepositoryException`
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ createUser

Prototype	`public String createUser(Object sessionID, String userName, String password, String desc) throws OZRepositoryException`	
Definition	새로운 사용자를 생성하고 생성된 사용자의 ID를 반환합니다.	
Argument	*sessionID*	세션 ID
	userName	사용자 이름
	password	패스워드
	desc	사용자에 대한 설명 내용

■ modifyUserName

Prototype	`public String modifyUserName(Object sessionID, String userID, String userName) throws OZRepositoryException`	
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 변경하고 사용자 이름이 변경된 사용자 ID를 반환합니다.	
Argument	*sessionID*	세션 ID
	userID	사용자 ID
	userName	변경할 사용자 이름

■ modifyUserPassword

Prototype	`public boolean modifyUserPassword(Object sessionID, String userID, String old_password, String new_password) throws OZRepositoryException`	
Definition	지정한 사용자의 패스워드를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*sessionID*	세션 ID
	userID	패스워드를 변경할 사용자 ID
	old_password	변경 전 패스워드
	new_password	변경할 패스워드

■ **modifyUserDesc**

Prototype	`public boolean modifyUserDesc(Object sessionID, String userID, String desc) throws OZRepositoryException`	
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 설명을 변경하고 변경 성공 여부를 반환합니다.	
Argument	*sessionID*	세션 ID
	userID	사용자 설명 내용을 변경할 사용자 ID
	desc	변경할 사용자 설명

■ **deleteUser**

Prototype	`public boolean deleteUser(Object sessionID, String userID) throws OZRepositoryException`	
Definition	지정한 사용자 ID에 해당하는 사용자의 모든 정보를 삭제하고 삭제 성공 여부를 반환합니다.	
Argument	*sessionID*	세션 ID
	userID	삭제할 사용자의 ID

■ **getUserInfo**

Prototype	`public IOZUserInfo getUserInfo(Object sessionID, String userID) throws OZRepositoryException`	
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	userID	사용자 정보를 가져올 사용자 ID

■ **checkUserPassword**

Prototype	public boolean checkUserPassword(Object sessionID, String userID, String password) throws OZRepositoryException	
Definition	사용자의 패스워드가 맞는지 여부를 체크합니다.	
Argument	*sessionID*	세션 ID
	userID	패스워드를 체크할 사용자 ID
	password	체크할 패스워드

■ **getUserInfoList**

Prototype	public IOZUserInfoList getUserInfoList(Object sessionID) throws OZRepositoryException	
Definition	모든 사용자의 사용자 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID

■ **isCheckAdmin**

Prototype	public boolean isCheckAdmin(Object sessionID, String userID) throws OZRepositoryException	
Definition	지정한 사용자 ID에 해당하는 사용자가 리파지토리의 관리자인지 여부를 확인합니다.	
Argument	*sessionID*	세션 ID
	userID	관리자인지 체크할 사용자 ID

[IOZRepositoryMultiLoginUser]

■ **getAccessTypeMultiLoginUser**

Prototype	public int getAccessTypeMultiLoginUser()
Definition	멀티 로그인 인터페이스에 접근 가능한 메소드를 숫자 값으로 리턴한다. int ACCESS_MULTILOGINUSER_NOT = 0x00000000 int ACCESS_DISABLE_USER_LOGIN = 0x00000001 int ACCESS_ENABLE_USER_LOGIN = 0x00000002 int ACCESS_IS_LOGIN_ENABLED = 0x00000004

■ **getOZRepository**

Prototype	public IOZRepository getOZRepository() throws OZRepositoryException
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ **disableUserLogin**

Prototype	`public void disableUserLogin(Object sessionID) throws OZRepositoryException`
Definition	지정한 세션 ID의 사용자를 로그인하지 못하도록 설정합니다.
Argument	*sessionID* 로그인하지 못하도록 할 세션 ID

■ **enableUserLogin**

Prototype	`public void enableUserLogin(Object sessionID) throws OZRepositoryException`
Definition	지정한 세션 ID의 사용자를 로그인 가능하도록 설정합니다.
Argument	*sessionID* 로그인 가능하도록 할 세션 ID

■ **isLoginEnabled**

Prototype	`public boolean isLoginEnabled(Object sessionID) throws OZRepositoryException`
Definition	지정한 세션 ID의 사용자가 로그인이 가능한지 여부를 가져옵니다.
Argument	*sessionID* 로그인 가능 가능 여부를 체크할 세션 ID

[IOZRepositoryAuthGroupCategory]

■ **getAccessTypeAuthGroupCategory**

Prototype	`public int getAccessTypeAuthGroupCategory()`
Definition	카테고리에 대한 그룹 권한 인터페이스에 접근 가능한 메소드를 숫자 값으로 가져옵니다. int ACCESS_TYPE_AUTH_GROUP_CATEGORY_NOT = 0x00000000 int ACCESS_TYPE_MODIFY_CATEGORY_GROUPAUTH = 0x00000001 int ACCESS_TYPE_GET_GROUPAUTH_TO_CATEGORY = 0x00000002 int ACCESS_TYPE_GET_GROUPLIST_AUTH_TO_CATEGORY = 0x00000004 int ACCESS_TYPE_GET_ITEMLIST_AUTH_TO_GROUP_IN_CATEGORY = 0x00000008 int ACCESS_TYPE_GET_CATEGORYLIST_AUTH_TO_GROUP_IN_CATEGORY = 0x00010000

■ **getOZRepository**

Prototype	`public IOZRepository getOZRepository() throws OZRepositoryException`
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ modifyCategoryGroupAuth

Prototype	`public boolean modifyCategoryGroupAuth(Object sessionID, String categoryID, String groupID, int perm) throws OZRepositoryException`	
Definition	지정한 카테고리와 그룹의 권한 정보를 변경하고 변경 성공 여부를 가져옵니다.	
Argument	*sessionID*	세션 ID
	categoryID	권한 정보를 변경할 카테고리 ID
	groupID	권한 정보를 변경할 그룹 ID
	perm	권한 정보 • 1 : VIEW • 2 : READ • 4 : WRITE ※ 참고사항 : 권한 정보는 OR 조건으로 처리해야합니다.

■ getGroupAuthToCategory

Prototype	`public int getGroupAuthToCategory(Object sessionID, String categoryID, String groupID) throws OZRepositoryException`	
Definition	지정한 카테고리에 대한 지정한 그룹의 그룹 권한을 가져옵니다.	
Argument	*sessionID*	세션 ID
	categoryID	권한 정보를 가져올 카테고리 ID
	groupID	그룹 ID

■ getGroupListAuthToCategory

Prototype	`public IOZGroupInfoList getGroupListAuthToCategory(Object sessionID, String categoryID, int perm) throws OZRepositoryException`	
Definition	지정한 카테고리 ID에 대해 perm 이상의 권한을 가지고 있는 모든 사용자 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	categoryID	카테고리 ID
	perm	권한 정보

■ getItemListAuthToGroupInCategory

Prototype	public IOZItemInfoList getItemListAuthToGroupInCategory(Object sessionID, String groupID, String categoryID, int perm) throws OZRepositoryException	
Definition	지정한 카테고리의 아이템 중에 지정한 그룹의 권한이 perm 이상인 모든 아이템의 목록을 가져옵니다.	
Argument	*sessionID*	세션 ID
	groupID	그룹 ID
	categoryID	카테고리 ID
	perm	권한 정보

■ **getCategoryListAuthToGroupInCategory**

Prototype	public IOZCategoryInfoList getCategoryListAuthToGroupInCategory(Object sessionID, String groupID, String categoryID, int perm) throws OZRepositoryException	
Definition	지정한 카테고리의 하위 카테고리 중에 지정한 그룹의 권한이 perm 이상인 모든 카테고리의 목록을 가져옵니다.	
Argument	*sessionID*	세션 ID
	groupID	그룹 ID
	categoryID	카테고리 ID
	perm	권한 정보

[IOZRepositoryAuthGroupItem]

■ **getAccessTypeAuthGroupItem**

Prototype	public int getAccessTypeAuthGroupItem()
Definition	아이템에 대한 그룹 권한 인터페이스에 접근 가능한 메소드를 숫자 값으로 가져옵니다. int ACCESS_TYPE_AUTH_GROUP_ITEM_NOT = 0x00000000 int ACCESS_TYPE_MODIFY_ITEM_GROUPAUTH = 0x00000010 int ACCESS_TYPE_GET_GROUPAUTH_TO_ITEM = 0x00000020 int ACCESS_TYPE_GET_GROUPLIST_AUTH_TO_ITEM = 0x00000040 int ACCESS_TYPE_GET_ITEMLIST_AUTH_TO_GROUP = 0x00000080

■ **getOZRepository**

Prototype	public IOZRepository getOZRepository() throws OZRepositoryException

Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ **modifyItemGroupAuth**

Prototype	`public boolean modifyItemGroupAuth(Object sessionID, String groupID, String itemID, int perm) throws OZRepositoryException`	
Definition	지정한 아이템 ID에 대한 지정한 그룹 ID의 권한 정보를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*sessionID*	세션 ID
	groupID	그룹 ID
	itemID	아이템 ID
	perm	권한 정보

■ **getGroupAuthToItem**

Prototype	`public int getGroupAuthToItem(Object sessionID, String groupID, String itemID) throws OZRepositoryException`	
Definition	지정한 아이템 ID에 대한 지정한 그룹 ID의 권한 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	groupID	그룹 ID
	itemID	아이템 ID

■ **getGroupListAuthToItem**

Prototype	`public IOZGroupInfoList getGroupListAuthToItem(Object sessionID, String itemID, int perm) throws OZRepositoryException`	
Definition	지정한 아이템 ID에 대해 perm 이상의 권한이 부여된 모든 그룹 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	itemID	아이템 ID
	perm	권한 정보

■ **getItemListAuthToGroup**

Prototype	`public IOZItemInfoList getItemListAuthToGroup(Object sessionID, String groupID, int perm) throws OZRepositoryException`
Definition	지정한 그룹 ID에 해당하는 그룹에게 perm 이상의 권한이 부여된 모든 아이템의 정보를 가져옵니다.

Argument	*sessionID*	세션 ID
	groupID	그룹 ID
	perm	권한 정보

[IOZRepositoryAuthUserCategory]

■ getAccessTypeAuthUserCategory

Prototype	`public int getAccessTypeAuthUserCategory()`
Definition	카테고리에 대한 사용자 권한 인터페이스에 접근 가능한 메소드를 숫자 값으로 가져옵니다. int ACCESS_TYPE_AUTH_USER_CATEGORY_NOT = 0x00000000 int ACCESS_TYPE_MODIFY_CATEGORY_USERAUTH = 0x00000100 int ACCESS_TYPE_GET_USERAUTH_TO_CATEGORY = 0x00000200 int ACCESS_TYPE_GET_USERLIST_AUTH_TO_CATEGORY = 0x00000400 int ACCESS_TYPE_GET_ITEMLIST_AUTH_TO_USER_IN_CATEGORY= 0x00000800 int ACCESS_TYPE_GET_CATEGORYLIST_AUTH_TO_USER_IN_CATEGORY = 0x00020000

■ getOZRepository

Prototype	`public IOZRepository getOZRepository() throws OZRepositoryException`
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ modifyCategoryUserAuth

Prototype	`public boolean modifyCategoryUserAuth(Object sessionID, String userID, String categoryID, int perm) throws OZRepositoryException`	
Definition	지정한 카테고리 ID에 대한 지정한 사용자 ID의 권한 정보를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*sessionID*	세션 ID
	userID	사용자 ID
	categoryID	카테고리 ID
	perm	권한 정보

■ getUserAuthToCategory

Prototype	`public int getUserAuthToCategory(Object sessionID, String userID, String categoryID) throws OZRepositoryException`
Definition	지정한 카테고리 ID에 대한 지정한 사용자 ID의 권한 정보를 가져옵니다.

Argument	*sessionID*	세션 ID
	userID	사용자 ID
	categoryID	카테고리 ID

■ **getUserListAuthToCategory**

Prototype	`public IOZUserInfoList getUserListAuthToCategory(Object sessionID, String categoryID, int perm) throws OZRepositoryException`
Definition	지정한 카테고리 ID에 대해 perm 이상의 권한이 부여된 모든 사용자 정보를 가져옵니다.

Argument	*sessionID*	세션 ID
	categoryID	카테고리 ID
	perm	권한 정보

■ **getItemListAuthToUserInCategory**

Prototype	`public IOZItemInfoList getItemListAuthToUserInCategory(Object sessionID, String userID, String categoryID, int perm) throws OZRepositoryException`
Definition	지정한 카테고리의 아이템 중에 지정한 사용자의 권한이 perm 이상인 모든 아이템의 목록을 가져옵니다.

Argument	*sessionID*	세션 ID
	userID	사용자 ID
	categoryID	카테고리 ID
	perm	권한 정보

■ **getCategoryListAuthToUserInCategory**

Prototype	`public IOZCategoryInfoList getCategoryListAuthToUserInCategory(Object sessionID, String userID, String categoryID, int perm) throws OZRepositoryException`
Definition	지정한 카테고리의 하위 카테고리 중에 지정한 사용자의 권한이 perm 이상인 모든 카테고리의 목록을 가져옵니다.

Argument	*sessionID*	세션 ID
	userID	사용자 ID
	categoryID	카테고리 ID
	perm	권한 정보

[IOZRepositoryAuthUserItem]

■ getAccessTypeAuthUserItem

Prototype	`public int getAccessTypeAuthUserItem()`
Definition	아이템에 대한 사용자 권한 인터페이스에 접근 가능한 메소드를 숫자 값으로 가져옵니다. int ACCESS_TYPE_AUTH_USER_ITEM_NOT = 0x00000000 int ACCESS_TYPE_MODIFY_ITEM_USERAUTH = 0x00001000 int ACCESS_TYPE_GET_USERAUTH_TO_ITEM = 0x00002000 int ACCESS_TYPE_GET_USERLIST_AUTH_TO_ITEM = 0x00004000 int ACCESS_TYPE_GET_ITEMLIST_AUTH_TO_USER = 0x00008000

■ getOZRepository

Prototype	`public IOZRepository getOZRepository() throws OZRepositoryException`
Definition	OZRepostiroy 인터페이스를 구현한 클래스를 가져옵니다.

■ modifyItemUserAuth

Prototype	`public boolean modifyItemUserAuth(Object sessionID, String userID, String itemID, int perm) throws OZRepositoryException`	
Definition	지정한 아이템에 대한 사용자의 권한 정보를 변경하고 변경 성공 여부를 가져옵니다.	
Argument	*sessionID*	세션 ID
	userID	사용자 ID
	itemID	아이템 ID
	perm	권한 정보

■ getUserAuthToItem

Prototype	`public int getUserAuthToItem(Object sessionID, String userID, String itemID) throws OZRepositoryException`	
Definition	지정한 아이템에 대한 사용자의 권한 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	userID	사용자 ID
	itemID	권한 정보

■ getUserListAuthToItem

Prototype	public IOZUserInfoList getUserListAuthToItem(Object sessionID, String itemID, int perm) throws OZRepositoryException	
Definition	지정한 아이템에 대한 사용자의 권한이 perm 이상인 모든 아이템 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	iitemID	아이템 ID
	perm	권한 정보

■ **getItemListAuthToUser**

Prototype	public IOZItemInfoList getItemListAuthToUser(Object sessionID, String userID, int perm) throws OZRepositoryException	
Definition	지정한 사용자의 권한이 perm 이상인 모든 아이템 정보를 가져옵니다.	
Argument	*sessionID*	세션 ID
	userID	아이템 ID
	perm	권한 정보

[IOZCategoryInfo]

■ **getCategoryID**

Prototype	public String getCategoryID()
Definition	현재 카테고리의 ID를 가져옵니다.

■ **getCategoryName**

Prototype	public String getCategoryName()
Definition	현재 카테고리의 이름을 가져옵니다.

■ **getParentCategoryID**

Prototype	public String getParentCategoryID()
Definition	현재 카테고리의 상위 카테고리 ID를 가져옵니다.

■ **getParentCategoryName**

Prototype	public String getParentCategoryName()
Definition	현재 카테고리의 상위 카테고리 이름을 가져옵니다.

- **getComment**

Prototype	public String getComment()
Definition	현재 카테고리의 주석문을 가져옵니다.

- **getDesc**

Prototype	public String getDesc()
Definition	현재 카테고리의 설명 내용을 가져옵니다.

- **getUpdateTime**

Prototype	public long getUpdateTime()
Definition	현재 카테고리를 갱신한 시간 정보를 가져옵니다.

- **getCreateUserID**

Prototype	public String getCreateUserID()
Definition	현재 카테고리를 생성한 사용자 ID를 가져옵니다.

- **getCreateUserName**

Prototype	public String getCreateUserName()
Definition	현재 카테고리를 생성한 사용자 이름을 가져옵니다.

- **getPerm**

Prototype	public int getPerm()
Definition	현재 카테고리의 권한 정보를 가져옵니다.

[IOZItemInfo]

- **isCheckOut**

Prototype	public boolean isCheckOut()
Definition	해당 아이템의 체크 아웃 여부를 가져옵니다.

- **isDeletable**

Prototype	public boolean isDeletable()

Definition	해당 아이템의 삭제 여부를 가져옵니다. ※ 참고사항 : "isDestroys=false"로 설정하여 아이템을 삭제하였을 경우 해당 아이템의 삭제 여부를 가져옵니다.

■ getItemID

Prototype	`public String getItemID()`
Definition	해당 아이템의 아이템 ID를 가져옵니다.

■ getItemName

Prototype	`public String getItemName()`
Definition	해당 아이템의 아이템 이름을 가져옵니다.

■ getCategoryID

Prototype	`public String getCategoryID()`
Definition	해당 아이템이 속한 카테고리의 카테고리 ID를 가져옵니다.

■ getCategoryName

Prototype	`public String getCategoryName()`
Definition	해당 아이템이 속한 카테고리의 카테고리 이름을 가져옵니다.

■ getUpdateTime

Prototype	`public long getUpdateTime()`
Definition	해당 아이템을 갱신한 시간 정보를 가져옵니다.

■ getDesc

Prototype	`public String getDesc()`
Definition	해당 아이템의 설명 내용을 가져옵니다.

■ getComment

Prototype	`public String getComment()`
Definition	해당 아이템의 주석문을 가져옵니다.

■ getCreateUserID

Prototype	`public String getCreateUserID()`

Definition	해당 아이템을 생성한 사용자 ID를 가져옵니다.

- **getCreateUserName**

Prototype	`public String getCreateUserName()`
Definition	해당 아이템을 생성한 사용자 이름을 가져옵니다.

- **getCheckOutUser**

Prototype	`public String getCheckOutUser()`
Definition	해당 아이템을 체크 아웃한 사용자 이름을 가져옵니다.

- **getCheckOutLocalPath**

Prototype	`public String getCheckOutLocalPath()`
Definition	체크 아웃한 아이템의 폴더 이름을 가져옵니다.

- **getPerm**

Prototype	`public int getPerm()`
Definition	해당 아이템의 권한 정보를 가져옵니다.

[IOZHistoryInfo]

- **getItemID**

Prototype	`public String getItemID()`
Definition	아이템 ID를 가져옵니다.

- **getItemName**

Prototype	`public String getItemName()`
Definition	아이템 이름을 가져옵니다.

- **getHistoryItemVersion**

Prototype	`public int getHistoryItemVersion()`
Definition	아이템의 버전 개수를 가져옵니다.

- **getHistoryCheckInTime**

Prototype	public long getHistoryCheckInTime()
Definition	아이템의 버전 별 체크인 시간을 가져옵니다.

- **getHistoryCheckInUser**

Prototype	public String getHistoryCheckInUser()
Definition	아이템의 버전 별 체크인 사용자 ID를 가져옵니다.

- **getHistoryCheckInComment**

Prototype	public String getHistoryCheckInComment()
Definition	아이템의 버전 별 체크인 주석문을 가져옵니다.

[IOZGroupInfo]

- **isGroupAdmin**

Prototype	public boolean isGroupAdmin(String userID)
Definition	지정한 사용자 ID가 해당 그룹의 관리자인지 여부를 가져옵니다.
Argument	*userID* 사용자 ID

- **getGroupAdminList**

Prototype	public HashMap getGroupAdminList()
Definition	해당 그룹의 관리자 정보를 가져옵니다. ※ 참고사항 : 관리자 정보는 "key=*사용자ID*, value=*사용자 이름*"으로 구성된 HashMap으로 반환됩니다.

- **getGroupID**

Prototype	public String getGroupID()
Definition	해당 그룹의 그룹 ID를 가져옵니다.

- **getGroupName**

Prototype	public String getGroupName()
Definition	해당 그룹의 그룹 이름을 가져옵니다.

- **getParentGroupID**

Prototype	public String getParentGroupID()

Definition	해당 그룹의 상위 그룹 ID를 가져옵니다.

- **getParentGroupName**

Prototype	public String getParentGroupName()
Definition	해당 그룹의 상위 그룹 이름을 가져옵니다.

- **getDesc**

Prototype	public String getDesc()
Definition	해당 그룹의 설명 내용을 가져옵니다.

- **getUpdateTime**

Prototype	public long getUpdateTime()
Definition	해당 그룹을 갱신한 시간 정보를 가져옵니다.

[IOZUserInfo]

- **getUserID**

Prototype	public String getUserID()
Definition	해당 사용자의 사용자 ID를 가져옵니다.

- **getPassword**

Prototype	public String getPassword()
Definition	해당 사용자의 패스워드를 가져옵니다.

- **getUserName**

Prototype	public String getUserName()
Definition	해당 사용자의 사용자 이름을 가져옵니다.

- **getGroupID**

Prototype	public String getGroupID()
Definition	해당 사용자가 속한 그룹의 그룹 ID를 가져옵니다.

- **getGroupName**

Prototype	public String getGroupName()
Definition	해당 사용자가 속한 그룹의 그룹 이름을 가져옵니다.

- **getDesc**

Prototype	public String getDesc()
Definition	해당 사용자의 설명 내용을 가져옵니다.

- **getUpdateUserTime**

Prototype	public long getUpdateUserTime()
Definition	해당 사용자를 갱신한 시간 정보를 가져옵니다.

- **isLoggedIn**

Prototype	public boolean isLoggedIn()
Definition	해당 사용자의 로그인 여부를 가져옵니다.

- **getSessionID**

Prototype	public String getSessionID()
Definition	해당 사용자가 로그인 시 생성된 세션 ID를 가져옵니다.

- **isLoginEnabled**

Prototype	public boolean isLoginEnabled()
Definition	해당 사용자의 로그인 가능 여부를 가져옵니다.

- **getLoginIP**

Prototype	public String getLoginIP()
Definition	해당 사용자가 로그인한 PC의 IP 정보를 가져옵니다.

- **getLastLoginTime**

Prototype	public long getLastLoginTime()
Definition	해당 사용자가 마지막으로 로그인한 시간 정보를 가져옵니다.

- **getPerm**

Prototype	public int getPerm()

Definition	해당 사용자의 권한 정보를 가져옵니다.

[IOZLoginInfo]

- **getViewerType**

Prototype	`public int getViewerType()`
Definition	클라이언트에서 호출한 클라이언트의 종류를 가져옵니다.

- **getMACAddress**

Prototype	`public String getMACAddress()`
Definition	클라이언트로부터 MAC Address 정보를 가져옵니다.

- **getIPAddress**

Prototype	`public String getIPAddress()`
Definition	클라이언트로부터 IP Address 정보를 가져옵니다.

- **getHDDSerialNo**

Prototype	`public String getHDDSerialNo()`
Definition	클라이언트로부터 HDD Serial 번호를 가져옵니다.

- **getUserID**

Prototype	`public String getUserID()`
Definition	클라이언트가 로그인한 사용자 ID를 가져옵니다.

- **getUserName**

Prototype	`public String getUserName()`
Definition	클라이언트가 로그인한 사용자 이름을 가져옵니다.

- **getSessionID**

Prototype	`public String getSessionID()`
Definition	클라이언트가 로그인 시 생성된 세션 ID를 가져옵니다.

- **getClientIP**

Prototype	public String getClientIP()
Definition	클라이언트가 요청한 클라이언트 IP를 가져옵니다.

[IOZCategoryInfoList]

- **getCategoryInfo**

Prototype	public IOZCategoryInfo getCategoryInfo(int index)
Definition	해당 인덱스에 해당하는 카테고리 정보를 가져옵니다.
Argument	*index* 인덱스

- **getSize**

Prototype	public String getSize()
Definition	카테고리 목록의 개수를 가져옵니다.

- **getCategoryInfos**

Prototype	public IOZCategoryInfo[] getCategoryInfos()
Definition	카테고리 정보를 배열로 가져옵니다.

[IOZCategoryInfoList]

- **getItemInfo**

Prototype	public IOZItemInfo getItemInfo(int index)
Definition	해당 인덱스에 해당하는 아이템 정보를 가져옵니다.
Argument	*index* 인덱스

- **getSize**

Prototype	public String getSize()
Definition	아이템 목록의 개수를 가져옵니다.

- **getItemInfos**

Prototype	public IOZItemInfo[] getItemInfos()
Definition	아이템 정보를 배열로 가져옵니다.

[IOZHistoryInfoList]

■ **getHistoryInfo**

Prototype	`public IOZHistoryInfo getHistoryInfo(int index)`
Definition	해당 인덱스에 해당하는 히스토리 정보를 가져옵니다.
Argument	*index* 인덱스

■ **getSize**

Prototype	`public String getSize()`
Definition	히스토리 목록의 개수를 가져옵니다.

■ **getHistoryInfos**

Prototype	`public IOZHistoryInfo[] getHistoryInfos()`
Definition	히스토리 정보를 배열로 가져옵니다.

[IOZGroupInfoList]

■ **getGroupInfo**

Prototype	`public IOZGroupInfo getGroupInfo(int index)`
Definition	해당 인덱스에 해당하는 그룹 정보를 가져옵니다.
Argument	*index* 인덱스

■ **getSize**

Prototype	`public String getSize()`
Definition	그룹 목록의 개수를 가져옵니다.

■ **getGroupInfos**

Prototype	`public IOZGroupInfo[] getGroupInfos()`
Definition	그룹 정보를 배열로 가져옵니다.

[IOZUserInfoList]

■ **getGroupInfo**

Prototype	`public IOZUserInfoList getUserInfo(int index)`
Definition	해당 인덱스에 해당하는 사용자 정보를 가져옵니다.
Argument	*index* 인덱스

■ **getSize**

Prototype	`public String getSize()`
Definition	사용자 목록의 개수를 가져옵니다.

■ **getGroupInfos**

Prototype	`public IOZUserInfo[] getUserInfos()`
Definition	사용자 정보를 배열로 가져옵니다.

Class OZRepositoryException

[패키지명 : oz.framework.repositoryex]

■ **public class OZRepositoryException extends Exception**

■ **Method Summary**

- **public OZRepositoryException(String _msg)**
- **public OZRepositoryException(int code, String _msg)**
- **public OZErrorCode ErrorCode{get;}**
- **public String Message{get;}**

■ **Method Detail**

- OZRepositoryException

Prototype	`public OZRepositoryException(String _msg)` `public OZRepositoryException(int code,String _msg)`	
Definition	`OZRepositoryException` 생성자	
Argument	*code*	에러 코드
	_msg	에러 메시지

- ErrorCode

Prototype	`public OZErrorCode ErrorCode{get;}`
Definition	에러 코드

- Message

Prototype	`public String Message{get;}`
Definition	에러 메시지

Class RepositoryEX

Constructor Summary

- **RepositoryEx(String ip, int port, String id, String pw) throws OZCPException**
- **RepositoryEx(String ip, int port, String id, String pw, boolean useUSL) throws OZCPException**
- **RepositoryEx(String iurl, String id, String pw) throws OZCPException**
- **RepositoryEx(String iurl, String id, String pw, boolean useUSL) throws OZCPException**

Method Summary

- **void setRepositoryConfig(SortProperties prop) throws OZCPException**
- **SortProperties getRepositoryConfig()throws OZCPException**
- **void reloadRepository() throws OZCPException**
- **String[] createItem(String[] itemNames, String[] itemDescs, String[] categoryIDs, boolean[] isComp, InputStream[] itemStreams, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **String[] createItem(String[] itemNames, String[] itemDescs, String[] categoryIDs, boolean[] isComp, InputStream[] itemStreams, String comment, long[] createdTimes, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **String modifyItemName(String itemID, String itemName, String comment) throws OZCPException**
- **boolean modifyItemDatetime(String itemIDs, long datetime) throws OZCPException**
- **boolean[] deleteItem(String[] itemIDs, boolean[] isDestorys, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **boolean[] unDeleteItem(String[] itemIDs, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException**

- **boolean modifyItemDesc(String itemID, String iDesc) throws OZCPException**
- **IOZItemInfo getItemInfo(String itemID) throws OZCPException**
- **IOZItemInfoList getItemInfoListInCategory(String categoryID, boolean isRecursive) throws OZCPException**
- **boolean hasTheItem(String itemID) throws OZCPException**
- **InputStream[] getItemsUnCondition(String[] itemIDs, boolean[] isComp, boolean[] isObjStream, String[] categoryIDs, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **InputStream[] getItems(String[] itemIDs, long[] modifiedTimes, boolean[] isComp, boolean[] isObjStream, String[] categoryIDs, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **InputStream[] checkOutItem(String itemIDs[], String[] localCheckOutFolders, long[] localFileTimes, boolean[] isComp, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **public InputStream[] checkOutItem(String itemIDs[], String[] localCheckOutFolders, long[] localFileTimes, boolean[] isComp, String[] checkOutCmts, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **boolean[] checkInItem(String[] itemIDs, boolean[] isComp, InputStream[] iStreams, String comment, boolean[] keepCheckOut, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **boolean[] checkInItem(String[] itemIDs, boolean[] isComp, InputStream[] iStreams, String comment, boolean[] keepCheckOut, long[] datetimes, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **InputStream[] undoCheckOutItem(String[] itemIDs, boolean[] isReplaces, boolean[] isComp, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **boolean[] isCheckOutUser(String[] itemIDs) throws OZCPException**
- **boolean rollBackItem(String itemID, int itemVersion, String comment) throws OZCPException**
- **InputStream getItemByVersion(String itemID, int version, boolean isComp) throws OZCPException**
- **IOZHistoryInfoList getHistoryItemList(String itemID) throws OZCPException**
- **InputStream getHistoryItemByDatetime(String itemID, long datetime,**

boolean isComp, boolean isObjStream) throws OZCPException

- **IOZHistoryInfoList getDeleteHistoryItemInfo(String itemID) throws OZCPException**
- **boolean removeHistoryItem(String itemID, int itemVersion) throws OZCPException**
- **boolean modifyHistoryItemComment(String itemID, int itemVersion, String newComment) throws OZCPException**
- **boolean modifyHistoryItemDatetime(String itemID, int itemVersion, long datetime) throws OZCPException**
- **String[] createCategory(String[] categoryNames, String[] pCategoryIDs, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **boolean modifyCategoryDesc(String categoryID, String desc) throws OZCPException**
- **String modifyCategoryName(String categoryID, String new_CategoryName, String comment) throws OZCPException**
- **boolean[] deleteCategory(String[] categoryIDs, boolean[] isDestroys, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **boolean[] unDeleteCategory(String[] categoryIDs, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException**
- **int getItemCountInCategory(String categoryID) throws OZCPException**
- **IOZItemInfoList getItemListInCategory(String categoryID) throws OZCPException**
- **IOZCategoryInfoList getCategoryListInCategory(String categoryID) throws OZCPException**
- **String getCategoryIdOfItem(String itemID) throws OZCPException**
- **IOZCategoryInfo getCategoryInfo(String categoryID) throws OZCPException**
- **IOZItemInfoList getDeletedItemListInCategory(String categoryID) throws OZCPException**
- **boolean transferCategory(String categoryID, String target_CategoryID) throws OZCPException**
- **boolean transferItem(String[] itemIDs, String target_CategoryID) throws OZCPException**

- **String createUser(String userName, String password, String desc) throws OZCPException**
- **String modifyUserName(String userID, String userName) throws OZCPException**
- **boolean modifyUserPassword(String userID, String old_password, String new_password) throws OZCPException**
- **boolean modifyUserDesc(String userID, String desc) throws OZCPException**
- **boolean deleteUser(String userID) throws OZCPException**
- **IOZUserInfo getUserInfo(String userID) throws OZCPException**
- **boolean checkUserPassword(String userID, String password) throws OZCPException**
- **IOZUserInfoList getUserInfoList() throws OZCPException**
- **boolean isCheckAdmin(String userID) throws OZCPException**
- **void disableUserLogin(String userID) throws OZCPException**
- **void enableUserLogin(String userID) throws OZCPException**
- **boolean isLoginEnabled(String userID) throws OZCPException**
- **String createGroup(String gName, String pGroupID, String desc) throws OZCPException**
- **String modifyGroupName(String groupID, String gName) throws OZCPException**
- **boolean modifyGroupDesc(String groupID, String desc) throws OZCPException**
- **boolean deleteGroup(String groupID) throws OZCPException**
- **String createUserInGroup(String userName, String password, String groupID, String desc) throws OZCPException**
- **boolean transferGroup(String groupID, String target_GroupID) throws OZCPException**
- **boolean transferUser(String userID, String target_GroupID) throws OZCPException**
- **IOZUserInfoList getUserInfoListInGroup(String groupID) throws OZCPException**
- **IOZUserInfoList getUserInfoListInGroup(String groupID, boolean isRecursive) throws OZCPException**
- **IOZGroupInfo getGroupInfo(String groupID) throws OZCPException**
- **IOZGroupInfoList getGroupInfoListInGroup(String groupID, boolean**

isRecursive) throws OZCPException

- **IOZGroupInfoList getSubGroupInfoList(String groupID) throws OZCPException**
- **IOZGroupInfo getParentGroupInfo(String groupID) throws OZCPException**
- **String getGroupIdOfUser(String userID) throws OZCPException**
- **boolean addGroupAdmin(String userID, String groupID) throws OZCPException**
- **boolean removeGroupAdmin(String userID, String groupID) throws OZCPException**
- **boolean isUserGroupAdmin(String userID, String groupID) throws OZCPException**
- **IOZUserInfoList getUserAdminListInGroup(String groupID) throws OZCPException**
- **InputStream distributeRepository(String[] itemIDs) throws OZCPException**
- **InputStream distributeRepository(String categoryID, boolean isRecursive) throws OZCPException**
- **boolean uploadZipItem(String categoryID, InputStream stream) throws OZCPException**
- **boolean modifyCategoryGroupAuth(String categoryID, String groupID, int perm) throws OZCPException**
- **int getGroupAuthToCategory(String categoryID, String groupID) throws OZCPException**
- **IOZGroupInfoList getGroupListAuthToCategory(String categoryID, int perm) throws OZCPException**
- **IOZItemInfoList getItemListAuthToGroupInCategory(String groupID, String categoryID, int perm) throws OZCPException**
- **IOZCategoryInfoList getCategoryListAuthToGroupInCategory(String groupID, String categoryID, int perm) throws OZCPException**
- **boolean modifyItemGroupAuth(String groupID, String itemID, int perm) throws OZCPException**
- **int getGroupAuthToItem(String groupID, String itemID) throws OZCPException**
- **IOZGroupInfoList getGroupListAuthToItem(String itemID, int perm) throws OZCPException**
- **IOZItemInfoList getItemListAuthToGroup(String groupID, int perm)**

throws OZCPException

- **boolean modifyCategoryUserAuth(String userID, String categoryID, int perm) throws OZCPException**
- **int getUserAuthToCategory(String userID, String categoryID) throws OZCPException**
- **IOZUserInfoList getUserListAuthToCategory(String categoryID, int perm) throws OZCPException**
- **IOZItemInfoList getItemListAuthToUserInCategory(String userID, String categoryID, int perm) throws OZCPException**
- **IOZCategoryInfoList getCategoryListAuthToUserInCategory(String userID, String categoryID, int perm) throws OZCPException**
- **boolean modifyItemUserAuth(String userID, String itemID, int perm) throws OZCPException**
- **int getUserAuthToItem(String userID, String itemID) throws OZCPException**
- **IOZUserInfoList getUserListAuthToItem(String itemID, int perm) throws OZCPException**
- **IOZItemInfoList getItemListAuthToUser(String userID, int perm) throws OZCPException**
- **int setUserDefinedResourceSync(String alias, String[] itemIDs, String comment, String[] errorMsg) throws OZCPException**
- **Vector encryptFile(String[] itemIDs) throws OZCPException**
- **Vector encryptFile(String categoryID, boolean isRecursive) throws OZCPException**
- **Vector decryptFile(String[] itemIDs) throws OZCPException**
- **Vector decryptFile(String categoryID, boolean isRecursive) throws OZCPException**

Constructor Detail

- **RepositoryEx**

Prototype	**//Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우**
	`public RepositoryEx(String ip, int port, String id, String pw) throws OZCPException`
	
	// Daemon 타입 - 오즈 서버 타입이 TCP Server인 경우 (자동 로그인)
	`public RepositoryEx(String ip, int port, String id, String pw, boolean useUSL) throws OZCPException`
	
	//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우
	`public RepositoryEx(String iurl, String id, String pw) throws OZCPException`
	
	//Servlet 타입 - 오즈 서버 타입이 HTTP Server인 경우 (자동 로그인)
	`public RepositoryEx(String iurl, String id, String pw, boolean useUSL) throws OZCPException`	

Argument	*ip*	Servlet 타입 오즈 서버의 URL ex) String url = "http://127.0.0.1/oz/server";
	port	Daemon 타입 오즈 서버의 IP ex) String ip = "127.0.0.1";
	id	Daemon 타입 오즈 서버의 포트 번호 ex) int port = 8003;
	pw	사용자 아이디 ex) String id = "admin";
	useUSL	USL 사용 여부 ex) boolean useUSL = false;

Method Detail

■ setRepositoryConfig

Prototype	`public void setRepositoryConfig(SortProperties prop) throws OZCPException`
Definition	리파지토리 설정을 변경합니다.
Argument	*SortProperties* 변경할 설정

■ getRepositoryConfig

Prototype	`public SortProperties getRepositoryConfig() throws OZCPException`
Definition	리파지토리 설정을 가져옵니다.

■ **reloadRepository**

Prototype	`public void reloadRepository() throws OZCPException`
Definition	리파지토리를 재시작합니다.

■ **createItem**

Prototype	`public String[] createItem(String[] itemNames, String[] itemDescs, String[] categoryIDs, boolean[] isComp, InputStream[] itemStreams, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException`
	
	`public String[] createItem(String[] itemNames, String[] itemDescs, String[] categoryIDs, boolean[] isComp, InputStream[] itemStreams, String comment, long[] createdTimes, int[] errorCodes, String[] errorMsg) throws OZCPException`	

Definition	아이템을 생성하고 생성된 아이템 ID를 반환합니다. 시간을 지정하여 아이템을 생성하고 생성된 아이템 ID를 반환합니다.(NONE 타입은 지원 안 함)	
Argument	*itemNames*	새로 생성할 아이템의 이름
	itemDescs	새로 생성할 아이템의 설명 내용
	categoryIDs	새로 생성할 아이템이 카테고리 이름
	isComp	압축 여부
	itemStreams	새로 생성할 아이템의 입력 스트림
	comment	주석문
	createdTimes	아이템 생성 시간 0 또는 음수로 설정할 경우 현재 시간으로 적용됨
	errorCodes	에러 발생 시 반환되는 에러 코드
	errorMsg	에러 발생 시 반환되는 에러 메세지

■ **modifyItemName**

Prototype	`public String modifyItemName(String itemID, String itemName, String comment) throws OZCPException`	
Definition	아이템 이름을 변경하고 변경된 아이템 ID를 반환합니다.	
Argument	*itemID*	이름을 변경할 아이템 ID
	new_itemName	변경할 아이템 이름

	comment	주석문

■ **modifyItemDatetime**

Prototype	public boolean modifyItemDatetime(String itemID, long datetime) throws OZCPException	
Definition	아이템 생성 시간을 변경합니다.(NONE 타입은 지원 안 함)	
Argument	*itemID*	시간을 변경할 아이템의 ID
	datetime	변경할 시간 0 또는 음수로 설정할 경우 현재 시간으로 적용됨

■ **deleteItem**

Prototype	public boolean[] deleteItem(String[] itemIDs, boolean[] isDestorys, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException	
Definition	지정한 아이템을 리파지토리에서 삭제하고 아이템 삭제 성공 여부를 가져옵니다.	
Argument	*itemIDs*	삭제할 아이템 ID
	isDestroys	아이템을 영구 삭제할 지 여부
	comment	주석문
	errCodes	에러 발생 시 반환되는 에러 코드
	errMsgs	에러 발생 시 반환되는 에러 메세지

■ **unDeleteItem**

Prototype	public boolean[] unDeleteItem(String[] itemIDs, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException	
Definition	삭제된 아이템을 복원하고 복원 성공 여부를 가져옵니다. ※ 참고사항 : 아이템 삭제 시 "isDestroys=false"로 설정하였을 경우 삭제된 아이템의 복원이 가능합니다.	
Argument	*itemIDs*	복원할 아이템 ID
	comment	주석문
	errCodes	에러 발생 시 반환되는 에러 코드
	errMsgs	에러 발생 시 반환되는 에러 메세지

■ **modifyItemDesc**

Prototype	public boolean modifyItemDesc(String itemID, String iDesc) throws OZCPException	

Definition	지정한 아이템 ID에 해당하는 아이템의 설명을 변경하고 변경 성공 여부를 반환합니다.	
Argument	*itemID*	설명을 변경할 아이템의 ID
	desc	아이템 설명 내용

■ **getItemInfo**

Prototype	`public IOZItemInfo getItemInfo(String itemID) throws OZCPException`
Definition	지정한 아이템 ID에 해당하는 아이템의 정보를 가져옵니다.
Argument	*itemID* 정보를 가져올 아이템 ID

■ **getItemInfoListInCategory**

Prototype	`public IOZItemInfoList getItemInfoListInCategory(String categoryID, boolean isRecursive) throws OZCPException`	
Definition	아이템 정보를 가져옵니다.(NONE 타입은 지원 안 함)	
Argument	*categoryID*	카테고리 ID(Full Path로 설정)
	isRecursive	하위 카테고리 포함 여부

■ **hasTheItem**

Prototype	`public boolean hasTheItem(String itemID) throws OZCPException`
Definition	지정한 아이템의 존재 여부를 확인합니다.
Argument	*itemID* 체크할 아이템의 이름

■ **getItemsUnCondition**

Prototype	`public InputStream[] getItemsUnCondition(String[] itemIDs, boolean[] isComp, boolean[] isObjStream, String[] categoryIDs, int[] errorCodes, String[] errorMsg) throws OZCPException`	
Definition	지정한 아이템ID을 조건 없이 가져옵니다.	
Argument	*itemIDs*	가져올 아이템 ID
	isComp	압축 여부
	isObjStream	ODI 객체 스트림 필요 여부
	errCodes	에러 발생 시 반환되는 에러 코드
	errMsgs	에러 발생 시 반환되는 에러 메세지

■ getItems

Prototype	public InputStream[] getItems(String[] itemIDs, long[] modifiedTimes, boolean[] isComp, boolean[] isObjStream, String[] categoryIDs, int[] errorCodes, String[] errorMsg) throws OZCPException	
Definition	지정한 아이템ID에 해당하는 아이템을 가져옵니다.	
Argument	*itemIDs*	가져올 아이템 ID
	modifiedTimes	아이템을 최종 변경한 시간
	isComp	압축 여부
	isObjStream	ODI 객체 스트림 필요 여부
	errCodes	에러 발생 시 반환되는 에러 코드
	errMsgs	에러 발생 시 반환되는 에러 메세지

■ checkOutItem

Prototype	public InputStream[] checkOutItem(String itemIDs[], String[] localCheckOutFolders, long[] localFileTimes, boolean[] isComp, int[] errorCodes, String[] errorMsg) throws OZCPException public InputStream[] checkOutItem(String itemIDs[], String[] localCheckOutFolders, long[] localFileTimes, boolean[] isComp, String[] checkOutCmts, int[] errorCodes, String[] errorMsg) throws OZCPException	
Definition	지정한 아이템을 체크 아웃할 폴더에 체크 아웃합니다.	
Argument	*itemIDs*	체크 아웃할 아이템의 ID
	localCheckOut Folders	체크 아웃할 폴더 이름
	localFileTimes	체크 아웃할 아이템의 로컬 파일 시간
	isComp	압축 여부
	checkOutCmts	주석
	errorCodes	에러 발생 시 반환되는 에러 코드
	errorMsg	에러 발생 시 반환되는 에러 메세지

■ checkInItem

Prototype	public boolean[] checkInItem(String[] itemIDs, boolean[] isComp, InputStream[] iStreams, String comment, boolean[] keepCheckOut, int[] errorCodes, String[] errorMsg) throws OZCPException

Prototype	`public boolean[] checkInItem(String[] itemIDs, boolean[] isComp, InputStream[] itemStreams, String comment, boolean[] keepCheckOut, long[] datetimes, int[] errorCodes, String[] errorMsg) throws OZCPException`
Definition	지정한 아이템을 체크인합니다. 시간을 지정하여 아이템을 체크인합니다.(NONE 타입은 지원 안 함)
Argument	*itemIDs* 체크인할 아이템의 ID *isComp* 압축 여부 *itemStreams* 아이템의 입력 스트림 *comment* 주석문 *keepCheckOut* 체크 아웃 상태를 유지할지 여부 *datetimes* 설정할 시간 0 또는 음수로 설정할 경우 현재 시간으로 적용됨 *errorCodes* 에러 발생 시 반환되는 에러 코드 *errorMsg* 에러 발생 시 반환되는 에러 메세지

■ **undoCheckOutItem**

Prototype	`public InputStream[] undoCheckOutItem(String[] itemIDs, boolean[] isReplaces, boolean[] isComp, int[] errorCodes, String[] errorMsg) throws OZCPException`
Definition	지정한 아이템을 체크 아웃 취소합니다.
Argument	*itemIDs* 체크 아웃 취소할 아이템의 ID *isReplaces* 로컬 작업 폴더의 아이템을 가져올지 여부 *errCodes* 에러 발생 시 반환되는 에러 코드 *errMsgs* 에러 발생 시 반환되는 에러 메세지

■ **isCheckOutUser**

Prototype	`public boolean[] isCheckOutUser(String[] itemIDs) throws OZCPException`
Definition	현재 사용자가 지정한 아이템을 체크 아웃했는지 여부를 확인합니다.
Argument	*itemIDs* 아이템 ID

■ **rollBackItem**

Prototype	`public boolean rollBackItem(String itemID, int itemVersion, String comment) throws OZCPException`

Definition	지정한 아이템을 지정한 버전으로 복원시킵니다.	
Argument	*itemID*	복원시킬 아이템의 ID
	itemVersion	복원시킬 버전
	comment	주석문

■ getItemByVersion

Prototype	`public InputStream getItemByVersion(String itemID, int version, boolean isComp) throws OZCPException`
Definition	지정한 아이템 ID에 해당하는 아이템 중 지정한 버전의 아이템을 가져옵니다.
Argument	*itemID* 가져올 아이템의 ID
	version 가져올 아이템의 버전
	isComp 압축 여부

■ getHistoryItemList

Prototype	`public IOZHistoryInfoList getHistoryItemList(String itemID) throws OZCPException`
Definition	지정한 아이템의 히스토리 정보를 가져옵니다.
Argument	*itemID* 히스토리 정보를 가져올 아이템의 ID

■ getHistoryItemByDatetime

Prototype	`public InputStream getHistoryItemByDatetime(String itemID, long datetime, boolean isComp, boolean isObjStream) throws OZCPException`
Definition	지정한 시간 기준으로 최근 아이템을 가져옵니다.(NONE 타입은 지원 안 함)
Argument	*itemID* 가져올 아이템 ID
	datetime 가져올 시간
	isComp 압축 여부
	isObjStream 가져올 아이템이 ODI인 경우 obj stream으로 가져올지 여부

■ getDeleteHistoryItemInfo

Prototype	`public IOZHistoryInfoList getDeleteHistoryItemInfo(String itemID) throws OZCPException`
Definition	해당 삭제된 아이템의 정보를 가져옵니다. ※ 참고사항 : "isDestroys=false"로 설정하여 삭제한 아이템의 정보만 가져옵니다.

Argument	*itemID*	정보를 가져올 아이템의 ID

■ **removeHistoryItem**

Prototype	public boolean removeHistoryItem(String itemID, int itemVersion) throws OZCPException	
Definition	지정한 아이템에 대해 특정 버전의 히스토리를 삭제합니다.	
Argument	*itemID*	히스토리를 삭제할 아이템의 ID
	itemversion	히스토리를 삭제할 버전

■ **modifyHistoryItemComment**

Prototype	public boolean modifyHistoryItemComment(String itemID, int itemVersion, String newComment) throws OZCPException	
Definition	아이템 히스토리 중 특정 버전의 주석을 수정합니다.	
Argument	*itemID*	아이템 ID(Full Path로 설정)
	itemversion	주석을 수정할 버전
	newComment	수정할 주석

■ **modifyHistoryItemDatetime**

Prototype	public boolean modifyHistoryItemDatetime(String itemID, int itemVersion, long datetime) throws OZCPException	
Definition	지정한 버전의 아이템 시간을 변경합니다.(NONE 타입은 지원 안 함)	
Argument	*itemID*	시간을 변경할 아이템의 ID
	itemVersion	시간을 변경할 버전
	datetime	변경할 시간 0 또는 음수로 설정할 경우 현재 시간으로 적용됨

■ **createCategory**

Prototype	public String[] createCategory(String[] categoryNames, String[] pCategoryIDs, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException	
Definition	카테고리를 새로 생성하고 생성된 카테고리의 ID를 반환합니다.	
Argument	*categoryName*	생성할 카테고리의 이름
	pCategoryIDs	생성할 카테고리의 상위 카테고리 ID
	comment	주석문
	errorCode	에러 코드
	errorMsg	에러 메시지

■ modifyCategoryDesc

Prototype	`public boolean modifyCategoryDesc(String categoryID, String categoryDesc) throws OZCPException`	
Definition	카테고리 설명을 변경합니다.(NONE 타입은 지원 안 함)	
Argument	*categoryID*	카테고리 ID(Full Path로 설정)
	categoryDesc	카테고리 설명

■ modifyCategoryName

Prototype	`public String modifyCategoryName(String categoryID, String new_CategoryName, String comment) throws OZCPException`	
Definition	지정한 카테고리 ID에 해당하는 카테고리의 이름을 변경합니다.	
Argument	*categoryID*	이름을 변경할 카테고리의 ID
	new_categoryName	변경할 카테고리 이름
	comment	주석문

■ deleteCategory

Prototype	`public boolean[] deleteCategory(String[] categoryIDs, boolean[] isDestroys, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException`	
Definition	지정한 카테고리 ID에 해당하는 카테고리를 삭제합니다.	
Argument	*categoryIDs*	삭제할 카테고리의 ID
	isDestroys	카테고리를 영구 삭제할 지 여부
	comment	주석문
	errorCode	에러 코드
	errorMsg	에러 메시지

■ unDeleteCategory

Prototype	`public boolean[] unDeleteCategory(String[] categoryIDs, String comment, int[] errorCodes, String[] errorMsg) throws OZCPException`	
Definition	삭제된 카테고리를 복원하고 복원 성공 여부를 가져옵니다. ※ 참고사항 : 카테고리 삭제 시 "isDestroys=false"로 설정하였을 경우 삭제된 카테고리의 복원가 가능합니다.	
Argument	*categoryIDs*	삭제된 카테고리의 ID
	comment	주석문
	errorCode	에러 코드

	errorMsg	에러 메시지

■ **getItemCountInCategory**

Prototype	`public int getItemCountInCategory(String categoryID) throws OZCPException`
Definition	지정한 카테고리에 속해 있는 모든 아이템의 개수를 가져옵니다.
Argument	*categoryID* 아이템의 개수를 얻을 카테고리의 ID

■ **getItemListInCategory**

Prototype	`public IOZItemInfoList getItemListInCategory(String categoryID) throws OZCPException`
Definition	지정한 카테고리에 속해 있는 모든 아이템 정보를 가져옵니다.
Argument	*categoryID* 아이템 정보를 얻을 카테고리 ID

■ **getCategoryListInCategory**

Prototype	`public IOZCategoryInfoList getCategoryListInCategory(String categoryID) throws OZCPException`
Definition	지정한 카테고리에 속해 있는 모든 하위 카테고리 정보를 가져옵니다.
Argument	*categoryID* 아이템의 개수를 얻을 카테고리의 ID

■ **getCategoryIdOfItem**

Prototype	`public String getCategoryIdOfItem(String itemID) throws OZCPException`
Definition	지정한 아이템이 존재하는 카테고리의 ID를 반환합니다.
Argument	*itemID* 카테고리의 ID를 얻을 아이템 ID

■ **getCategoryInfo**

Prototype	`public IOZCategoryInfo getCategoryInfo(String categoryID) throws OZCPException`
Definition	지정한 카테고리의 정보를 가져옵니다.
Argument	*categoryID* 정보를 가져올 카테고리 ID

■ **getDeletedItemListInCategory**

Prototype	`public IOZItemInfoList getDeletedItemListInCategory(String categoryID) throws OZCPException`
Definition	지정한 카테고리의 삭제된 아이템 정보를 가져옵니다. ※ 참고사항 : "isDestroys=false"로 설정하여 삭제한 아이템의 정보만 가져옵니다.
Argument	*categoryID* 삭제된 아이템 정보를 가져올 카테고리 ID

■ **transferCategory**

Prototype	`public boolean transferCategory(String categoryID, String target_CategoryID) throws OZCPException`
Definition	지정한 카테고리를 다른 카테고리로 이동하고 카테고리 이동 성공 여부를 가져옵니다.
Argument	*categoryID* 카테고리를 이동할 카테고리 ID
	target_CategoryID 이동할 카테고리 ID

■ **transferItem**

Prototype	`public boolean transferItem(String[] itemIDs, String target_CategoryID) throws OZCPException`
Definition	지정한 아이템의 카테고리를 이동하고 카테고리 이동 성공 여부를 가져옵니다.
Argument	*itemIDs* 카테고리를 이동할 아이템 ID
	target_CategoryID 이동할 카테고리 ID

■ **createUser**

Prototype	`public String createUser(String userName, String password, String desc) throws OZCPException`
Definition	새로운 사용자를 생성하고 생성된 사용자의 ID를 반환합니다.
Argument	*userName* 사용자 이름
	password 패스워드
	desc 사용자에 대한 설명 내용

■ **modifyUserName**

Prototype	`public String modifyUserName(String userID, String userName) throws OZCPException`
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 변경하고 변경 성공 여부를 반환합니다.
Argument	*userID* 사용자 ID

	userName	변경할 사용자 이름

■ **modifyUserPassword**

Prototype	`public boolean modifyUserPassword(String userID, String old_password, String new_password) throws OZCPException`
Definition	지정한 사용자 ID에 해당하는 사용자의 패스워드를 변경하고 변경 성공 여부를 반환합니다.
Argument	*userID* 사용자 ID
	old_password 변경 전 패스워드
	new_password 변경 후 패스워드

■ **modifyUserDesc**

Prototype	`public boolean modifyUserDesc(String userID, String desc) throws OZCPException`
Definition	지정한 사용자 ID에 해당하는 사용자의 설명 내용을 변경하고 변경 성공 여부를 반환합니다.
Argument	*userID* 사용자 ID
	desc 변경할 설명 내용

■ **deleteUser**

Prototype	`public boolean deleteUser(String userID) throws OZCPException`
Definition	지정한 사용자 ID에 해당하는 사용자의 모든 정보를 삭제하고 삭제 성공 여부를 반환합니다.
Argument	*userID* 삭제할 사용자의 ID

■ **getUserInfo**

Prototype	`public IOZUserInfo getUserInfo(String userID) throws OZCPException`
Definition	지정한 사용자 ID에 해당하는 사용자의 사용자 정보를 가져옵니다.
Argument	*userID* 사용자 정보를 가져올 사용자 ID

■ **checkUserPassword**

Prototype	`public boolean checkUserPassword(String userID, String password) throws OZCPException`
Definition	사용자의 패스워드가 맞는지 여부를 체크합니다.
Argument	*userID* 패스워드를 체크할 사용자 ID

password	체크할 패스워드

■ **getUserInfoList**

Prototype	`public IOZUserInfoList getUserInfoList() throws OZCPException`
Definition	모든 사용자의 정보를 가져옵니다.

■ **DisableLogin**

Prototype	`public void DisableLogin(String userID)`
Definition	지정한 사용자 ID의 사용자를 로그인하지 못하도록 설정합니다.
Argument	*userID* 로그인하지 못하도록 할 사용자 ID

■ **EnableLogin**

Prototype	`public void EnableLogin(String userID)`
Definition	지정한 사용자 ID의 사용자를 로그인 가능하도록 설정합니다.
Argument	*userID* 로그인 가능하도록 할 사용자 ID

■ **isCheckAdmin**

Prototype	`public boolean isCheckAdmin(String userID) throws OZCPException`
Definition	지정한 사용자 ID에 해당하는 사용자가 리파지토리의 관리자인지 여부를 확인합니다.
Argument	*userID* 관리자인지 체크할 사용자 ID

■ **disableUserLogin**

Prototype	`public void disableUserLogin(String userID) throws OZCPException`
Definition	지정한 사용자 ID의 사용자를 로그인하지 못하도록 설정합니다.
Argument	*userID* 로그인하지 못하도록 할 사용자 ID

■ **enableUserLogin**

Prototype	`public void enableUserLogin(String userID) throws OZCPException`
Definition	지정한 사용자 ID의 사용자를 로그인 가능하도록 설정합니다.
Argument	*userID* 로그인 가능하도록 할 사용자 ID

■ **isLoginEnabled**

Prototype	`public boolean isLoginEnabled(String userID) throws OZCPException`
Definition	지정한 사용자 ID의 사용자가 로그인이 가능한지 여부를 가져옵니다.
Argument	*userID* 로그인 가능 가능 여부를 체크할 사용자 ID

■ **createGroup**

Prototype	`public String createGroup(String gName, String pGroupID, String desc) throws OZCPException`
Definition	새로운 그룹을 생성하고 생성된 그룹의 ID를 반환합니다.
Argument	*gName* 새로 생성할 그룹의 이름
	pGroupID 새로 생성할 그룹의 상위 그룹 ID
	desc 그룹에 대한 설명 내용

■ **modifyGroupName**

Prototype	`public String modifyGroupName(String groupID, String gName) throws OZCPException`
Definition	지정한 사용자 ID에 해당하는 사용자의 이름을 변경하고 변경 성공 여부를 반환합니다.
Argument	*groupID* 그룹의 이름을 변경할 그룹 ID
	gName 변경할 그룹의 이름

■ **modifyGroupDesc**

Prototype	`public boolean modifyGroupDesc(String groupID, String desc) throws OZCPException`
Definition	지정한 사용자 ID에 해당하는 사용자의 설명을 변경하고 변경 성공 여부를 반환합니다.
Argument	*groupID* 그룹의 설명을 변경할 그룹 ID
	desc 변경할 그룹의 설명

■ **deleteGroup**

Prototype	`public boolean deleteGroup(String groupID) throws OZCPException`
Definition	지정한 그룹 ID에 해당하는 그룹을 삭제하고 삭제 성공 여부를 반환합니다.
Argument	*groupID* 삭제할 그룹 ID

■ **createUserInGroup**

Prototype	`public String createUserInGroup(String userName, String password, String groupID, String desc) throws OZCPException`
Definition	지정한 그룹에 새로운 사용자를 생성하고 생성된 사용자의 ID를 반환합니다.
Argument	*userName* 새로 생성할 사용자의 이름
	password 패스워드
	groupID 사용자를 생성할 그룹 ID
	desc 사용자에 대한 설명 내용

■ **transferUser**

Prototype	`public boolean transferUser(String userID, String target_GroupID) throws OZCPException`
Definition	지정한 사용자의 그룹을 이동하고 그룹 이동 성공 여부를 가져옵니다.
Argument	*userID* 그룹을 이동할 사용자 ID
	target_groupID 이동할 그룹 ID

■ **transferGroup**

Prototype	`public boolean transferGroup(String groupID, String target_GroupID) throws OZCPException`
Definition	지정한 그룹을 다른 그룹으로 이동하고 그룹 이동 성공 여부를 가져옵니다.
Argument	*groupID* 그룹을 이동할 그룹 ID
	target_groupID 이동할 그룹 ID

■ **getUserInfoListInGroup**

Prototype	`public IOZUserInfoList getUserInfoListInGroup(String groupID) throws OZCPException`
	
	`public IOZUserInfoList getUserInfoListInGroup(String groupID, boolean isRecursive) throws OZCPException`

Definition	지정한 그룹 ID에 등록되어 있는 모든 사용자의 정보를 가져옵니다. 그룹에 추가된 사용자 정보를 가져옵니다.(NONE 타입은 지원 안 함)
Argument	*groupID* 사용자의 정보를 가져올 그룹 ID
	isRecursive 하위 그룹 사용자 포함 여부

■ **getGroupInfo**

Prototype	`public IOZGroupInfo getGroupInfo(String groupID) throws OZCPException`
Definition	지정한 그룹 ID에 해당하는 그룹의 정보를 가져옵니다.
Argument	*groupID* 그룹 정보를 가져올 그룹 ID

■ **getGroupInfoListInGroup**

Prototype	`public IOZGroupInfoList getGroupInfoListInGroup(String groupID, boolean isRecursive) throws OZCPException`
Definition	그룹에 추가된 그룹 정보를 가져옵니다.(NONE 타입은 지원 안 함)
Argument	*groupID* 그룹 ID(Full Path로 설정)
	isRecursive 하위 그룹 포함 여부

■ **getSubGroupInfoList**

Prototype	`public IOZGroupInfoList getSubGroupInfoList(String groupID) throws OZCPException`
Definition	지정한 그룹의 하위 그룹 정보를 가져옵니다.
Argument	*groupID* 하위 그룹의 정보를 가져올 그룹 ID

■ **getParentGroupInfo**

Prototype	`public IOZGroupInfo getParentGroupInfo(String groupID) throws OZCPException`
Definition	지정한 그룹의 상위 그룹의 정보를 가져옵니다.
Argument	*groupID* 상위 그룹의 정보를 가져올 그룹 ID

■ **getGroupIdOfUser**

Prototype	`public String getGroupIdOfUser(String userID) throws OZCPException`
Definition	지정한 사용자가 속해 있는 그룹의 정보를 가져옵니다.
Argument	*userID* 그룹 정보를 가져올 사용자 ID

■ **addGroupAdmin**

Prototype	`public boolean addGroupAdmin(Object sessionID, String userID, String groupID) throws OZRepositoryException`
Definition	지정한 그룹의 그룹 관리자를 추가하고 추가 성공 여부를 반환합니다.
Argument	*userID* 그룹 관리자로 추가할 사용자 ID

	groupID	그룹 관리자를 추가할 그룹 ID

■ removeGroupAdmin

Prototype	`public boolean removeGroupAdmin(String userID, String groupID) throws OZCPException`	
Definition	지정한 그룹의 그룹 관리자를 해제하고 해제 성공 여부를 반환합니다.	
Argument	*userID*	그룹 관리자 권한을 해제할 사용자 ID
	groupID	그룹 관리자를 해제할 그룹 ID

■ isUserGroupAdmin

Prototype	`public boolean isUserGroupAdmin(String userID, String groupID) throws OZCPException`	
Definition	지정한 사용자 ID에 해당하는 사용자가 해당 그룹의 관리자인지 여부를 확인합니다.	
Argument	*userID*	관리자인지 체크할 사용자 ID
	groupID	그룹 ID

■ getUserAdminListInGroup

Prototype	`public IOZUserInfoList getUserAdminListInGroup(String groupID) throws OZCPException`	
Definition	지정한 그룹의 그룹 관리자 정보를 가져옵니다.	
Argument	*groupID*	그룹 관리자 정보를 가져올 그룹 ID

■ distributeRepository

Prototype	`public InputStream distributeRepository(String[] itemIDs) throws OZCPException`
	
	`public InputStream distributeRepository(String categoryID, boolean isRecursive) throws OZCPException`	

Definition	아이템을 배포 파일 형식으로 저장합니다. 지정한 카테고리의 아이템을 배포 파일 형식으로 저장합니다.	
Argument	*itemIDs*	아이템 ID(Full Path로 설정)
	categoryID	카테고리 ID
	isRecursive	하위 카테고리 포함 여부

■ **uploadZipItem**

Prototype	public boolean uploadZipItem(String categoryID, InputStream stream) throws OZCPException	
Definition	압축된 아이템을 지정한 카테고리에 압축 해제한 후 압축 해제 성공 여부를 반환합니다.	
Argument	*categoryID*	카테고리 ID
	Stream	압축한 스트림

■ **modifyCategoryGroupAuth**

Prototype	public boolean modifyCategoryGroupAuth(String categoryID, String groupID, int perm) throws OZCPException	
Definition	지정한 카테고리에 대한 그룹의 권한 정보를 변경하고 변경 성공 여부를 가져옵니다.	
Argument	*categoryID*	권한 정보를 변경할 카테고리 ID
	groupID	권한 정보를 변경할 그룹 ID
	perm	권한 정보

■ **getGroupAuthToCategory**

Prototype	public int getGroupAuthToCategory(String categoryID, String groupID) throws OZCPException	
Definition	지정한 카테고리에 대한 지정한 그룹의 그룹 권한을 가져옵니다.	
Argument	*categoryID*	권한 정보를 가져올 카테고리 ID
	groupID	그룹 ID

■ **getGroupListAuthToCategory**

Prototype	public IOZGroupInfoList getGroupListAuthToCategory(String categoryID, int perm) throws OZCPException	
Definition	지정한 카테고리 ID에 대해 perm 이상의 권한을 가지고 있는 모든 사용자 정보를 가져옵니다.	
Argument	*categoryID*	권한 정보를 가져올 카테고리 ID
	perm	그룹 ID

■ **getItemListAuthToGroupInCategory**

Prototype	public IOZItemInfoList getItemListAuthToGroupInCategory(String groupID, String

	categoryID, int perm) throws OZCPException	
Definition	지정한 카테고리 ID에 대해 perm 이상의 권한을 가지고 있는 모든 아이템 정보를 가져옵니다.	
Argument	*groupID*	그룹 ID
	categoryID	카테고리 ID
	perm	권한 정보

■ **getCategoryListAuthToGroupInCategory**

Prototype	public IOZCategoryInfoList getCategoryListAuthToGroupInCategory(String groupID, String categoryID, int perm) throws OZCPException	
Definition	지정한 그룹 ID에 해당하는 그룹에게 perm 이상의 권한이 부여된 모든 카테고리의 정보를 가져옵니다.	
Argument	*groupID*	그룹 ID
	categoryID	카테고리 ID
	perm	권한 정보

■ **modifyItemGroupAuth**

Prototype	public boolean modifyItemGroupAuth(String groupID, String itemID, int perm) throws OZCPException	
Definition	지정한 아이템 ID에 대한 지정한 그룹 ID의 권한 정보를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*groupID*	그룹 ID
	itemID	카테고리 ID
	perm	권한 정보

■ **getGroupAuthToItem**

Prototype	public int getGroupAuthToItem(String groupID, String itemID) throws OZCPException	
Definition	지정한 아이템 ID에 대한 지정한 그룹 ID의 권한 정보를 가져옵니다.	
Argument	*groupID*	그룹 ID
	itemID	아이템 ID

■ **getGroupListAuthToItem**

Prototype	`public IOZGroupInfoList getGroupListAuthToItem(String itemID, int perm) throws OZCPException`	
Definition	지정한 아이템 ID에 대해 perm 이상의 권한이 부여된 모든 그룹 정보를 가져옵니다.	
Argument	*itemID*	아이템 ID
	perm	권한 정보

■ **getItemListAuthToGroup**

Prototype	`public IOZItemInfoList getItemListAuthToGroup(String groupID, int perm) throws OZCPException`	
Definition	지정한 그룹 ID에 해당하는 그룹에게 perm 이상의 권한이 부여된 모든 아이템의 정보를 가져옵니다.	
Argument	*groupID*	그룹 ID
	perm	권한 정보

■ **modifyCategoryUserAuth**

Prototype	`public boolean modifyCategoryUserAuth(String uID, String cID, int perm) throws OZCPException`	
Definition	지정한 카테고리 ID에 대한 지정한 사용자 ID의 권한 정보를 변경하고 변경 성공 여부를 반환합니다.	
Argument	*userID*	사용자 ID
	categoryID	카테고리 ID
	perm	권한 정보

■ **getUserAuthToCategory**

Prototype	`public int getUserAuthToCategory(String userID, String categoryID) throws OZCPException`	
Definition	지정한 카테고리 ID에 대한 지정한 사용자 ID의 권한 정보를 가져옵니다.	
Argument	*userID*	사용자 ID
	categoryID	카테고리 ID

■ **getUserListAuthToCategory**

Prototype	`public IOZUserInfoList getUserListAuthToCategory(String categoryID, int perm) throws OZCPException`
Definition	지정한 카테고리 ID에 대해 perm 이상의 권한이 부여된 모든 사용자 정보를

Argument	가져옵니다.	
	categoryID	카테고리 ID
	perm	권한 정보

■ **getItemListAuthToUserInCategory**

Prototype	`public IOZItemInfoList getItemListAuthToUserInCategory(Object sessionID, String userID, String categoryID, int perm) throws OZRepositoryException`
Definition	지정한 카테고리의 아이템 중에 사용자의 권한이 perm 이상인 모든 아이템 정보를 가져옵니다.
Argument	*userID* 사용자 ID
	categoryID 카테고리 ID
	perm 권한 정보

■ **getCategoryListAuthToUserInCategory**

Prototype	`public IOZCategoryInfoList getCategoryListAuthToUserInCategory(String userID, String categoryID, int perm) throws OZCPException`
Definition	지정한 카테고리의 하위 카테고리 중에 지정한 사용자의 권한이 perm 이상인 모든 카테고리의 목록을 가져옵니다.
Argument	*userID* 사용자 ID
	categoryID 카테고리 ID
	perm 권한 정보

■ **modifyItemUserAuth**

Prototype	`public boolean modifyItemUserAuth(String userID, String itemID, int perm) throws OZCPException`
Definition	지정한 아이템에 대한 사용자의 권한 정보를 변경하고 변경 성공 여부를 가져옵니다.
Argument	*userID* 사용자 ID
	itemID 아이템 ID
	perm 권한 정보

■ **getUserAuthToItem**

Prototype	`public int getUserAuthToItem(String userID, String itemID) throws OZCPException`

Definition	지정한 아이템에 대한 사용자의 권한 정보를 가져옵니다.	
Argument	*userID*	사용자 ID
	itemID	아이템 ID
	perm	권한 정보

■ **getUserListAuthToItem**

Prototype	`public IOZUserInfoList getUserListAuthToItem(String itemID, int perm) throws OZCPException`
Definition	지정한 아이템에 대한 사용자의 권한이 perm 이상인 모든 아이템 정보를 가져옵니다.
Argument	*itemID* 아이템 ID
	perm 권한 정보

■ **getItemListAuthToUser**

Prototype	`public IOZItemInfoList getItemListAuthToUser(String userID, int perm) throws OZCPException`
Definition	지정한 카테고리의 아이템 중에 사용자의 권한이 perm 이상인 모든 아이템 정보를 가져옵니다.
Argument	*userID* 사용자 ID
	perm 권한 정보

■ **setUserDefinedResorceSync**

Prototype	`public int setUserDefinedResourceSync(String alias, String[] itemIDs, String comment, String[] errorMsg) throws OZCPException`	
Definition	보고서를 사용자가 정의한 리소스로 동기화합니다.	
Argument	*alias*	ozudr.properties에서 설정한 앨리어스 이름
	itemIDs	보고서 전체 경로
	Comment	주석
	errorMsg	에러 메시지
Return	*responseCode*	응답 코드(1 : 전체 성공, 2 : 부분 성공, 3 : 실패)

■ **encryptFile**

Prototype	`public Vector encryptFile(String[] itemIDs) throws OZCPException`

Definition	`public Vector encryptFile(String categoryID, boolean isRecursive) throws OZCPException` 아이템을 암호화하여 저장하고, 암호화를 실패한 아이템 리스트를 리턴합니다. 암호화할 수 있는 파일은 ozf, oza, ozc, ozs, ozr 파일입니다. 이미 암호화된 아이템은 다시 암호화하지 않습니다.
Argument	*itemIDs* 암호화할 아이템 ID(Full Path로 설정) *categoryID* 암호화할 아이템을 가진 카테고리 ID(Full Path로 설정) *isRecursive* 하위 카테고리 포함 여부

■ **decryptFile**

Prototype	`public Vector decryptFile(String[] itemIDs) throws OZCPException` `public Vector decryptFile(String categoryID, boolean isRecursive) throws OZCPException`
Definition	아이템을 복호화하여 저장하고, 복호화를 실패한 아이템 리스트를 리턴합니다. 복호화할 수 있는 파일은 ozf, oza, ozc, ozs, ozr 파일입니다. 이미 복호화된 아이템은 다시 복호화하지 않습니다.
Argument	*itemIDs* 복호화할 아이템 ID(Full Path로 설정) *categoryID* 복호화할 아이템을 가진 카테고리 ID(Full Path로 설정) *isRecursive* 하위 카테고리 포함 여부

오즈 리파지토리 구현

리파지토리 매니저 호출 방식

리파지토리 매니저에서 API(oz.framework.cp.client.OZFrameworkAPI)를 사용하는 방법은 기존과 동일하며 리파지토리 매니저에서 API를 호출할 경우 아래와 같은 순서로 API가 처리됩니다.

- 리파지토리 매니저에서 서버에 요청하기 전에 리파지토리 매니저가 새로운 CP를 지원하는 버전인지 확인합니다.
- 리파지토리 매니저가 새로운 CP를 지원하는 버전일 경우 현재 호출한 매개 변수를 새 버전의 CP에 맞게 변환하여 새로 추가된 API를 호출합니다.
- 리파지토리 매니저가 새로운 CP를 지원하지 않는 버전일 경우 이전과 동일한 방식으로 API를 호출합니다.

※ 참고사항 : 새로 추가된 API는 서버의 RepositoryEx API와 형태가 동일하며 클라이언트에서 직접 사용하지 않습니다.

개발 시 주의사항

- Adapter 추가 개발 시 IOZRepository 인터페이스를 반드시 구현하여야 합니다.
- Adapter 추가 개발 시 Category를 개발할 경우 반드시 Item을 개발하여야 하며 Group을 개발하는 경우에는 반드시 User를 개발하여야 합니다.
- 현재 리파지토리에서 지원하지 않는 기능은 다음과 같습니다.
  - 카테고리, 아이템 등을 삭제한 후 다시 복원하는 기능은 지원하지 않습니다.
  - 주석문 기능은 지원하지 않습니다.

Sample : RepositoryExSample.java

```
package sample;

import oz.framework.api.RepositoryEx;
```

```
import oz.util.SortProperties;
import oz.framework.repositoryex.info.*;
import oz.framework.repositoryex.adapter.impl.*;
import java.io.*;

public class RepositoryExSample {

   public RepositoryExSample() {
   }

   private static RepositoryEx rep = null;
   public static void main(String[] args) {
      String IP = "127.0.0.1"; //서버가 설치되어 있는 호스트 컴퓨터의 IP
      int PORT = 8003; //서버가 사용하는 TCP 포트
      String ID = "admin"; //default
      String PASSWORD = "admin"; //default
      try {
         rep = new RepositoryEx(IP, PORT, ID, PASSWORD, /*usl*/ false);
         //getConf();
         //setConf();
         //reloadConf();
         itemTest();
      }
      catch (Exception e) {
         e.printStackTrace();
      }
   }

   private static void getConf() throws Exception {
      //Repository configuration 정보를 가져옵니다.
      System.out.println("Repository.getRepositoryConfig()");
      SortProperties props = rep.getRepositoryConfig();
      props.list(System.out);
   }

   private static void setConf() throws Exception {
      //Repository Configuration 설정
      SortProperties prop = new SortProperties();
      prop.setProperty("REPOSITORY_TYPE", "RDB"); //RDB, FILESYSTEM, USER
      prop.setProperty("REPOSITORY_FILE_PATH", "c:/temp_repository");
      prop.setProperty("REPOSITORY_ITEM_NUMBER_PER_DIRECTORY", "100");
      prop.setProperty("REPOSITORY_HISTORY_ITEM_VALID_DAYS", "20");
      rep.setRepositoryConfig(prop);
   }

   private static void reloadConf() throws Exception {
      rep.reloadRepository();
```

```
    }

    private static void itemTest() throws Exception {
        String[] categoryNames = {
            "182",
            "183",
            "184",
            "185",
            "186"
        };
        String[] parentCategoryNames = {
            "/REQ",
            "/REQ1",
            "/REQ2",
            "/REQ3",
            "/REQ4"
        };
        int[] errCodes = new int[categoryNames.length];
        String[] errMsgs = new String[categoryNames.length];
        //카테고리를 생성합니다.
        String[] categoryIDs = rep.createCategory(categoryNames,
                                            parentCategoryNames,
                                            "comment", errCodes, errMsgs);
        //카테고리 ID를 출력합니다.
        for (int i = 0; i < categoryIDs.length; i++) {
            if (categoryIDs[i] != null) {
                System.out.println("categoryID=" + categoryIDs[i]);
            }
            else {
                System.out.println("error=[code=" + errCodes[i] + "] " +
                                errMsgs[i]);
            }
        }
//Item을 생성합니다.
        String[] itemNames = {
            "car.odi",
            "car.ozr",
            "car.oza",
            "cap.txt"
        };
        String[] itemDesc = {
            "자동차 odi 파일",
            "자동차 ozr 파일",
            "자동차 oza 파일",
            "0 byte text 파일 "
        };
        FileInputStream[] fis = new FileInputStream[itemNames.length];
        fis[0] = new FileInputStream("d:/REQ/182/car.odi");
```

```
fis[1] = new FileInputStream("d:/REQ/182/car.ozr");
fis[2] = new FileInputStream("d:/REQ/182/car.oza");
fis[3] = new FileInputStream("d:/cap.txt");
errCodes = new int[itemNames.length];
errMsgs = new String[itemNames.length];
String[] itemIDs = rep.createItem(itemNames, itemDesc, categoryIDs,
                            new boolean[] {false, false, false, false}
                            ,
                            //압축파일여부
                            fis, "comment", errCodes, errMsgs);
//ItemID ID를 출력합니다.
for (int i = 0; i < itemIDs.length; i++) {
   if (itemIDs[i] != null) {
      System.out.println("itemID=" + itemIDs[i]);
   }
   else {
      System.out.println("error=[code=" + errCodes[i] + "] " +
                     errMsgs[i]);
   }
}
errCodes = new int[itemNames.length];
errMsgs = new String[itemNames.length];

//Item을 가져옵니다.
download(rep.getItems(itemIDs, new long[] {0, 0, 0, 0}
                  ,
                  new boolean[] {true, false, false, false}
                  , //압축 여부
                  new boolean[] {false, false, false, false}
                  , //odi
                  categoryIDs, errCodes, errMsgs),
       new
       String[] {"d:/car.odi", "d:/car.ozr", "d:/car.oza",
       "d:/cap1.txt"});

//아이템 이름을 변경합니다.
String newItemID =
   rep.modifyItemName(itemIDs[3],
                   "modify" + System.currentTimeMillis() + ".txt",
                   "comment");
System.out.println("new Item ID =" + newItemID);
itemIDs[3] = newItemID;

//ItemID 정보를 가져와서 출력합니다.
OZItemInfoImpl info = new OZItemInfoImpl(rep.getItemInfo(newItemID));
System.out.println(info.toString());

//카테고리ID를 입력하여 ItemList을 얻어옵니다.
```

```
        OZItemInfoListImpl itemList =
            new OZItemInfoListImpl(rep.getItemListInCategory("/REQ1/182/"));

        //ItemList을 ItemInfo을 출력합니다.
        OZItemInfoImpl itemInfo = null;
        for (int i = 0; i < itemList.getSize(); i++) {
            itemInfo = new OZItemInfoImpl(itemList.getItemInfo(i));
            System.out.println(itemInfo.toString());
        }

        //아이템이 존재는지 검사
        System.out.println(rep.hasTheItem("/REQ1/182/car.odi"));

        //아이템이 존재하는 경우 Item을 checkin(update) 합니다.
        errCodes = new int[itemIDs.length];
        errMsgs = new String[itemIDs.length];
        fis[0] = new FileInputStream("d:/REQ/182/car.odi");
        fis[1] = new FileInputStream("d:/REQ/182/car.ozr");
        fis[2] = new FileInputStream("d:/REQ/182/car.oza");
        fis[3] = new FileInputStream("d:/cap.txt");
        boolean[] isResult =
            rep.checkInItem(itemIDs, new boolean[] {false, false, false, false}
                        , fis,
                        "comment",
                        new boolean[] {false, false, false, false}
                        , errCodes,
                        errMsgs);
            for (int i = 0; i < isResult.length; i++) {
            System.out.println("check in success = " + isResult[i]);
            if (!isResult[i]) {
                System.out.println("error [" + errCodes[i] + "] =" + errMsgs[i]);
            }
        }
//아이템을 삭제합니다.
        errCodes = new int[itemIDs.length];
        errMsgs = new String[itemIDs.length];
        isResult = rep.deleteItem(itemIDs, new boolean[] {true, true, true, true}
                              ,
                              "comment",
                              errCodes, errMsgs);
        for (int i = 0; i < isResult.length; i++) {
            System.out.println("delete item success = " + isResult[i]);
            if (!isResult[i]) {
                System.out.println("error [" + errCodes[i] + "] =" + errMsgs[i]);
            }
        }
//카테고리를 삭제합니다.
        errCodes = new int[itemIDs.length];
```

```
      errMsgs = new String[itemIDs.length];
      isResult = rep.deleteCategory(categoryIDs, new boolean[] {true, true,
true, true, true}
                     ,
                     "comment",
                     errCodes, errMsgs);
      for (int i = 0; i < isResult.length; i++) {
         System.out.println("delete category success = " + isResult[i]);
         if (!isResult[i]) {
            System.out.println("error [" + errCodes[i] + "] =" + errMsgs[i]);
         }
      }
//카테고리를 다른 카테고리로 이동합니다.
      rep.transferCategory("/REQ1", "/REQ");
//아이템을 다른 카테고리로 이동합니다.
      rep.transferItem(new String[] {"/test1.odi", "/test1.ozr"}
                    , "/REQ");
//카테고리를 다른 카테고리로 이동합니다.
      rep.transferCategory("/REQ/REQ1", "/REQ2");
//아이템을 다른 카테고리로 이동합니다.
      rep.transferItem(new String[] {"/REQ/test1.odi", "/REQ/test1.ozr"}
                    , "/");

      IOZCategoryInfo[]                          infos                          =
rep.getCategoryListInCategory("/",true).getCategoryInfos();
      String[] d = new String[infos.length-1];
      int index = 0;
      for(int i=0; i<d.length; i++) {
         if (!"/".equals(infos[i].getCategoryID())){

System.out.println("infos[i].getCategoryID()="+infos[i].getCategoryID());
            d[index++] = infos[i].getCategoryID();
         }
      }
      errCodes = new int[d.length];
      errMsgs = new String[d.length];
      boolean[] b = new boolean[d.length];
      java.util.Arrays.fill(b, true);
      isResult = rep.deleteCategory(d, b
                     ,
                     "comment",
                     errCodes, errMsgs);
      for (int i = 0; i < isResult.length; i++) {
         System.out.println("final delete category success = " + isResult[i]);
         if (!isResult[i]) {
            System.out.println("error [" + errCodes[i] + "] =" + errMsgs[i]);
         }
      }
```

```
    }

    private static void download(InputStream[] in, String[] fileName) throws
        Exception {
        for (int i = 0; i < in.length; i++) {
            if (in[i] != null) {
                byte[] buf = new byte[1024];
                int len;
                FileOutputStream fos = new FileOutputStream(fileName[i]);
                while ( (len = in[i].read(buf)) >= 0) {
                    fos.write(buf, 0, len);
                }
                in[i].close();
                fos.flush();
                fos.close();
                System.out.println("ok downlod = " + fileName[i]);
            }
            else {
                System.out.println(fileName[i] + " stream is null");
            }
        }
    }
}
```

Ⅳ. 오즈 리포트 뷰어 API

- OZLauncherDll을 이용한 호출 방법

OZLauncherDll을 이용한 호출 방법

Function Summary

- **void __stdcall SetPath(LPCTSTR strpath)**
- **void __stdcall SetCommand(LPCTSTR strcommand)**
- **BOOL __stdcall CreateOZViewer(LPCTSTR str_param, int n_type)**
- **void __stdcall Release()**

Function Detail

- **SetPath**

Prototype	void __stdcall SetPath(LPCTSTR strpath)
Definition	오즈 리포트 뷰어의 경로를 설정합니다.
Argument	*strpath* 오즈 리포트 뷰어 경로

- **SetCommand**

Prototype	void __stdcall SetCommand(LPCTSTR strcommand)
Definition	오즈 리포트 뷰어 호출시 선행 처리되어야 하는 명령을 지정합니다.
Argument	*strcommand* 선행 처리 명령을 설정한 문자열
Example	SetCommand("/locale ko/kr /mode alone /slp true /launchstring");

- **CreateOZViewer**

Prototype	BOOL __stdcall CreateOZViewer(LPCTSTR str_param, int n_type)	
Definition	오즈 리포트 뷰어를 생성하여 실행하는 함수입니다. 성공하면 true를 반환하고 실패하면 false를 반환합니다.	
Argument	*str_param*	오즈 리포트 뷰어 패러미터를 설정한 문자열입니다. 패러미터들은 "□n"으로 구분되어야 합니다.
	n_type	오즈 리포트 뷰어의 타입으로 2로 입력합니다.

Example	CreateOZViewer("connection.server=127.0.0.1\n connection.port=8003\n connection.reportname=/ozsample.ozr\n", 2);

■ Release()

Prototype	void __stdcall Release()
Definition	뷰어를 실행하면서 사용된 리소스를 모두 해제시키는 함수입니다. 뷰어의 사용이 끝나면 반드시 호출해 주어야 합니다.

Example	SetCommand("/String");
	SetPath("./");
	CreateOZViewer("connection.server=127.0.0.1\n
	toolbar.all=true\n
	information.debug=debug\n
	information.bmt=true\n
	connection.port=8003\n
	connection.reportname=/ozsample.ozr\n
	connection.compressedForm=true",1);
	result = GetResult();
	Release();

Ⅴ. RDB Store DataAction

- RDB Store DataAction 인터페이스
- RDB Store DataAction 구현

RDB Store DataAction 인터페이스

■ 인터페이스 방식

```
public interface IRDBDelegate extends IDelegate
{

   public void init(String param) throws Exception;
   public void close();
   public      String      insert(oz.framework.dac.OZDACItem      dac,
java.util.Dictionary parameters)
       throws Exception;
   public      String      update(oz.framework.dac.OZDACItem      dac,
java.util.Dictionary parameters)
       throws Exception;
   public      String      delete(oz.framework.dac.OZDACItem      dac,
java.util.Dictionary parameters)
       throws Exception;
   public String commit() throws Exception;
   public void rollback() throws Exception;

   public void setConnection(java.sql.Connection con);
}
```

※ 참고사항 : 사용자 데이터 액션 기능 구현 시 함수를 호출하는 순서는 다음과 같습니다.

1. setConnection()
2. init()
3. insert(), update(), delete()
4. commit(), rollback()
5. close()

RDB Store DataAction 구현

DataAction 인터페이스 예제

다음은 사용자 정의 DataAction 인터페이스를 구현한 예제입니다.

```
package com.forcs.sample.dataaction.rdb;

import com.forcs.log4oz.OZLog;

public class DelegateSample
   implements oz.udd.IRDBDelegate, oz.udd.ILoggerRef {

   private OZLog cat;
   private java.sql.Connection con;

   public DelegateSample() {
   }

   public void init(String param) throws Exception {
      //초기화 스크립트를 입력합니다.
   }

   public void close() {
      //close하는 스크립트를 입력합니다.
   }

   public String insert(oz.framework.dac.OZDACItem dac,
                        java.util.Dictionary parameters) throws Exception {
      //insert 하는 스크립트를 입력합니다.
      return null;
    }

   public String update(oz.framework.dac.OZDACItem dac,
                        java.util.Dictionary parameters) throws Exception {
      //Update하는 스크립트를 입력합니다.
      return null;
   }

   public String delete(oz.framework.dac.OZDACItem dac,
                        java.util.Dictionary parameters) throws Exception {
      //Delete하는 스크립트를 입력합니다.
      return null;
   }
```

```
    public String commit() throws Exception {
        //commit하는 스크립트를 입력합니다.
        return null;
    }

    public void rollback() throws Exception {
        //rollback하는 스크립트를 입력합니다.
    }

    public void setConnection(java.sql.Connection con) {
        this.con = con;
    }

    public void setLogger(OZLog cat) {
        this.cat = cat;
    }
}
```

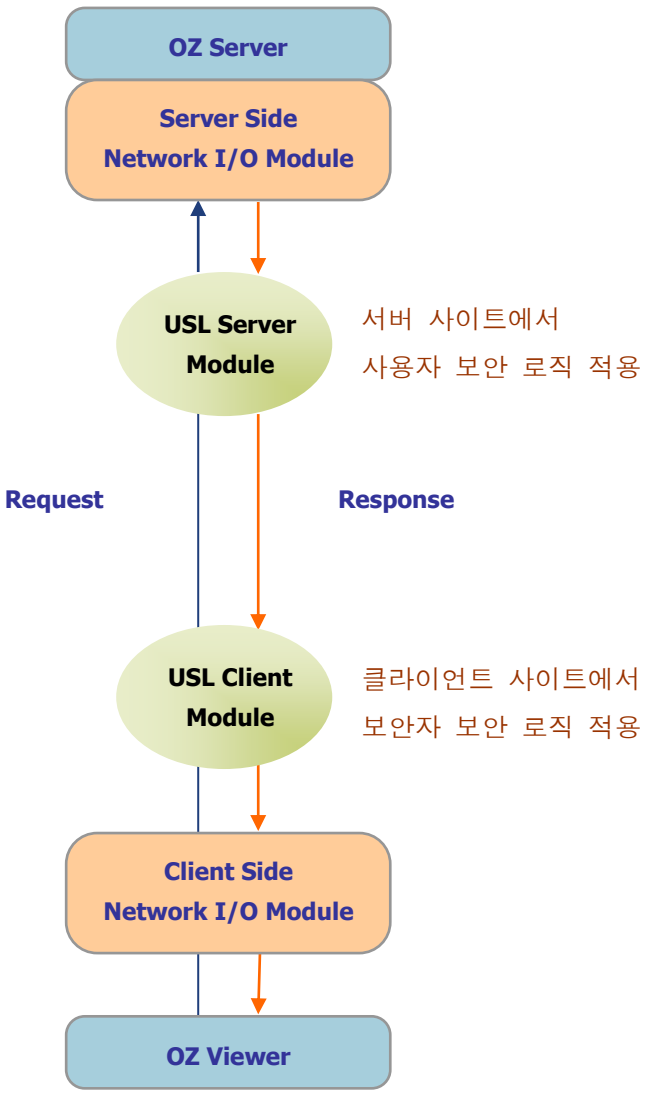
Ⅵ. User Security Logic

- USL 설정
- USL 구현

USL 설정

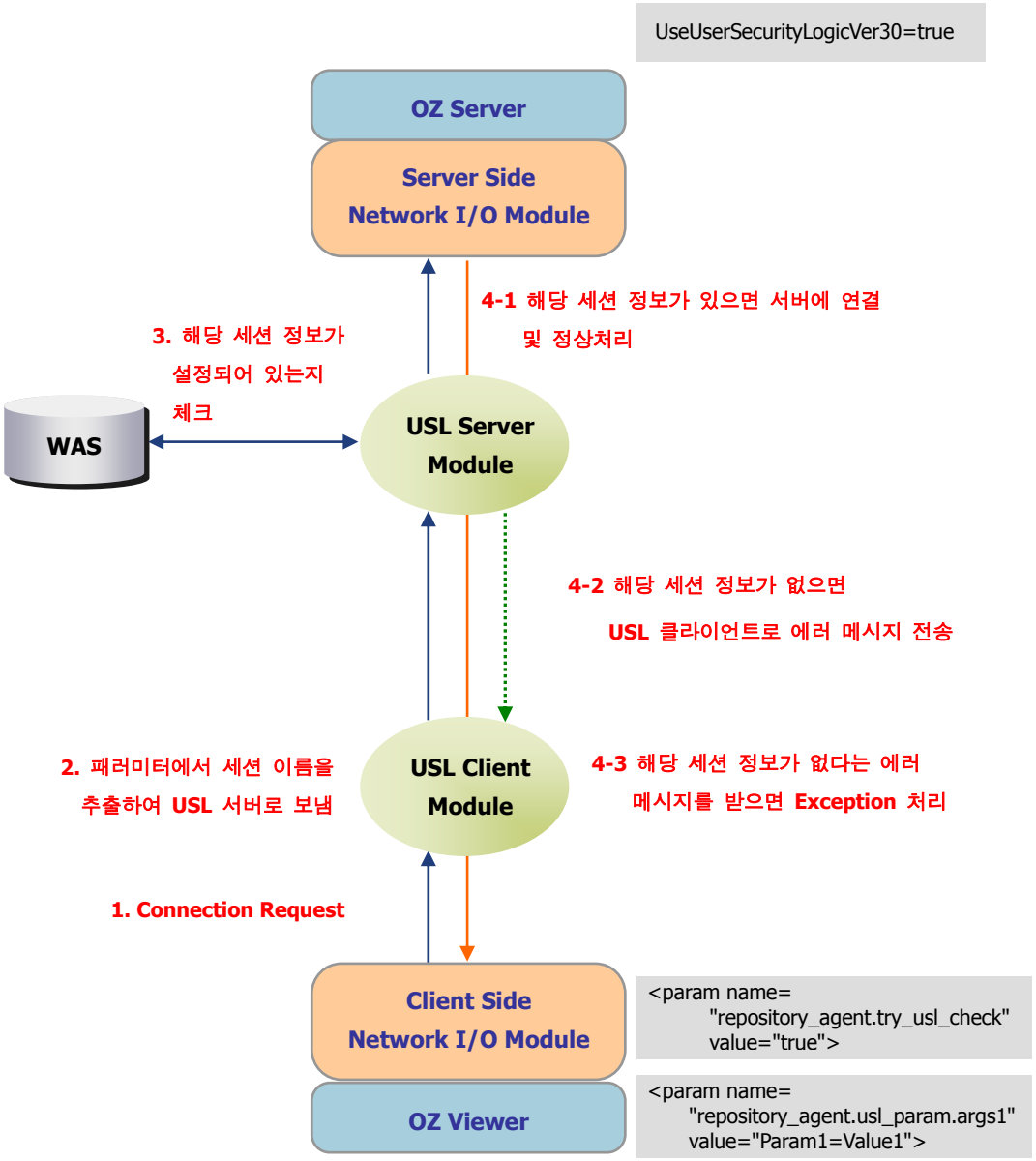
USL(User Security Logic)은 오즈 서버와 오즈 뷰어간 전송되는 데이터에 대해 사용자 보안 로직을 적용할 수 있도록 하기 위한 표준 인터페이스입니다. USL은 USL Server와 USL Client 모듈로 구성되며, 오즈 서버와 오즈 뷰어 양쪽의 네트워크 I/O 모듈 앞단에서 실행됩니다.

USL Flow



USL은 사용자 환경에서 다양하게 활용될 수 있는데 일반적으로 Web 브라우저와 WAS간의 Session 또는 Cookie 체크(오즈 서버 서블릿 타입에서만 적용 가능), Network I/O Stream에 대한 암•복호화, PKI 솔루션과의 연동을 위해 사용됩니다.

Session 체크 활용 예(기본 Session 체크 USL 제공)



오즈 서버 설정

■ 관련 파일

- USL 관련 라이브러리 파일 : OZ_HOME/lib/ozsfw60.jar
- USL 관련 설정 파일 : OZ_HOME/conf/uslmngr.properties

■ 파일 설정값 변경

- ozsfw60.jar 파일이 CLASSPATH에 잡혀있는지 확인합니다.
- uslmngr.properties 파일을 열어 USL 사용 여부를 설정하고, 서버 및 클라이언트 모듈의 클래스명 또는 라이브러리명을 설정합니다.

```
#
# use user security logic apply
#
UseUserSecurityLogicVer30=true
#
# default USL(Server&Client) class name (with package name)
#
OZDefault_SERVER=oz.usl.USL서버모듈 클래스명
OZDefault_CLIENT=oz.usl.USL클라이언트모듈 라이브러리명
```

  - uslmngr.properties에 UseUserSecurityLogicVer30=true로 설정하여, USL을 사용함으로 설정합니다.
  - 기본 USL 서버 및 클라이언트 모듈을 합니다.
    - OZDefault_SERVER= oz.usl.USL서버모듈 클래스명
    - OZDefault_CLIENT= oz.usl.USL클라이언트모듈 라이브러리명

    ex)

    OZDefault_SERVER=oz.usl.OZUSLServerSession

    OZDefault_CLIENT=oz.usl.OZUSLClientSession

OZ Report Viewer 설정

■ **USL 사용 여부 설정 뷰어 패러미터**

보고서 파일(OZR)를 호출할 때 USL 사용 여부를 설정하는 리포트 뷰어 패러미터인 "repository_agent.try_usl_check"를 "true"로 설정하여야 합니다.

```
<param name="repository_agent.try_usl_check" value="true">
```

※ 주의사항 : 만일 오즈 서버에서 "UseUserSecurityLogicVer30" 옵션을 "true"로 설정

한 경우에는 "repository_agent.try_usl_check"를 반드시 "true"로 설정하여야 합니다.

■ **USL 패러미터 설정 뷰어 패러미터**

USL 패러미터값을 설정할 경우에는 "repository_agent.usl_param.pcount"에 USL 패러미터 개수를 설정하고, "repository_agent.usl_param.args#"를 이용하여 패러미터명과 패러미터값을 설정합니다.

※ 주의사항 : 만일 오즈 서버에서 "UseUserSecurityLogicVer30" 옵션을 "true"로 설정하고, "repository_agent.try_usl_check"을 "true"로 설정한 상태에서 USL 패러미터값을 설정하지 않으면 보고서의 기본 패러미터인 폼 패러미터가 USL 패러미터로 적용됩니다.

■ **보고서를 호출하는 HTML 파일의 예**

```
<html>
<body>
<object id="OZReportViewer" width="100%" height="100%" classid="CLSID:64DA633F-E73B-4344-83BF-48483346CD53">
    <param name="connection.servlet" value="http://127.0.0.1:8080/oz60/server">
    <param name="viewer.configmode" value="html">

    <param name="repository_agent.try_usl_check" value="true">
    <param name="repository_agent.usl_param.pcount" value="2">
    <param name="repository_agent.usl_param.args1" value="Param1=Value1">
    <param name="repository_agent.usl_param.args2" value="Param2=Value2">

    <param name="connection.reportname" value="sample.ozr">
</object>
</body>
</html>
```

OZ Application Viewer 설정

■ **USL 사용 여부 설정 뷰어 패러미터**

애플리케이션 폼 파일(OZA)을 호출할 때 USL 사용 여부를 설정하는 애플리케이션 뷰어 패러미터인 "repository_agent.try_usl_check"를 "true"로 설정하여야 합니다.

※ 주의사항 : 만일 오즈 서버에서 "UseUserSecurityLogicVer30" 옵션을 "true"로 설정한 경우에는 "repository_agent.try_usl_check"를 반드시 "true"로 설정

하여야 합니다.

■ **USL 패러미터 설정 뷰어 패러미터**

USL 패러미터값을 설정할 경우에는 "repository_agent.usl_param.pcount"에 USL 패러미터 개수를 설정하고, "repository_agent.usl_param.args#"를 이용하여 패러미터명과 패러미터값을 설정합니다.

■ 애플리케이션 폼 파일을 호출하는 **HTML** 파일의 예

```
<html>
<body>
<object id="OZApplicationViewer" width="100%" height="100%" classid="CLSID:907A00F3-7390-4EF2-931B-360546587804">
    <param name="repository_agent.type" value="FROM_OZSERVER">
    <param name="repository_agent.ozserver.servlet"
            value="http://127.0.0.1:8080/oz60/server">

    <param name="repository_agent.try_usl_check" value="true">
    <param name="repository_agent.usl_param.pcount" value="2">
    <param name="repository_agent.usl_param.args1" value="Param1=Value1">
    <param name="repository_agent.usl_param.args2" value="Param2=Value2">

    <param name="ozadoc.path" value="sample.oza">
</object>
</body>
</html>
```

USL 구현

USL 서버 구성

서버쪽 USL 모듈은 클라이언트쪽 USL 모듈에서 보낸 Request Stream에 대한 처리와 반대로 클라이언트로 보낼 Response Stream에 대한 처리를 담당합니다.
구현할 리스트는 다음과 같습니다.

- **USL 스트림 관리 인터페이스**

  oz.cp.OZUSLServer 추상 클래스를 상속 받아 createSecureOutputStream(), createSecureInputStream() 메소드를 구현합니다.
  클라이언트에서 온 Request Stream을 그대로 오즈 서버에 보낼 경우에는 createSecureInputStream() 메소드를 구현하지 않아도 됩니다. 마찬가지로 서버에서 받은 Response Stream을 클라이언트에 그대로 보낼 경우 createSecureOutputStream() 메소드를 구현하지 않아도 됩니다.

- **InputStream 구현**

  클라이언트로부터 받은 데이터에 어떤 처리(복호화 처리 등)를 하여 오즈 서버로 넘겨야 할 경우 InputStream 클래스를 구현해야 합니다.
  이 클래스는 java.io.InputStream 클래스를 상속받아 read() 명령을 구현합니다.
  오즈 서버에서 USL 서버로 DataInputStream을 넘겨주므로 read()가 호출될때마다 DataInputStream으로부터 데이터를 읽어서 복호화 처리 등을 한 뒤 오즈 서버로 넘겨주면 됩니다. 처음부터 생성자의 스트림으로부터 모든 데이터를 한꺼번에 가져온 후 복호화 등의 처리를 하고 read가 호출될 때마다 하나씩 반환하는 식으로 구현해도 됩니다.

- **OutputStream 구현**

  클라이언트로 보낼 데이터에 어떤 처리(암호화 처리 등)를 하고자 할 경우 OutputStream 클래스를 구현해야 합니다.
  이 클래스는 java.io.OutputStream 클래스를 상속받아 write(), flush() 명령을 구현합니다.
  오즈 서버에서 USL 서버 모듈로 DataOutputStream을 넘겨주므로 write로 들어오는 데이터를 암호화 등의 처리를 하면서 오즈 서버에서 넘겨받은 원 DataOutputStream에 써 넣어 줍니다.
  또는 write()되는 데이터를 쌓아 놓았다가 flush될 때 한꺼번에 암호화 등의 처리를 하여

원 DataOutputStream으로 보내도 됩니다. 오즈 서버는 flush를 마지막에 한번만 호출하는 것을 보장합니다.

- **사용자 및 아이템에 대한 인증 구현**
  오즈 서버에서 사용자 별로 SSO 인증을 하거나 리파지토리 아이템에 대한 인가 처리를 하려면 check 함수를 이용하여 구현합니다.

USL 클라이언트 구성

클라이언트쪽 USL 모듈은 서버로 보낼 Request Stream에 대한 처리와 반대로 서버쪽 USL 모듈에서 보낸 Response Stream에 대한 처리를 담당합니다. 구현할 리스트는 다음과 같습니다.

- **USL 스트림 관리 인터페이스**
  OZ USL DLL프로젝트에 오즈에서 제공하는 기본 USL 소스가 제공됩니다. 서버에서 데이터를 주고 받는 스트림은 내부적인 스트림 클래스를 상속받고 이는 createSecureOutputStream(), createSecureInputStream() 메소드를 구현합니다.
  오즈 뷰어에서 온 Request Stream을 그대로 오즈 서버에 보낼 경우 createSecureOutputStream()을 구현하지 않아도 됩니다.
  마찬가지로 서버에서 온 Response Stream을 그대로 오즈 뷰어로 보낼 경우 createSecureInputStream()을 구현하지 않아도 됩니다.
  USL 클라이언트 모듈에서는 뷰어의 패러미터를 읽어서 서버에 넘겨주기도 하는데 이때 필요한 값을 서버에 넘기기 위해 createSecureOutputStream() 메소드에서 OutputStream 클래스의 writeUTF() 또는 writeINT() 등을 이용하기도 합니다.

- **InputStream 구현**
  서버로부터 받은 데이터에 어떤 처리(복호화 처리 등)를 하여 오즈 뷰어로 넘겨야 할 경우 CJDataInputStream클래스를 상속받아 구현해야 합니다.
  이 클래스는 기본적으로 오즈가 사용하는 기본적인 스트림의 메소드를 구현합니다.
  오즈 뷰어에서 USL 클라이언트로 CJDataInputStream을 넘겨주므로 read()가 호출될 때마다 CJDataInputStream으로부터 데이터를 읽어서 복호화 처리 등을 한 뒤 오즈 뷰어로 넘겨주면 됩니다.
  처음부터 생성자의 스트림으로부터 모든 데이터를 한꺼번에 가져온 후 복호화 등의 처리를 하고 read가 호출될 때마다 하나씩 반환하는 식으로 구현해도 됩니다.

■ **OutputStream 구현**

서버로 보낼 데이터에 어떤 처리(암호화 처리 등)를 하고자 할 경우 CJDataOutputStream 클래스를 구현해야 합니다.

이 클래스는 CJDataOutputStream 클래스를 상속받아 서버로 보낼 필요한 데이터를 각각의 구현 메소드를 통하여 전송하게 됩니다.

오즈 뷰어에서 USL 클라이언트 모듈로 CJDataOutputStream을 넘겨주며 해당 스트림을 사용하는 각각의 구현 메소드로 들어오는 데이터를 암호화 등의 처리를 하면서 오즈 뷰어에서 넘겨받은 원 CJDataOutputStream에 써 넣어 줍니다.

또는 write되는 데이터를 flush될 때 한꺼번에 암호화 등의 처리를 하여 원 CJDataOutputStream에 보내도 됩니다.

오즈 뷰어는 flush를 마지막에 한 번만 호출하는 것을 보장합니다.

USL 예제 1 - 세션 체크

USL을 이용하여 웹 브라우저와 WAS간의 사용자 세션을 체크하여 보고서 또는 애플리케이션 폼 파일 요청이 Valid한 세션 안에서 이루어지고 있는지 체크할 수 있습니다.

세션 체크를 위해서는 서버쪽에서는 OZUSLServerSession.jar를 이용하고, 클라이언트쪽에서는 OZUSLClientSession.dll 파일을 이용합니다.

■ **서버 환경 설정**

서버의 USL 환경 설정 파일인 OZ_HOME/conf/uslmngr.properties 파일을 편집기로 열어 아래와 같이 수정합니다.

```
#
# use user security logic apply
#
UseUserSecurityLogicVer30=true

#
# default USL(Server&Client) class name (with package name)
#
OZDefault_SERVER=oz.usl.OZUSLServerSession
OZDefault_CLIENT=oz.usl.OZUSLClientSession
```

■ **서버쪽 USL 모듈 구현**

OZUSLServerSession 클래스는 추상 클래스인 OZUSLServer를 상속 받습니다.

OZUSLServerSession 클래스는 클라이언트로부터 리포트 또는 애플리케이션 폼 파일을 요

청 받고, WAS에 있는 OZ Servlet과 통신하여 세션을 체크합니다.
세션 체크 결과 세션이 없는 경우 사용자에게 오류 메시지를 보냅니다.

```
<<OZUSLServerSession.java>>

package oz.usl;


import java.io.*;
import java.util.Enumeration;
import javax.servlet.http.HttpSession;
import oz.cp.message.OzcmException;
import oz.framework.cp.io.OZDataInputStream;
import oz.framework.cp.io.OZDataOutputStream;
import org.apache.log4j.*;

public class OZUSLServerSession extends OZUSLServer
{
   ////////////////////////////////////////////////////////////////////
   // Non-custom section. Leave below as is.
   static            protected            Category            cat            =
Category.getInstance(oz.version.OZSERVERFRAMEWORKVER.getLogVer()+".server");//C
ategory.getInstance("oz30.server");

   public OZUSLServerSession(){
      super();
   }

   ////////////////////////////////////////////////////////////////////
   // custom section. add your codes & modify

   // invoked after set httpSession & clientIP
   public void initialize(){

   }

   // invoked once per a transaction when prepare to read from client
   public InputStream createSecureInputStream(DataInputStream raw_in)
      throws OzcmException
   {
      cat.debug("OZUSLServerSession::CreateSecureInputStream");
      try{
         params_from_client.read(new OZDataInputStream(raw_in));
         for(int i=params_from_client.getLength()-1; i>=0; i--){
            cat.debug("OZUSLServerSession::CreateSecureInputStream: "
                    +     params_from_client.getKey(i)     +     ",     "     +
params_from_client.getValue(i));
```

```
         }
         //String class_nam  = raw_in.read

      }catch(Exception e){
         cat.error("OZUSLServerSession: read params from client:" , e);
         throw new OzcmException(e.getMessage());
      }
      return null;
   }

   // invoked once per a transaction when prepare to write to client
   public OutputStream createSecureOutputStream(DataOutputStream raw_out)
      throws OzcmException
   {
      cat.debug("OZUSLServerSession::CreateSecureOutputStream");
      try{
         if(this.http_request != null){
            HttpSession http_session = http_request.getSession(false);

            if(http_session == null){
               http_session = http_request.getSession(true);

               http_session.setAttribute("user_id",
params_from_client.get("user_id"));
               http_session.setAttribute("user_pw",
params_from_client.get("user_pw"));
            }

            if(http_session!=null){

              for(int i = params_from_client.getLength()-1; i>=0; i--){
                 http_session.setAttribute(params_from_client.getName(i),
params_from_client.getValue(i));
              }

              Enumeration e = http_session.getAttributeNames();
              for (; e.hasMoreElements(); ) {
                 String key = (String) e.nextElement();
                 String value = (String)http_session.getAttribute(key);
                 params_to_client.put(key, value);
                 cat.debug("OZUSLServerSession::http_session:          "+key+",
"+value);
              }
              params_to_client.write(new OZDataOutputStream(raw_out));
            }
         }
      }catch(Exception e){
         cat.error("OZUSLServerSession: write params to client:" , e);
```

```
            throw new OzcmException(e.getMessage());
        }
        return null;
    }

    public String check(String categoryName, String reportName) throws
OzcmException{
      String msg = "";
      oz.usl.SSOAdapter auth = null;
      try {
         auth                                                              =
(oz.usl.SSOAdapter)Class.forName("oz.usl.SSOAdapter").newInstance();
         msg = auth.check(this,categoryName,reportName);

       } catch (Exception e) {
         throw new OzcmException(e.getMessage());
       }
      return msg;
    }
}
```

※ 세션 관련 주의사항

- 세션 문제

  오즈 서버 Servlet 버전을 포팅하기 위해서는 WAS에서 /oz라는 애플리케이션을 만들어 오즈 서버(Servlet) 디렉토리를 매핑하게 되어 있는데 여기서 문제점은 루트나 특정 애플리케이션에서 설정한 세션 값이 다른 애플리케이션과 공유되지 않고 각각 따로 설정되는데 있습니다. 이는 애플리케이션 생성시 각 WEB-INF 폴더가 생성되고 세션 값은 WEB-INF\sessions 아래에 만들어지기 때문입니다.

  즉, http://localhost:8100/setcookie.jsp에서 설정한 값이 하위 폴더인 /http://localhost:8100/test/getcookie.jsp에서는 잘 나오는데 WAS에 웹 애플리케이션으로 등록된 /oz 즉, http://localhost:8100/oz/getcookie.jsp에서는 세션 값을 읽어올 수 없습니다. 마찬가지로 OZ 웹 애플리케이션에서 설정한 세션 값은 루트나 타 애플리케이션에서 읽어올 수 없습니다.

- 해결 방안

  오즈 서버를 WAS에 웹 애플리케이션으로 등록하지 않고 세션 값을 설정하는 사용자 웹 애플리케이션에 같이 포팅합니다.

  사용자 애플리케이션의 세션 값을 설정하는 Process에서 /oz 애플리케이션 아래에도 똑같은 세션을 설정하는 프로그램을 두고 똑같이 설정되도록 합니다.

  위의 문제는 Cookie를 사용할 경우에는 해당되지 않습니다.

■ **클라이언트쪽 USL 모듈 구현 – 클래스를 상속 받아 구현하는 방법**

OZUSLClientSession 클래스는 추상 클래스인 OZUSLClient를 상속 받습니다. createSecureInputStream 함수를 통해 생성된 InputStream에는 사용자 설정 내용이 포함됩니다.

```
<<OZUSLClientSession.cpp>>

#include <stdafx.h>
#include <oz/usl/OZUSLClient.h>
#include <oz/usl/OZUSLClientSession.h>
#include <common/io/PC1Encode128OutputStream.h>
#include <common/io/PC1Decode128InputStream.h>
#include <common/io/OZFuncInputStream.h>
#include <common/io/OZFuncOutputStream.h>
#include <oz/usl/OZUSLClientWrapper.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

OZUSLClientSession::OZUSLClientSession() : OZUSLClient()
{
}

OZUSLClientSession::~OZUSLClientSession()
{
}

CJInputStream*    OZUSLClientSession::createSecureInputStream(CJDataInputStream*
raw_in, CJOZAttributeList& params_from_server_to_fillup, bool bInit)
{
__super::createSecureInputStream(raw_in, params_from_server_to_fillup, bInit);
                              return NULL;
}

CJOutputStream                                                              *
OZUSLClientSession::createSecureOutputStream(CJDataOutputStream*        raw_out,
CJOZAttributeList& params_from_client_to_send)
{
 __super::createSecureOutputStream(raw_out, params_from_client_to_send);
                              return NULL;
}

void OZUSLClientSession::destory()
```

```
{
                              delete this;
}
/*
 * OZUSLClient_create: initialize User Secure Logic
 *
 * this function is called every time a trasaction with OZServer occured
 *
 * @param [in] tags           null terminated UTF8 OZParameter tag string array
to send to server
 * @param [in] values        null terminated UTF8 OZParameter value string array
to send to server
 * @param [in] env_raw       environment variable used in OZClient. just forget
about what is it.
 *                             however you must care about to pass env_raw to
rawRead(Write)Byte(BArray)
 *                        functions.
 *
 * @param [in] rawReadByte     function to get the encryptied raw byte from
server side USL to decrypt
 * @param [in] rawReadBArray  function to get the encryptied raw byte array from
server side USL to decrypt
 * @param [in] rawAvailable   function to get the tempory readable input buffer
size
 * @param [in] rawReadClose   function to close raw inputstream
 *
 * @param [in] rawWriteByte   function to get the encryptied raw byte to send to
server side USL
 * @param [in] rawWriteBArray function to get the encryptied raw byte array to
send to server side USL
 * @param [in] rawFlush      function to flush raw outputstream
 * @param [in] rawWriteClose  function to close raw outputstream
 *
 * @param [in] rawGetLastErrorMessage function to get error message from raw
in/out stream
 *
 *  @param  [out]  do_read_hook    whether  do  input  stream  wrappering  with
OZUSLClient_read or not
 *  @param  [out]  do_write_hook  whether  do  output  stream  wrappering  with
OZUSLClient_write or not
 * @param [out] env         the state variable if needs.
 *                        state variable is passed to all USL functions
 *                        to cover stateless function call interface.
 *
 * @return                 returns >= 0 if success returns returns < 0 if error.
 *                                  if got error return, OZClient will call
OZUSLClient_getLastErrorMessage
 *                        to get detailed error message.
```

```
 */


extern "C" __declspec(dllexport) int OZUSLClient_create
                                    (
                byte* params_bytes, int params_bytes_len,

                                void* env_raw,

   int (__cdecl *rawReadByte)(void* env_raw, byte* p_value, int *read_len),
int (__cdecl *rawReadBArray)(void* env_raw, byte* array_value, int len, int
*read_len),
   int (__cdecl *rawAvailable)(void* env_raw, int *available_len),
   int (__cdecl *rawReadClose)(void* env_raw, BOOL do_not_close_raw_in),

        int (__cdecl *rawWriteByte)(void* env_raw, byte value),
int (__cdecl *rawWriteBArray)(void* env_raw, byte* array_value, int len),
                int (__cdecl *rawFlush)(void* env_raw),
   int (__cdecl *rawWriteClose)(void* env_raw),

   byte* (__cdecl *rawGetLastErrorMessage)(void* env_raw),

   void** env
                                    )
{

                  TRACE(_T("OZUSLClient_create\n"));

                   OZUSLClientWrapper *uslw = NULL;

                   uslw = new OZUSLClientWrapper();
                 uslw->raw_in = new CJDataInputStream
                                          (new OZFuncInputStream(env_raw, uslw,
rawReadByte, rawReadBArray, rawAvailable, rawReadClose, rawGetLastErrorMessage),
TRUE);

                uslw->raw_out = new CJDataOutputStream
                                                  (new OZFuncOutputStream(env_raw,
rawWriteByte, rawWriteBArray, rawFlush, rawWriteClose, rawGetLastErrorMessage),
TRUE);

                uslw->ozusl = new OZUSLClientSession();

                        CJOZAttributeList attrs;
CJByteArrayInputStream bin((char*)params_bytes, params_bytes_len, FALSE);
                 CJDataInputStream din(&bin, FALSE);

                            attrs.read(din);
```

```
        uslw->ozusl->setServerIP(attrs.get(_T("OZ_SERVER_IP")));
       uslw->ozusl->setServerURL(attrs.get(_T("OZ_SERVER_URL")));

                                *env = uslw;
    return OZUSLCLIENT_INTERFACE_VERSION; // all ok.
}
```

■ **클라이언트쪽 USL 모듈 구현 – 오즈 스크립트로 구현하는 방법**

오즈 스크립트(getUSLServerParam)를 이용하여 서버네서 설정된 USL패러미터 값을 가져올 수 있습니다.

다음은 보고서에서 서버에 설정된 USL패러미터 값을 가져오는 예제입니다.

- 리포트 디자이너를 구동한 후 [파일]-[새로 만들기] 메뉴를 선택하거나 툴바의 새로 만들기 아이콘()을 클릭하여 새 보고서를 생성합니다.

- 추가된 리포트 위에 [추가]-[컴포넌트]-[밴드]-[타이틀 밴드] 메뉴를 선택하거나 툴바의 타이틀 밴드 아이콘()을 클릭하여 타이틀 밴드 하나를 추가합니다.

- 타이틀 밴드 위에 [추가]-[컴포넌트]-[라벨]-[라벨] 메뉴를 선택하거나 툴바의 라벨 아이콘()을 클릭하여 라벨을 하나 추가합니다.

- 라벨을 오른쪽 마우스로 선택하면 나타나는 [스크립트 편집] 메뉴를 선택 한 후 아래 스크립트 창에 다음과 같은 스크립트를 추가합니다.

```
String str;
str = getUSLServerParam("OZUSL_SERVER_URL");
// name에 해당하는 USL패러미터값이 OZUSL_SERVER_URL 일 경우
setattr("caption",str);
write("str"); // 디버그인포(뷰어에서 CTRL+Z)에서 확인할 경우
```

- 메뉴바에서 [파일] 메뉴의 [저장] 메뉴나 툴바에서 저장 아이콘()을 클릭하여 리파지토리 루트 폴더에 'getUSLServerParam.ozr' 파일로 저장합니다. 본 매뉴얼에서 리파지토리 루트 폴더는 'D:\Repository'입니다.

- 아래와 같은 html을 만들어 실행합니다.

```
<HTML>
<BODY>
<OBJECT    id    =    "ozviewer"    CLASSID="CLSID:0DEF32F8-170F-46f8-B1FF-
4BF7443F5F25" width="100%" height="100%">
```

```
<param name="connection.servlet" value="http://127.0.0.1:8080/oz60/server">
<param name="connection.reportname" value="/getUSLServerParam.ozr">
<param name="connection.usingusl" value="true">
<param name="viewer.isframe" value="true">
<param name="information.debug" value="true">

</OBJECT>
</body>
</HTML>
```

※ connection.usingusl의 값은 반드시 true로 설정해야합니다.

- 실행된 보고서에서 라벨의 caption 값으로 설정한 USL패러미터 값이 표시되는지 확인합니다.

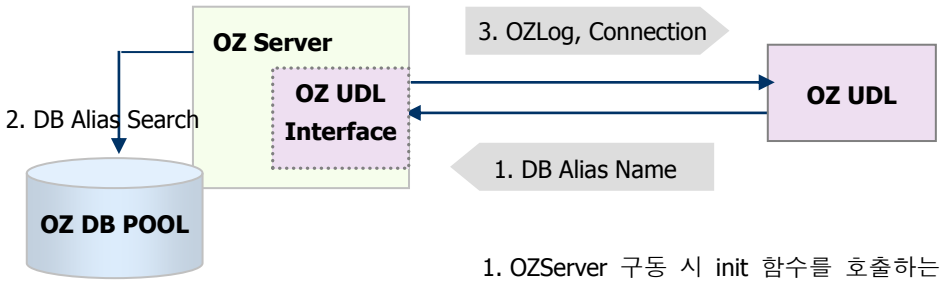
Ⅶ. User Defined Log

- 오즈 엔터프라이즈 서버에서 사용
- 오즈 스케줄러 서버에서 사용

오즈 엔터프라이즈 서버에서 사용

UDL(User Defined Log)은 오즈 서버를 통하여 데이터 액션을 수행하거나 데이터를 조회할 경우 데이터 베이스나 보고서 관련 정보를 임의의 파일이나 DB에 사용자가 정의한 타입의 형식으로 기록할 수 있도록 하기 위한 인터페이스입니다.

UDL 인터페이스 호출 구조



1. OZServer 구동 시 init 함수를 호출하는 경우
2. 서비스 호출하여 traceLog하는 경우

OZUserDefinedLogger 클래스

OZ UDL을 사용하기 위하여 oz.udl.OZUserDefinedLogger를 먼저 구현하여야 하며, oz.udl.OZUserDefinedLogger 구현 시 init, tracelog, getOZDBAlias 메소드를 반드시 구현하여야 합니다.

■ **init**

Prototype	`public void init(oz.framework.db.OZConnection ozconn, com.forcs.log4oz.OZLog cat) throws OZUserDefinedLogException`
Definition	오즈 서버 구동 시 사용자 정의 로그를 남길 대상을 정의합니다.

Argument	*ozconn*	DB에 사용자 정의 로그를 기록할 경우 사용할 DB 커넥션 정보 및 기록할 로그 형식
	cat	파일에 사용자 정의 로그를 기록할 경우 사용할 파일 정보 및 기록할 로그 형식

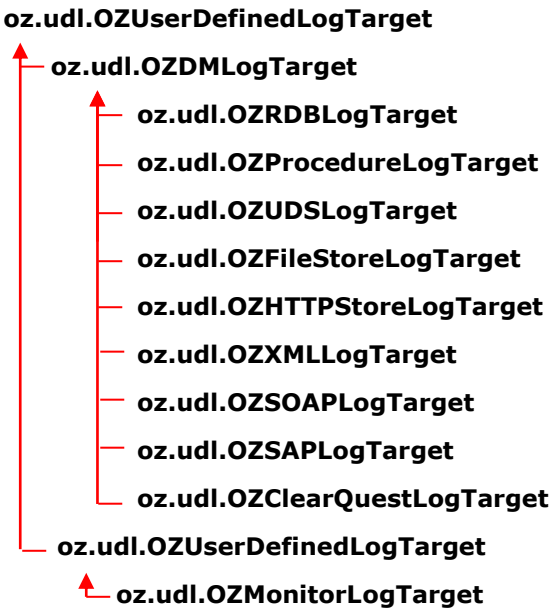
■ traceLog

Prototype	`public void traceLog(oz.udl.OZUserDefinedLogTarget target, oz.framework.db.OZConnection ozconn, com.forcs.log4oz.OZLog cat) throws OZUserDefinedLogException`	
Definition	로그에 기록할 타겟 내용을 정의합니다.	
Argument	*target*	사용자 정의 로그 타겟 설정할 수 있는 사용자 정의 로그 타겟은 아래 OZUserDefinedLogTarget 인터페이스 부분을 참조하시기 바랍니다.
	ozconn	DB에 사용자 정의 로그를 기록할 경우 사용할 DB 커넥션 정보 및 기록할 로그 형식
	cat	파일에 사용자 정의 로그를 기록할 경우 사용할 파일 정보 및 기록할 로그 형식

■ getOZDBAlias

Prototype	`public String getOZDBAlias()`
Definition	DB에 사용자 정의 로그를 기록할 경우 오즈 서버에 정의된 커넥션 풀의 앨리어스 명을 리턴합니다.

OZUserDefinedLogTarget 인터페이스의 클래스 구조



■ **OZUserDefinedLogTarget 클래스 함수**

OZ UDL에서 제공되는 정보를 요청하여 사용하기 위한 인터페이스입니다.

- getIPAddress

Prototype	`public String getIPAddress()`
Definition	클라이언트가 요청한 오즈 서버 IP 주소를 가져옵니다.

- getHttpRequest

Prototype	`public HttpServletRequest getHttpRequest()`
Definition	오즈 서버가 서블릿 타입일 경우 요청한 HttpRequest를 가져옵니다. ※ 참고사항 : 오즈 서버가 데몬 타입을 경우 null을 리턴합니다.

- getHttpServlet

Prototype	`public HttpServlet getHttpServlet()`
Definition	오즈 서버가 서블릿 타입일 경우 요청한 HttpServlet을 가져옵니다. ※ 참고사항 : 오즈 서버가 데몬 타입을 경우 null을 리턴합니다.

- getUserID

Prototype	`public String getUserID()`

Definition	클라이언트에서 쿼리문을 실행한 오즈 서버 사용자 ID를 가져옵니다. ※ 제약사항 : 오즈 웹 서비스를 통하여 사용자 정의 모니터 로그를 사용할 경우 UserID 값이 공백(" ")으로 리턴됩니다.

■ **OZDMLogTarget 클래스 함수**

OZ UDL에서 데이터 모듈 관련 정보를 사용하기 위한 인터페이스입니다.

- getODIName

Prototype	`public String getODIName()`
Definition	쿼리문을 실행한 ODI 이름을 가져옵니다.

- getDataSetName

Prototype	`public String getDataSetName()`
Definition	쿼리문을 실행한 데이터 셋 이름을 가져옵니다.

- getDataStoreName

Prototype	`public String getDataStoreName()`
Definition	쿼리문을 실행한 데이터 스토어 이름을 가져옵니다.

- getOZParam

Prototype	`public HashMap getOZParam()`
Definition	오즈 사용자 지정 패러미터를 HashMap 형태(key와 value)로 가져옵니다.

■ **OZRDBLogTarget 클래스 함수**

OZ UDL에서 RDB 스토어 관련 정보를 사용하기 위한 인터페이스로 쿼리문이 실행된 후 로그가 기록됩니다.

- getDBAlias

Prototype	`public String getDBAlias()`
Definition	실행한 ODI에서 사용한 커넥션 풀의 DB 앨리어스 명을 가져옵니다. ※ 참고사항 : 데이터 스토어에서 앨리어스를 사용하지 않는 경우 커넥션 풀의 키 값을 가져옵니다.

- getDBExecuteQuery

Prototype	`public String getDBExecuteQuery()`
Definition	클라이언트에서 요청한 쿼리문을 가져옵니다.

- getPreparedQueryValue

Prototype	`public Vector getPreparedQueryValue()`
Definition	컴파일된 질의문에 사용된 값(바이너리 데이터 포함)을 가져옵니다.

- getExecuteQueryState

Prototype	`public String getExecuteQueryState()`
Definition	클라이언트에서 요청한 쿼리문이 정상적으로 실행되었는지 여부를 가져옵니다. 쿼리문이 정상적으로 실행된 경우 "true", 정상적으로 실행되지 않은 경우 "false:에러 메시지" 형태로 리턴됩니다.

- getQueryExecuteTime

Prototype	`public long getQueryExecuteTime()`
Definition	클라이언트에서 요청한 쿼리문이 수행되는 시간을 가져옵니다. (단위 : mesc)

- isPrepared

Prototype	`public boolean isPrepared()`
Definition	컴파일된 질의문의 사용 여부를 가져옵니다.

- getCRUDFlagName

Prototype	`public String getCRUDFlagName()`
Definition	클라이언트에서 요청하여 실행된 쿼리문의 타입을 가져옵니다. Select 문이 사용된 경우 "select"로, Insert 문이 사용된 경우 "insert"로, Update 문이 사용된 경우 "update"로, Delete 문이 사용된 경우 "delete"로 리턴됩니다.

■ **OZProcedureLogTarget 클래스 함수**

OZ UDL에서 프로시저 데이터 관련 정보를 사용하기 위한 인터페이스로 프로시저가 실행된 후 로그가 기록됩니다.

- getDBAlias

Prototype	`public String getDBAlias()`
Definition	실행한 ODI에서 사용한 커넥션 풀의 DB 앨리어스 명을 가져옵니다. ※ 참고사항 : 데이터 스토어에서 앨리어스를 사용하지 않는 경우 커넥션 풀의 키 값을 가져옵니다.

- getProcedureName

Prototype	`public String getProcedureName()`
Definition	클라이언트에서 요청한 프로시저 이름을 가져옵니다.

- getParameterList

Prototype	`public ArrayList getParameterList()`
Definition	클라이언트에서 요청한 프로시저 패러미터를 배열로 가져옵니다.

- getProcedureExecuteTime

Prototype	`public long getProcedureExecuteTime()`
Definition	클라이언트에서 요청한 프로시저가 수행되는 시간을 가져옵니다. (단위 : mesc)

- getProcedureState

Prototype	`public String getProcedureState()`
Definition	클라이언트에서 요청한 프로시저가 정상적으로 실행되었는지 여부를 가져옵니다. 프로시저가 정상적으로 실행된 경우 "true", 정상적으로 실행되지 않은 경우 "false:에러 메시지" 형태로 리턴됩니다.

■ **OZUDSLogTarget 클래스 함수**

OZ UDL에서 사용자 데이터 스토어 정보를 사용하기 위한 인터페이스로 사용자 데이터 스토어 실행문이 수행된 후 로그가 기록됩니다.

- getExecuteCommand

Prototype	`public String getExecuteCommand()`
Definition	클라이언트에서 요청한 실행문을 가져옵니다.

- getCRUDFlagName

Prototype	`public String getCRUDFlagName()`
Definition	클라이언트에서 요청하여 실행된 쿼리문의 타입을 가져옵니다. Select 문이 실행된 경우 "select"로, Insert 문이 실행된 경우 "insert"로, Update 문이 실행된 경우 "update"로, Delete 문이 실행된 경우 "delete"로 리턴됩니다.

- getCommandExecuteTime

Prototype	`public long getCommandExecuteTime()`
Definition	클라이언트에서 요청한 실행문이 수행되는 시간을 가져옵니다. (단위 : mesc)

- getCommandState

Prototype	`public String getCommandState()`
Definition	클라이언트에서 요청한 실행문이 정상적으로 실행되었는지 여부를 가져옵니다. 실행문이 정상적으로 실행된 경우 "true", 정상적으로 실행되지 않은 경우 "false:에러 메시지" 형태로 리턴됩니다.

■ **OZFileStoreLogTarget 클래스 함수**

OZ UDL에서 파일 스토어 정보를 사용하기 위한 인터페이스로 파일 스토어의 파일을 읽어온 뒤 로그가 기록됩니다.

- getFilePath

Prototype	`public String getFilePath()`
Definition	클라이언트에서 요청한 파일 경로를 가져옵니다.

- getFileAccessTime

Prototype	`public long getFileAccessTime()`
Definition	클라이언트에서 요청한 파일을 가져오는 시간을 가져옵니다. (단위 : mesc)

- getFileAccessState

Prototype	`public String getFileAccessState()`
Definition	클라이언트에서 요청한 파일이 정상적으로 실행되었는지 여부를 가져옵니다. 파일이 정상적으로 실행된 경우 "true", 정상적으로 실행되지 않은 경우 "false:에러 메시지" 형태로 리턴됩니다.

■ **OZHTTPStoreLogTarget 클래스 함수**

OZ UDL에서 HTTP 스토어 정보를 사용하기 위한 인터페이스로 HTTP 스토어의 파일을 읽어온 뒤 로그가 기록됩니다.

- getURL

Prototype	`public String getURL()`
Definition	클라이언트에서 요청한 URL을 가져옵니다.

- getHTTPAccessTime

Prototype	`public long getHTTPAccessTime()`
Definition	클라이언트에서 요청한 HTTP 경로의 파일을 가져오는 시간을 가져옵니다. (단위 : mesc)

- getHTTPAccessState

Prototype	`public String getHTTPAccessState()`
Definition	클라이언트에서 요청한 HTTP 경로의 파일이 정상적으로 실행되었는지 여부를 가져옵니다. 파일이 정상적으로 실행된 경우 "true", 정상적으로 실행되지 않은 경우 "false:에러 메시지" 형태로 리턴됩니다.

■ **OZXMLLogTarget 클래스 함수**

OZ UDL에서 XML 스토어 정보를 사용하기 위한 인터페이스로 XML 파일을 읽어온 뒤 로그가 기록됩니다.

- getURL

Prototype	`public String getURL()`
Definition	클라이언트에서 요청한 파일 경로 또는 URL을 가져옵니다.

- getURLAccessTime

Prototype	`public long getURLAccessTime()`
Definition	클라이언트에서 요청한 URL의 파일을 가져오는 시간을 가져옵니다.

- getURLAccessState

Prototype	`public String getURLAccessState()`
Definition	클라이언트에서 요청한 URL의 파일이 정상적으로 실행되었는지 여부를 가져옵니다. 파일이 정상적으로 실행된 경우 "true", 정상적으로 실행되지 않은 경우 "false:에러 메시지" 형태로 리턴됩니다.

■ **OZSOAPLogTarget 클래스 함수**

OZ UDL에서 SOAP 스토어 정보를 사용하기 위한 인터페이스로 Service Response 후에 로그가 기록됩니다.

- getServiceName

Prototype	`public String getServiceName()`
Definition	클라이언트에서 요청한 서비스 이름을 가져옵니다.

- getPort

Prototype	`public String getPort()`
Definition	클라이언트에서 요청한 서비스의 포트를 가져옵니다.

- getOperation

Prototype	`public String getOperation()`
Definition	클라이언트에서 요청한 서비스의 오퍼레이션을 가져옵니다.

- getEndPoint

Prototype	`public String getEndPoint()`
Definition	클라이언트에서 요청한 서비스를 호출한 EndPoint를 가져옵니다.

- getRequestXML

Prototype	`public String getRequestXML()`
Definition	클라이언트에서 요청한 XML 내용을 가져옵니다.

- getServiceExecuteTime

Prototype	`public long getServiceExecuteTime()`
Definition	클라이언트에서 요청한 서비스가 실행되는 시간을 가져옵니다. (단위 : mesc)

- getServiceState

Prototype	`public String getServiceState()`
Definition	클라이언트에서 요청한 SOAP의 서비스가 정상적으로 실행되었는지 여부를 가져옵니다. 서비스가 정상적으로 실행된 경우 "true", 정상적으로 실행되지 않은 경우 "false:에러 메시지" 형태로 리턴됩니다.

■ OZSAPLogTarget 클래스 함수

OZ UDL에서 SAP 스토어 정보를 사용하기 위한 인터페이스로 함수가 실행된 후에 로그가 기록됩니다.

- getFunctionName

Prototype	`public String getFunctionName()`
Definition	클라이언트에서 요청한 함수 이름을 가져옵니다.

- getFunctionType

Prototype	`public String getFunctionType()`
Definition	클라이언트에서 요청한 함수 타입을 가져옵니다.

- getInputParameters

Prototype	`public HashMap getInputParameters()`
Definition	클라이언트에서 요청한 입력 패러미터를 HashMap 형태(key와 value)로 가져옵니다.

- getResultSetTypes

Prototype	`public String getResultSetTypes()`
Definition	클라이언트에서 요청한 ResultSet 타입을 가져옵니다. Structure 타입인 경우 "Structure"로, Table 타입인 경우 "Table"로, SimpleFields 타입인 경우 "SimpleFields"로 리턴됩니다.

- getFunctionExecuteTime

Prototype	`public long getFunctionExecuteTime()`
Definition	클라이언트에서 요청한 함수가 실행되는 시간을 가져옵니다. (단위 : mesc)

- getFunctionState

Prototype	`public String getFunctionState()`
Definition	클라이언트에서 요청한 함수가 정상적으로 실행되었는지 여부를 가져옵니다. 함수가 정상적으로 실행된 경우 "true", 정상적으로 실행되지 않은 경우 "false:에러 메시지" 형태로 리턴됩니다.

■ **OZClearQuestLogTarget 클래스 함수**

OZ UDL에서 Clear Quest 스토어 정보를 사용하기 위한 인터페이스로 Clear Quest 쿼리문이 실행된 후에 로그가 기록됩니다.

- getExecuteType

Prototype	`public String getExecuteType()`
Definition	클라이언트에서 요청한 Clear Quest 실행 타입을 가져옵니다. SQL 타입인 경우 "ExecuteSQL"로, 쿼리 타입인 경우 "ExecuteQuery"로, 다이나믹 쿼리 타입인 경우 "ExecuteDynamicQuery"로 리턴됩니다.

- getQuerySubType

Prototype	`public String getQuerySubType()`
Definition	클라이언트에서 요청한 QuerySub 타입을 가져옵니다. Query 타입인 경우 "Query"로, Chart 타입인 경우 "Chart"로 리턴됩니다.

- getQuery

Prototype	`public String getQuery()`
Definition	클라이언트에서 요청한 쿼리문을 가져옵니다.

- getQueryExecuteTime

Prototype	`public long getQueryExecuteTime()`
Definition	클라이언트에서 요청한 쿼리문이 실행되는 시간을 가져옵니다. (단위 : mesc)

- getQueryState

Prototype	`public String getQueryState()`
Definition	클라이언트에서 요청한 쿼리문이 정상적으로 실행되었는지 여부를 가져옵니다. 쿼리문이 정상적으로 실행된 경우 "true", 정상적으로 실행되지 않은 경우 "false:에러 메시지" 형태로 리턴됩니다.

■ **OZMonitorLogTarget 클래스 함수**

OZ UDL에서 모니터 로그 관련 정보를 사용하기 위한인터페이스입니다.

- getMark

Prototype	`public String getMark()`
Definition	서비스 시작 및 종료 표시를 가져옵니다. 서비스 시작인 경우 "start"로, 서비스 종료인 경우 "end"로 리턴됩니다.

- getThreadName

Prototype	`public String getThreadName()`
Definition	서비스 스레드 이름을 가져옵니다.

- getServiceTime

Prototype	`public long getServiceTime()`
Definition	서비스 시작 시간과 종료 시간을 가져옵니다. (단위 : mesc) ※ 참고사항 : 1970년 1월 1일부터 현재 시간을 Millisecond로 표시합니다.

- getFreeMemory

Prototype	`public long getFreeMemory()`
Definition	사용 가능한 메모리 크기를 가져옵니다. (단위 : byte)

- getTotalMemory

Prototype	`public long getTotalMemory()`
Definition	전체 JVM 메모리 크기를 가져옵니다. (단위 : byte)

- getServiceCode

Prototype	`public integer getServiceCode()`
Definition	서비스 구분 코드를 가져옵니다. ※ 참고사항 ▪ getMark 함수의 리턴 값이 "end"일 경우에만 상태 코드가 리턴되며, "start"일 경우 -1을 리턴합니다. ▪ 자세한 서비스 구분 코드는 오즈 엔터프라이즈 서버 관리자 가이드의 "서비스 코드" 부분을 참조하시기 바랍니다.

- getServiceStatus

Prototype	`public integer getServiceStatus()`
Definition	서비스 상태 코드를 가져옵니다. 성공일 경우 9001, 실패일 경우 9002를 리턴합니다. ※ 참고사항 : getMark 함수의 리턴 값이 "end"일 경우에만 상태 코드가 리턴되며, "start"일 경우 -1을 리턴합니다.

- getServiceParameter

Prototype	`public String getServiceParameter()`
Definition	클라이언트에서 서비스 요청 시 전송되는 패러미터 값을 가져옵니다. ※ 참고사항 : getMark 함수의 리턴 값이 "end"일 경우에만 값이 리턴되며, "start"일 경우 -1을 리턴합니다.

- getDBSessionID

Prototype	`public String getDBSessionID()`
Definition	DBMS 세션 ID를 가져옵니다. ※ 참고사항 : getMark 함수의 리턴 값이 "end"일 경우에만 값이 리턴되며, "start"일 경우 -1을 리턴합니다.

- getExecuteTime

Prototype	`public String getExecuteTime()`
Definition	서비스 처리 시 소요된 시간을 가져옵니다. (단위 : ms) ※ 참고사항 : getMark 함수의 리턴 값이 "end"일 경우에만 값이 리턴되며 "start"일 경우 -1을 리턴합니다.

- getErrorCode

Prototype	`public String getErrorCode()`
Definition	에러 발생 시 에러 코드를 가져옵니다. ※ 참고사항 : 에러 코드 내용은 conf/server_error_msg_언어명_국가명.xml 파일을 참고하시기 바랍니다.

- getErrorMsg

Prototype	`public String getErrorMsg()`
Definition	에러 발생 시 에러 메시지를 가져옵니다

- getErrorStackTrace

Prototype	`public String getErrorStackTrace()`
Definition	에러 발생 시 Stack Trace를 가져옵니다.

- getDBConns

Prototype	`public String getDBConns()`
Definition	Alias별 DB 사용 개수를 가져옵니다.

UDL 구현 시 오즈 서버 설정 방법

■ 관련 파일

- UDL 관련 라이브러리 파일 : OZ_HOME/lib/ozudl.jar
- UDL 관련 설정 파일 : OZ_HOME/conf/ozudl.properties

■ 오즈 서버 실행 설정 파일 변경

- ozudl.jar 파일이 CLASSPATH에 잡혀있는지 확인합니다.
- ozudl.properties 파일을 열어 UDL 사용 여부를 설정하고 사용할 경우 구현한 클래스의 경로를 설정합니다.

```
#---------------------------------
```

```
# configuraion of OZ User Defined log
#--------------------------------
#

OZ_USER_DEFINED_LOG.Active=true
OZ_USER_DEFINED_LOG.Class=oz.udl.UDL클래스명

OZ_UDL_MONITOR.Active=true
OZ_UDL_MONITOR.Class=oz.udl.UDL클래스명
```

- ozudl.properties에 아래와 같이 설정하여 UDL을 사용함으로 설정합니다.
  - 서버 로그를 사용자 정의 로그로 기록할 경우

    OZ_USER_DEFINED_LOG.Active=true
  - 모니터 로그를 사용자 정의 로그로 기록할 경우

    OZ_UDL_MONITOR.Active= true

- 서버 로그와 모니터 로그의 UDL 클래스를 설정합니다.
  - OZ_USER_DEFINED_LOG.Class=oz.udl.UDL클래스명
  - OZ_UDL_MONITOR.Class=oz.udl.UDL클래스명

  ex)

```
OZ_USER_DEFINED_LOG.Class=oz.udl.file.OZUserDefinedLogger
OZ_UDL_MONITOR.Class=oz.udl.db.OZMonitorLogForDB
```

UDL 예제 1 - 서버 로그를 파일에 기록하는 방법

UDL을 이용하여 오즈 서버 로그를 임의의 파일에 사용자가 정의한 타입의 형식으로 기록하는 방법에 대해 설명합니다.

■ **사용자 정의 로그 구현**

UDL을 사용하여 사용자 로그 파일을 생성하는 클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 oz.udl.file.OZUserDefinedLoggerForFile 클래스를 생성한 후 해당 클래스를 "ozudl.jar" 파일로 만들었습니다.

```
package oz.udl.file;

import java.io.*;
import java.util.*;
import org.apache.log4j.*;
```

```
import javax.servlet.http.*;

import oz.udl.*;
import oz.framework.db.*;
import com.forcs.log4oz.OZLog;
import oz.framework.dac.OZDACData;
import oz.dm.meta.OZProcedureParameter;

public class OZUserDefinedLoggerForFile implements OZUserDefinedLogger{

   Category logger = null;

   private final static String ROLLING_FILE_APPENDER = "RollingAppender";
   private final static String DAILY_ROLLING_FILE_APPENDER =
"DailyRollingAppender";


   private String logpath;
   private String layoutS;
   private String AppenderType;
   private String filesize;
   private String backupindex;
   private String dailytype;

   public OZUserDefinedLoggerForFile() {

   }

   /**
    * 서버구동시 유저가 필요한 내용을 구현해야 할 메소드
    * @param ozconn OZConnection 서버에서 제공되는 Connection
    * @param cat OZLog 서버에 trace할 로그
    * @throws OZUserDefinedLogException
    */
   public void init(OZConnection ozconn, OZLog cat) throws
OZUserDefinedLogException {
      try {
         load(cat);
         setAppender(cat);

      }catch(Exception ex){
         if (logger != null) {
            logger.error(ex.getMessage(), ex);
            cat.error(ex.getMessage(), ex);
         }
         throw new OZUserDefinedLogException(ex.getMessage());
      }
   }
```

```
   private void load(OZLog cat) {
      Properties props = new Properties();
      FileInputStream fis = null;
      try {
         fis = new FileInputStream("udl_file.properties");
         props.load(fis);

         logpath=props.getProperty("logpath");
         layoutS=props.getProperty("layout");
         AppenderType=props.getProperty("AppenderType");
         filesize=props.getProperty("filesize");
         backupindex=props.getProperty("backupindex");
         dailytype=props.getProperty("dailytype");

      }catch(Exception ex){
         cat.error(ex.getMessage(), ex);
      }finally{
         if (fis != null) try { fis.close(); }catch(Exception ex1){}
      }

   }


   private void setAppender(OZLog cat) throws Exception {
      PatternLayout layout = new PatternLayout(layoutS);

      if (ROLLING_FILE_APPENDER.equals(AppenderType)) {
         RollingFileAppender   appender   =   new   RollingFileAppender(layout,
logpath, true);

         appender.setMaxFileSize(filesize);
         try {
            appender.setMaxBackupIndex(Integer.parseInt(backupindex));
         }catch(Exception ex){
            appender.setMaxBackupIndex(5);
         }

         logger = Category.getInstance("oz.udl.file");
         logger.setPriority(Priority.DEBUG);

         logger.addAppender(appender);

         logger.debug("start....");
         cat.debug("user db logger init..."); //오즈서버에 trace하는 로그
      }

      else if (DAILY_ROLLING_FILE_APPENDER.equals(AppenderType)) {
```

```
            String datePattern = "'.'"+dailytype;
            DailyRollingFileAppender appender = new DailyRollingFileAppender();

            appender.setDatePattern(datePattern);
            appender.setLayout(layout);
            appender.setFile(logpath,true);
        }
    }

    /**
     * 로그에 trace할 내용인 target을 가지고 유저가 구현해야 할 메소드
     * @param target OZUserDefinedLogTarget
     * @param ozconn OZConnection
     * @param cat OZLog (서버로그에 trace되는 param 예)cat.debug(str);)
     * @throws OZUserDBLogException
     */
    public void traceLog(OZUserDefinedLogTarget target, OZConnection ozconn,
OZLog cat) throws OZUserDefinedLogException {

        StringBuffer sb = new StringBuffer();
        try {

                if(target instanceof OZProcedureLogTarget) {
                    sb.append(procedureStore(target));
                }
                else if(target instanceof OZRDBLogTarget){
                    sb.append(rdbstroe(target));
                }
                else if(target instanceof OZUDSLogTarget) {
                    sb.append(udsStore(target));
                }
                else if(target instanceof OZFileStoreLogTarget) {
                    sb.append(filestore(target));
                }
                else if(target instanceof OZHTTPStoreLogTarget) {
                    sb.append(httpStore(target));
                }
                else if(target instanceof OZXMLLogTarget) {
                    sb.append(xmlStore(target));
                }
                else if(target instanceof OZSOAPLogTarget) {
                    sb.append(soapStore(target));
                }
                else if(target instanceof OZSAPLogTarget) {
                    sb.append(sapStore(target));
                }
                else if(target instanceof OZClearQuestLogTarget) {
```

```
                sb.append(clearQuestStore(target));
            }
            sb.append(commonfactor(target));

            logger.info(sb.toString());
         }catch(Exception ex){
            logger.error(ex.getMessage(), ex);
            cat.error(ex.getMessage(),ex);
            throw new OZUserDefinedLogException(ex.getMessage());
         }

   }

   public String rdbstroe(OZUserDefinedLogTarget target){
         OZRDBLogTarget ref = (OZRDBLogTarget)target;
         StringBuffer sb = new StringBuffer();
         String dbAlias = (ref.getDBAlias() == null)? "" : ref.getDBAlias();
         sb.append("[DB Alias=");
         sb.append(dbAlias);
         sb.append("]");

         String crudFlagName = (ref.getCRUDFlagName() == null) ? "" :
ref.getCRUDFlagName();
         sb.append("[CRUDType=");
         sb.append(crudFlagName);
         sb.append("]");
         sb.append("[isPrepared=");
         sb.append(ref.isPrepared());
         sb.append("]");
         if(ref.getPreparedQueryValue() != null){
            sb.append("[QueryPreparedValue=");
            Vector vec = ref.getPreparedQueryValue();
            if (crudFlagName.equals("select")) {
               for (int i = 0; i < vec.size(); i++) {
                  sb.append(vec.elementAt(i));
               }
            }else {
               for (int i = 0; i < vec.size(); i++) {
                  OZDACData data = (OZDACData)vec.elementAt(i);
                  sb.append(data.toStringData());
                  sb.append(";");
               }
            }
            sb.append("]");
         }
         String query = (ref.getDBExecuteQuery() == null) ? "" :
ref.getDBExecuteQuery();
         sb.append("[Query=");
```

```
            sb.append(query);
            sb.append("]");
            long queryExec = ref.getQueryExecuteTime();
            sb.append("[Query exec=");
            sb.append(queryExec);
            sb.append("]");
            String queryState = (ref.getExecuteQueryState() == null) ? "" :
ref.getExecuteQueryState();
            sb.append("[QueryState=");
            sb.append(queryState);
            sb.append("]");

            return sb.toString();
        }

        public String udsStore(OZUserDefinedLogTarget target){
            OZUDSLogTarget ref = (OZUDSLogTarget)target;
            StringBuffer sb = new StringBuffer();

            sb.append("[ExecuteCommand=");
            sb.append(ref.getExecuteCommand());
            sb.append("]");
            sb.append("[CRUDFlagName=");
            sb.append(ref.getCRUDFlagName());
            sb.append("]");
            sb.append("[CommandExecuteTime=");
            sb.append(ref.getCommandExecuteTime());
            sb.append("]");
            sb.append("[CommandState=");
            sb.append(ref.getCommandState());
            sb.append("]");

            return sb.toString();
        }

        public String procedureStore(OZUserDefinedLogTarget target){
            OZProcedureLogTarget ref = (OZProcedureLogTarget)target;
            StringBuffer sb = new StringBuffer();

            sb.append("[DBAlias=");
            sb.append(ref.getDBAlias());
            sb.append("]");
            sb.append("[ProcedureName=");
            sb.append(ref.getProcedureName());
            sb.append("]");
            sb.append("[ParameterList=");
            ArrayList list = ref.getParameterList();
            for(int i=0;i<list.size();i++){
```

```
            OZProcedureParameter param = (OZProcedureParameter)list.get(i);
            sb.append(param.getValue());
            sb.append(",");
            sb.append(param.getDataTypeName());
            sb.append(";");
        }
        sb.append(ref.getParameterList());
        sb.append("]");
        sb.append("[ProcedureExecuteTime=");
        sb.append(ref.getProcedureExecuteTime());
        sb.append("]");
        sb.append("[ProcedureState=");
        sb.append(ref.getProcedureState());
        sb.append("]");

        return sb.toString();
    }

    public String xmlStore(OZUserDefinedLogTarget target){
        StringBuffer sb = new StringBuffer();
        OZXMLLogTarget ref = (OZXMLLogTarget)target;

        sb.append("[URL=");
        sb.append(ref.getURL());
        sb.append("]");
        sb.append("[URLAccessTime=");
        sb.append(ref.getURLAccessTime());
        sb.append("]");
        sb.append("[URLAccessState=");
        sb.append(ref.getURLAccessState());
        sb.append("]");

        return sb.toString();
    }

    public String soapStore(OZUserDefinedLogTarget target){
        StringBuffer sb = new StringBuffer();
        OZSOAPLogTarget ref = (OZSOAPLogTarget)target;

        sb.append("[ServiceName=");
        sb.append(ref.getServiceName());
        sb.append("]");
        sb.append("[Port=");
        sb.append(ref.getPort());
        sb.append("]");
        sb.append("[EndPoint=");
        sb.append(ref.getEndPoint());
        sb.append("]");
```

```
        sb.append("[ServiceExecuteTime=");
        sb.append(ref.getServiceExecuteTime());
        sb.append("]");
        sb.append("[ServiceState=");
        sb.append(ref.getServiceState());
        sb.append("]");
        sb.append("[RequestXML=");
        sb.append(ref.getRequestXML());
        sb.append("]");

        return sb.toString();
    }

    public String sapStore(OZUserDefinedLogTarget target){
        StringBuffer sb = new StringBuffer();
        OZSAPLogTarget ref = (OZSAPLogTarget)target;

        sb.append("[FunctionName=");
        sb.append(ref.getFunctionName());
        sb.append("]");
        sb.append("[FunctionType=");
        sb.append(ref.getFunctionType());
        sb.append("]");
        sb.append("[ResultSetTypes=");
        sb.append(ref.getResultSetTypes());
        sb.append("]");
        sb.append("[InputParameters=");
        sb.append(ref.getInputParameters());
        sb.append("]");
        sb.append("[FunctionExecuteTime=");
        sb.append(ref.getFunctionExecuteTime());
        sb.append("]");
        sb.append("[FunctionState=");
        sb.append(ref.getFunctionState());
        sb.append("]");

        return sb.toString();
    }

    public String clearQuestStore(OZUserDefinedLogTarget target){
        StringBuffer sb = new StringBuffer();
        OZClearQuestLogTarget ref = (OZClearQuestLogTarget)target;

        sb.append("[ExecuteType=");
        sb.append(ref.getExecuteType());
        sb.append("]");
        sb.append("[QuerySubType=");
        sb.append(ref.getQuerySubType());
```

```
            sb.append("]");
            sb.append("[Query=");
            sb.append(ref.getQuery());
            sb.append("]");
            sb.append("[QueryExecuteTime=");
            sb.append(ref.getQueryExecuteTime());
            sb.append("]");
            sb.append("[QueryState=");
            sb.append(ref.getQueryState());
            sb.append("]");

            return sb.toString();
        }

        public String httpStore(OZUserDefinedLogTarget target){
            OZHTTPStoreLogTarget ref = (OZHTTPStoreLogTarget)target;
            StringBuffer sb = new StringBuffer();

            sb.append("[URL=");
            sb.append(ref.getURL());
            sb.append("]");
            sb.append("[HTTPAccessTime=");
            sb.append(ref.getHTTPAccessTime());
            sb.append("]");
            sb.append("[HTTPAccessState=");
            sb.append(ref.getHTTPAccessState());
            sb.append("]");

            return sb.toString();
        }

        public String filestore(OZUserDefinedLogTarget target){
            StringBuffer sb = new StringBuffer();

                OZFileStoreLogTarget ref = (OZFileStoreLogTarget)target;
                if(ref!=null){
                    sb.append("[FilePath=");
                    sb.append(ref.getFilePath());
                    sb.append("]");
                    sb.append("[FileAccessTime=");
                    sb.append(ref.getFileAccessTime());
                    sb.append("]");
                    sb.append("[FileAccessState=");
                    sb.append(ref.getFileAccessState());
                    sb.append("]");
                }else{
                    sb.append("filestore is null");
                }
```

```
            return sb.toString();
        }

        public String commonfactor(OZUserDefinedLogTarget target){

            OZDMLogTarget ref = (OZDMLogTarget)target;
            StringBuffer sb = new StringBuffer();

            HttpServletRequest request = ref.getHttpRequest();
            if(request != null){
                sb.append("[Request Parameter=");
                Enumeration en = request.getParameterNames();
                while(en.hasMoreElements()){
                    String key = (String)en.nextElement();
                    String value = request.getParameter(key);
                    sb.append("(");
                    sb.append(key + "," + value);
                    sb.append(")");
                }
                sb.append("]");
                HttpSession session = request.getSession(false);
                if(session != null){
                    sb.append("[Session Attribute=");
                    en = session.getAttributeNames();
                    while(en.hasMoreElements()){
                        String key = (String)en.nextElement();
                        sb.append("(");
                        sb.append(key);
                        sb.append(",");
                        sb.append(session.getAttribute(key));
                        sb.append(")");
                    }
                    sb.append("]");
                }

            }
            String   ipAddress   =   (ref.getIPAddress()   ==   null)   ?   ""   :
ref.getIPAddress();
            sb.append("[IP Address=");
            sb.append(ipAddress);
            sb.append("]");
            String userid = (ref.getUserID() == null) ? "" : ref.getUserID();
            sb.append("[UserID=");
            sb.append(userid);
            sb.append("]");
            String odiName = (ref.getODIName() == null) ? "" : ref.getODIName();
            sb.append("[ODIName=");
```

```
            sb.append(odiName);
            sb.append("]");
            String dataSetName = (ref.getDataSetName() == null) ? "" :
ref.getDataSetName();
            sb.append("[DataSetName=");
            sb.append(dataSetName);
            sb.append("]");
            String dataStoreName = (ref.getDataStoreName() == null) ? "" :
ref.getDataStoreName();
            sb.append("[DataStoreName=");
            sb.append(dataStoreName);
            sb.append("]");

            return sb.toString();
        }

    /**
     * UDL을 DB로 할 경우 OZ Server에 Pool된 alias이름
     * @return String
     */
    public String getOZDBAlias() {
        return "";
    }
}
```

■ 서버 환경 설정

- ozudl.jar 파일을 서버가 설치된 경로의 \lib 폴더에 복사한 후 OZ_HOME/bin/ozcntrl.bat 파일을 편집기로 열어 사용자 정의 로그에서 사용할 클래스 파일을 설정합니다.

```
rem ---------------------------------
rem    User Defined Log Class
rem ---------------------------------
set OZLIB=%OZLIB%;%OZLIBALL%\ozudl.jar
```

- 서버의 UDL 환경 설정 파일인 OZ_HOME/conf/uslmngr.properties 파일을 편집기로 열어 아래와 같이 수정합니다.

```
#---------------------------------
# configuraion of OZ User Defined log
#---------------------------------

OZ_USER_DEFINED_LOG.Active=true
OZ_USER_DEFINED_LOG.Class=oz.udl.file.OZUserDefinedLoggerForFile
```

- 로그 파일이 생성될 경로와 파일에 기록할 로그 형식을 아래와 같이 설정한 후 서버가

설치된 경로의 \bin 폴더에 "udl_file.properties" 파일로 저장합니다.

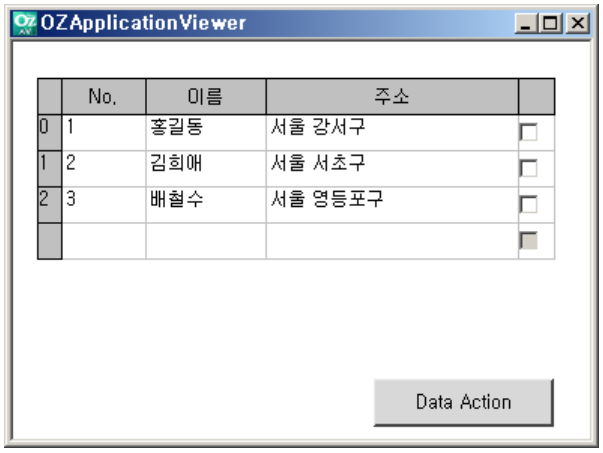
```
logpath=C:/Program Files/Forcs/OZ XStudio/OZ Server 6.0/logs/UDL.log
layout=%d{yyyy-MM-dd HH:mm:ss.SSS} [%-5p] %C{1}.%M - %m%n
AppenderType=RollingAppender
filesize=100KB
backupindex=5
#AppenderType=DailyRollingAppender
#dailytype=yyyy-MM-dd
```

※주의사항 : "logpath" 설정 시 경로에 "\"는 인식하지 못하므로 반드시 "/"로 로그 파일이 생성될 경로를 설정하여야합니다.

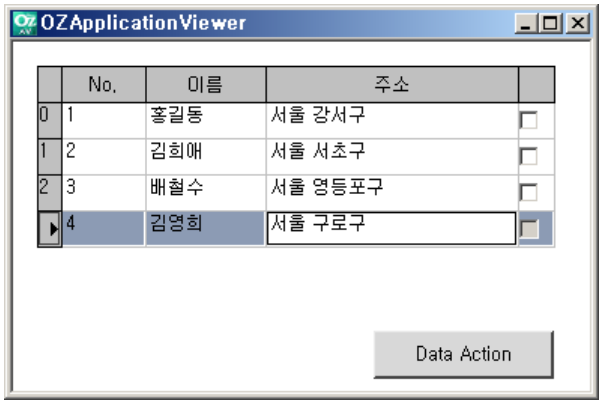
■ 데이터 액션을 수행하여 사용자 정의 로그 생성

- 자바 서버와 애플리케이션 디자이너를 구동한 후 애플리케이션 디자이너에서 구동된 서버에 접속합니다.

- 서버 리파지토리에서 데이터 액션을 수행하는 폼을 열어 미리보기합니다.



- 테이블에서 데이터를 추가한 후 [Data Action] 버튼을 클릭하여 데이터를 추가합니다.



- 서버가 설치된 경로의 /log 폴더 아래에 "UDL.log" 파일이 생성되었는지 확인합니다.

- 생성된 UDL.log 파일에 데이터 액션 시 실행된 쿼리문 및 ODI 정보 등이 로그로 기록되었는지 확인합니다.

```
[INFO ] OZUserDefinedLoggerForFile.traceLog - [DB Alias=qa_ms][IP
Address=127.0.0.1][UserID=admin][CRUDType=select][isPrepared=true][QueryP
reparedValue=][Query=select * from REQ397][Query
exec=16][QueryState=true][ODIName=/DataAction.odi][DataSetName=Sample][Da
taStoreName=DB]

[INFO ] OZUserDefinedLoggerForFile.traceLog - [DB Alias=qa_ms][IP
Address=127.0.0.1][UserID=admin][CRUDType=insert][isPrepared=true][QueryP
reparedValue=4;김영희;서울 구로구;][Query=insert into REQ397
(id,name,address)
values (?,?,?)][Query exec=16][QueryState=true]
[ODIName=/DataAction.odi][DataSetName=Sample][DataStoreName=DB]
```

UDL 예제 2 - 서버 로그를 데이터 베이스에 기록하는 방법

UDL을 이용하여 오즈 서버 로그를 데이터 베이스에 사용자가 정의한 타입의 형식으로 기록하는 방법에 대해 설명합니다.

■ **사용자 정의 로그 구현**

UDL을 사용하여 사용자 정의 로그를 데이터 베이스에 기록하는 클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 oz.udl.file.OZUserDefinedLoggerForDB 클래스를 생성한 후 해당 클래스를 "ozudl.jar"

파일로 만들었습니다.

(해당 클래스 파일에서는 오즈 서버의 db.properties 파일에 정의된 DB 커넥션 풀 중 "Sample" 데이터 베이스의 "OZDBHISTORY" 테이블에 로그를 기록하도록 설정합니다.)

```
package oz.udl.db;

import oz.udl.*;
import oz.framework.db.*;
import com.forcs.log4oz.OZLog;
import oz.framework.dac.OZDACData;

import java.sql.*;
import java.util.*;
import java.text.*;
import javax.servlet.http.*;
import oz.dm.meta.OZProcedureParameter;


public class OZUserDefinedLoggerForDB implements OZUserDefinedLogger{


   private final static String TABLENAME = "OZDBHISTORY";
   private final static String QUERY = "insert into "+TABLENAME +
                                 " values (?, ?, ?, ?, ?, ?, ?) ";


   public OZUserDefinedLoggerForDB(){}

   /**
    * 서버구동시 유저가 필요한 내용을 구현해야 할 메소드
    * @throws OZUserDefinedLogException
    */
   public     void     init(OZConnection     ozconn,     OZLog     cat)     throws
OZUserDefinedLogException{
       try{
          Connection conn = ozconn.getConnection();

          if(!isExistsTable(conn, TABLENAME)){
             String createQuery =
                "create table "+TABLENAME+" ( " +
                "property text NOT NULL, "+
                "SessionInfo text NULL, "+
                "IP varchar(27) NULL, "+   //
                 "USERID varchar(255) NULL, "+ //
                "ODINAME varchar(255) NOT NULL, "+ //
                "DataSetName varchar(255) NOT NULL, "+ //
```

```
                "DataStoreName varchar(255) NOT NULL "+ //
                " ) ";

            /*
                    "create table "+TABLENAME+" ( " +
                            "TIME_ID varchar(255) NOT NULL, "+
                            "DB_ALIAS varchar(128) NOT NULL, "+
                            "IP varchar(27) NULL, "+
                            "USERID varchar(255) NULL, "+
                            "CRUDNAME varchar(6) NULL, "+
                            "ISPrepared varchar(5) NULL, "+
                            "SessionInfo text NULL, "+
                            "preParedValue text NULL, "+
                            "Query text NULL, "+
                            "exe_time int NOT NULL, "+
                            "queryState varchar(255) NOT NULL, "+
                            "ODINAME varchar(255) NOT NULL, "+
                            "DataSetName varchar(255) NOT NULL, "+
                            "DataStoreName varchar(255) NOT NULL "+
                            " ) ";
*/

            PreparedStatement pstmt = conn.prepareStatement(createQuery);

            pstmt.executeUpdate();

            createQuery = "CREATE INDEX " + TABLENAME + "_idx_time_id ON "
                        + TABLENAME + "  (TIME_ID)";

            pstmt.close();
            pstmt = conn.prepareStatement(createQuery);
            pstmt.executeUpdate();
            pstmt.close();
        }
    }
    catch(Exception ex){
        cat.error(ex.getMessage(), ex);
        throw new OZUserDefinedLogException(ex.getMessage());
    }
  }

  /**
   * 로그에 trace할 내용인 target을 가지고 유저가 구현해야 할 메소드
   * @param target OZUserDefinedLogTarget
   * @param ozconn OZConnection
   * @param cat OZLog (서버로그에 trace되는 param 예)cat.debug(str);)
   * @throws OZUserDefinedLogException
```

```
    */
   public  void  traceLog(OZUserDefinedLogTarget  target,  OZConnection  ozconn,
OZLog cat) throws OZUserDefinedLogException{
       try{
//          OZRDBLogTarget ref = (OZRDBLogTarget)target;
          Connection conn = ozconn.getConnection();
          PreparedStatement pstmt = conn.prepareStatement(QUERY);
          StringBuffer sb2 =  new StringBuffer();
          if(target instanceof OZProcedureLogTarget) {
             sb2.append(procedureStore(target));
          }else if(target instanceof OZRDBLogTarget){
             sb2.append(rdbstroe(target));
          }else if(target instanceof OZUDSLogTarget) {
             sb2.append(udsStore(target));
          }else if(target instanceof OZFileStoreLogTarget) {
             sb2.append(filestore(target));
          }else if(target instanceof OZHTTPStoreLogTarget) {
             sb2.append(httpStore(target));
          }else if(target instanceof OZXMLLogTarget) {
             sb2.append(xmlStore(target));
          }else if(target instanceof OZSOAPLogTarget) {
             sb2.append(soapStore(target));
          }else if(target instanceof OZSAPLogTarget) {
             sb2.append(sapStore(target));
          }else if(target instanceof OZClearQuestLogTarget) {
             sb2.append(clearQuestStore(target));
          }

          ////
          OZDMLogTarget ref = (OZDMLogTarget)target;
          StringBuffer sb = new StringBuffer();

          HttpServletRequest request = ref.getHttpRequest();
          if(request != null){
             sb.append("[Request Parameter=");
             Enumeration en = request.getParameterNames();
             while(en.hasMoreElements()){
                String key = (String)en.nextElement();
                String value = request.getParameter(key);
                sb.append("(");
                sb.append(key + "," + value);
                sb.append(")");
             }
             sb.append("]");
             HttpSession session = request.getSession(false);
             if(session != null){
                sb.append("[Session Attribute=");
                en = session.getAttributeNames();
```

```
                while(en.hasMoreElements()){
                    String key = (String)en.nextElement();
                    sb.append("(");
                    sb.append(key);
                    sb.append(",");
                    sb.append(session.getAttribute(key));
                    sb.append(")");
                }
                sb.append("]");
            }

         }

         String   ipAddress   =   (ref.getIPAddress()   ==   null)   ?   ""   :
ref.getIPAddress();
         String userid = (ref.getUserID() == null) ? "" : ref.getUserID();
         String odiName = (ref.getODIName() == null) ? "" : ref.getODIName();
         String   dataSetName   =   (ref.getDataSetName()   ==   null)   ?   ""   :
ref.getDataSetName();
         String  dataStoreName  =  (ref.getDataStoreName()  ==  null)  ?  ""  :
ref.getDataStoreName();

         pstmt.setString(1, sb2.toString());
         pstmt.setString(2, sb.toString());
         pstmt.setString(3, ipAddress);
         pstmt.setString(4, userid);
         pstmt.setString(5, odiName);
         pstmt.setString(6, dataSetName);
         pstmt.setString(7, dataStoreName);

         pstmt.executeUpdate();
         pstmt.close();
      }
      catch(Exception ex){
         cat.error(ex.getMessage(), ex);
         throw new OZUserDefinedLogException(ex.getMessage());
      }

   }

   /**
    * UDL을 DB로 할 경우 OZ Server에 Pool된 alias이름
    * @return String
    */
   public String getOZDBAlias(){
      return "qa_ms";
   }
```

```
   private    SimpleDateFormat    sdf    =    new    SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS");

   /**
    * 포멧에 의한 현재시간을 가져오는 메소드
    * @return String
    */
   private String getDateTime() {
      return sdf.format(new java.util.Date(System.currentTimeMillis()));
   }



   /**
    * 테이블이 존재하는지 여부
    * @param tableName String
    * @throws SQLException
    * @return boolean
    */
   private  boolean  isExistsTable(Connection  conn,  String  tableName)  throws
SQLException
   {
      DatabaseMetaData meta = null;
      ResultSet rs = null;


      try { /////////////////;;//////////////
         if (conn != null) {
            meta = conn.getMetaData();
         } else {
            throw new SQLException("Connection is null");
         }

         String[]   tmpType   =   {"TABLE",   "VIEW",   "SYSTEM   TABLE",   "GLOBAL
TEMPORARY",
                           "LOCAL TEMPORARY", "ALIAS", "SYNONYM"};
         rs = meta.getTables(null, null, null, tmpType);

         while (rs.next()) {
            if (rs.getString(3).equals(tableName)) {
               return true;
            }
         }
      } catch (SQLException sqlEx) {
         throw sqlEx;
      } finally {
```

```
            if (rs != null)
                rs.close();
            rs = null;
            meta = null;
        }
        return false;

    }

    public String rdbstroe(OZUserDefinedLogTarget target){
            OZRDBLogTarget ref = (OZRDBLogTarget)target;
            StringBuffer sb = new StringBuffer();
            String dbAlias = (ref.getDBAlias() == null)? "" : ref.getDBAlias();
            sb.append("[DB Alias=");
            sb.append(dbAlias);
            sb.append("]");

            String crudFlagName = (ref.getCRUDFlagName() == null) ? "" :
ref.getCRUDFlagName();
            sb.append("[CRUDType=");
            sb.append(crudFlagName);
            sb.append("]");
            sb.append("[isPrepared=");
            sb.append(ref.isPrepared());
            sb.append("]");
            if(ref.getPreparedQueryValue() != null){
                sb.append("[QueryPreparedValue=");
                Vector vec = ref.getPreparedQueryValue();
                if (crudFlagName.equals("select")) {
                    for (int i = 0; i < vec.size(); i++) {
                        sb.append(vec.elementAt(i));
                    }
                }else {
                    for (int i = 0; i < vec.size(); i++) {
                        OZDACData data = (OZDACData)vec.elementAt(i);
                        sb.append(data.toStringData());
                        sb.append(";");
                    }
                }
                sb.append("]");
            }
            String query = (ref.getDBExecuteQuery() == null) ? "" :
ref.getDBExecuteQuery();
            sb.append("[Query=");
            sb.append(query);
            sb.append("]");
            long queryExec = ref.getQueryExecuteTime();
            sb.append("[Query exec=");
```

```
            sb.append(queryExec);
            sb.append("]");
            String queryState = (ref.getExecuteQueryState() == null) ? "" :
ref.getExecuteQueryState();
            sb.append("[QueryState=");
            sb.append(queryState);
            sb.append("]");

            return sb.toString();
        }

        public String udsStore(OZUserDefinedLogTarget target){
            OZUDSLogTarget ref = (OZUDSLogTarget)target;
            StringBuffer sb = new StringBuffer();

            sb.append("[ExecuteCommand=");
            sb.append(ref.getExecuteCommand());
            sb.append("]");
            sb.append("[CRUDFlagName=");
            sb.append(ref.getCRUDFlagName());
            sb.append("]");
            sb.append("[CommandExecuteTime=");
            sb.append(ref.getCommandExecuteTime());
            sb.append("]");
            sb.append("[CommandState=");
            sb.append(ref.getCommandState());
            sb.append("]");

            return sb.toString();
        }

        public String procedureStore(OZUserDefinedLogTarget target){
            OZProcedureLogTarget ref = (OZProcedureLogTarget)target;
            StringBuffer sb = new StringBuffer();

            sb.append("[DBAlias=");
            sb.append(ref.getDBAlias());
            sb.append("]");
            sb.append("[ProcedureName=");
            sb.append(ref.getProcedureName());
            sb.append("]");
            sb.append("[ParameterList=");
            ArrayList list = ref.getParameterList();
            for(int i=0;i<list.size();i++){
                OZProcedureParameter param = (OZProcedureParameter)list.get(i);
                sb.append(param.getValue());
                sb.append(",");
                sb.append(param.getDataTypeName());
```

```
            sb.append(";");
        }
        sb.append(ref.getParameterList());
        sb.append("]");
        sb.append("[ProcedureExecuteTime=");
        sb.append(ref.getProcedureExecuteTime());
        sb.append("]");
        sb.append("[ProcedureState=");
        sb.append(ref.getProcedureState());
        sb.append("]");

        return sb.toString();
    }

    public String xmlStore(OZUserDefinedLogTarget target){
        StringBuffer sb = new StringBuffer();
        OZXMLLogTarget ref = (OZXMLLogTarget)target;

        sb.append("[URL=");
        sb.append(ref.getURL());
        sb.append("]");
        sb.append("[URLAccessTime=");
        sb.append(ref.getURLAccessTime());
        sb.append("]");
        sb.append("[URLAccessState=");
        sb.append(ref.getURLAccessState());
        sb.append("]");

        return sb.toString();
    }

    public String soapStore(OZUserDefinedLogTarget target){
        StringBuffer sb = new StringBuffer();
        OZSOAPLogTarget ref = (OZSOAPLogTarget)target;

        sb.append("[ServiceName=");
        sb.append(ref.getServiceName());
        sb.append("]");
        sb.append("[Port=");
        sb.append(ref.getPort());
        sb.append("]");
        sb.append("[EndPoint=");
        sb.append(ref.getEndPoint());
        sb.append("]");
        sb.append("[ServiceExecuteTime=");
        sb.append(ref.getServiceExecuteTime());
        sb.append("]");
        sb.append("[ServiceState=");
```

```
        sb.append(ref.getServiceState());
        sb.append("]");
        sb.append("[RequestXML=");
        sb.append(ref.getRequestXML());
        sb.append("]");

        return sb.toString();
    }

    public String sapStore(OZUserDefinedLogTarget target){
        StringBuffer sb = new StringBuffer();
        OZSAPLogTarget ref = (OZSAPLogTarget)target;

        sb.append("[FunctionName=");
        sb.append(ref.getFunctionName());
        sb.append("]");
        sb.append("[FunctionType=");
        sb.append(ref.getFunctionType());
        sb.append("]");
        sb.append("[ResultSetTypes=");
        sb.append(ref.getResultSetTypes());
        sb.append("]");
        sb.append("[InputParameters=");
        sb.append(ref.getInputParameters());
        sb.append("]");
        sb.append("[FunctionExecuteTime=");
        sb.append(ref.getFunctionExecuteTime());
        sb.append("]");
        sb.append("[FunctionState=");
        sb.append(ref.getFunctionState());
        sb.append("]");

        return sb.toString();
    }

    public String clearQuestStore(OZUserDefinedLogTarget target){
        StringBuffer sb = new StringBuffer();
        OZClearQuestLogTarget ref = (OZClearQuestLogTarget)target;

        sb.append("[ExecuteType=");
        sb.append(ref.getExecuteType());
        sb.append("]");
        sb.append("[QuerySubType=");
        sb.append(ref.getQuerySubType());
        sb.append("]");
        sb.append("[Query=");
        sb.append(ref.getQuery());
        sb.append("]");
```

```
            sb.append("[QueryExecuteTime=");
            sb.append(ref.getQueryExecuteTime());
            sb.append("]");
            sb.append("[QueryState=");
            sb.append(ref.getQueryState());
            sb.append("]");

            return sb.toString();
        }

        public String httpStore(OZUserDefinedLogTarget target){
            OZHTTPStoreLogTarget ref = (OZHTTPStoreLogTarget)target;
            StringBuffer sb = new StringBuffer();

            sb.append("[URL=");
            sb.append(ref.getURL());
            sb.append("]");
            sb.append("[HTTPAccessTime=");
            sb.append(ref.getHTTPAccessTime());
            sb.append("]");
            sb.append("[HTTPAccessState=");
            sb.append(ref.getHTTPAccessState());
            sb.append("]");

            return sb.toString();
        }

        public String filestore(OZUserDefinedLogTarget target){
            StringBuffer sb = new StringBuffer();

                OZFileStoreLogTarget ref = (OZFileStoreLogTarget)target;
                if(ref!=null){
                    sb.append("[FilePath=");
                    sb.append(ref.getFilePath());
                    sb.append("]");
                    sb.append("[FileAccessTime=");
                    sb.append(ref.getFileAccessTime());
                    sb.append("]");
                    sb.append("[FileAccessState=");
                    sb.append(ref.getFileAccessState());
                    sb.append("]");
                }else{
                    sb.append("filestore is null");
                }

            return sb.toString();
        }
}
```

■ 서버 환경 설정

- ozudl.jar 파일을 서버가 설치된 경로의 \lib 폴더에 복사한 후 OZ_HOME/bin/ozcntrl.bat 파일을 편집기로 열어 사용자 정의 로그에서 사용할 클래스 파일을 설정합니다.

```
rem ---------------------------------
rem    User Defined Log Class
rem ---------------------------------
set OZLIB=%OZLIB%;%OZLIBALL%\ozudl.jar
```

- 서버의 UDL 환경 설정 파일인 OZ_HOME/conf/uslmngr.properties 파일을 편집기로 열어 아래와 같이 수정합니다.

```
#---------------------------------
# configuraion of OZ User Defined log
#---------------------------------

OZ_USER_DEFINED_LOG.Active=true
OZ_USER_DEFINED_LOG.Class=oz.udl.db.OZUserDefinedLoggerForFile
```

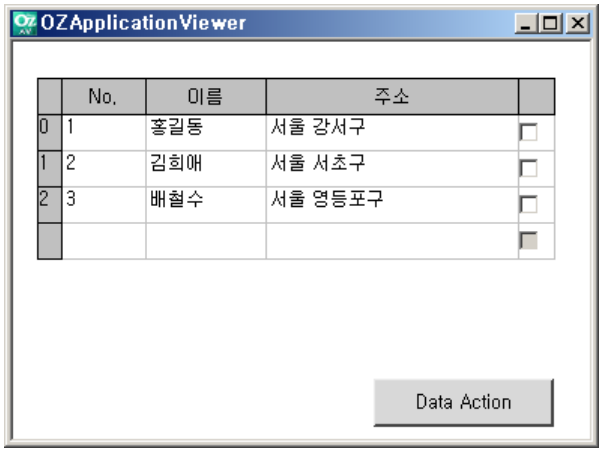
- OZ_HOME/conf/db.properties 파일에 oz.udl.db.OZUserDefinedLoggerForDB에서 사용할 데이터 베이스 커넥션 풀을 추가합니다.

```
#
#  Sample
#
Sample.vendor=MSSQL
Sample.serverAddress=127.0.0.1
Sample.user=user
Sample.password=userpw
Sample.portNo=1433
Sample.dbName=DBName
Sample.maxconns=20
Sample.initconns=5
Sample.timeout=5
Sample.doConnectionCheck=false
```

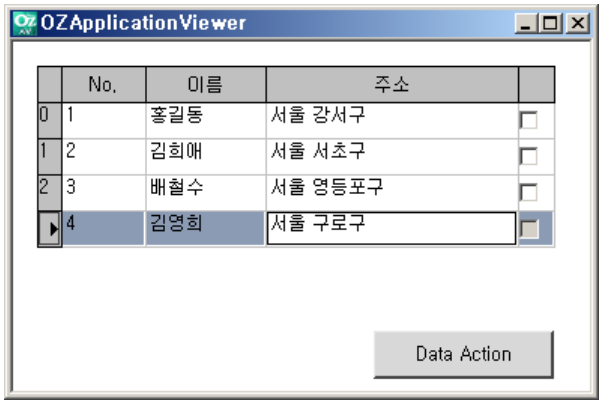
■ 데이터 액션을 수행하여 사용자 정의 로그 생성

- 자바 서버와 애플리케이션 디자이너를 구동한 후 애플리케이션 디자이너에서 구동된 서버에 접속합니다.

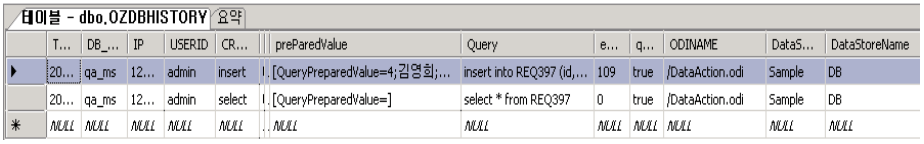
- 서버 리파지토리에서 데이터 액션을 수행하는 폼을 열어 미리보기합니다.



- 테이블에서 데이터를 추가한 후 [Data Action] 버튼을 클릭하여 데이터를 추가합니다.



- 설정한 데이터 베이스 테이블에 데이터 액션 시 실행된 쿼리문 및 ODI 정보 등이 로그로 기록되었는지 확인합니다.



UDL 예제 3 - 모니터 로그를 파일과 데이터 베이스에 기록하는 방법

자바 서버에서 OZ UDL을 사용하여 monitor.log 파일과 데이터 베이스에 사용자 정의 모니터 로그를 남기는 방법에 대해 설명합니다.

■ 사용자 정의 로그 구현

UDL을 사용하여 사용자 정의 모니터 로그를 monitor.log 파일과 데이터 베이스에 기록하는

클래스 파일을 만듭니다. 본 예제에서는 아래와 같은 java 소스 파일을 이용하여 oz.udl.db.OZUserDefinedMonitorLogger 클래스를 생성한 후 해당 클래스를 "ozudl.jar" 파일로 만들었습니다.
(해당 클래스 파일에서는 오즈 서버의 db.properties 파일에 정의된 DB 커넥션 풀 중 "ㄴ Sample" 데이터 베이스의 "OZ_MONITOR_JAVA" 테이블에 로그를 기록하도록 설정합니다.)

```
package oz.udl.db;

import oz.udl.*;
import oz.framework.db.*;
import com.forcs.log4oz.OZLog;
import oz.framework.dac.OZDACData;

import java.sql.*;
import java.util.*;
import java.text.*;
import javax.servlet.http.*;

//사용자 정의 모니터 로그를 기록할 테이블과 테이블에 데이터 추가 시 실행될 쿼리문 설정
public class OZUserDefinedMonitorLogger implements OZUserDefinedLogger {
   private final static String TABLENAME = "OZ_MONITOR_JAVA";
   private final static String QUERY = "insert into "+TABLENAME +
                                 " values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
";

   public OZUserDefinedMonitorLogger(){}

   /**
    * 서버구동시 유저가 필요한 내용을 구현해야 할 메소드
    * @throws OZUserDefinedLogException
    */
   public    void    init(OZConnection    ozconn,    OZLog    cat)    throws
OZUserDefinedLogException{
      try{
         Connection conn = ozconn.getConnection();

         if(!isExistsTable(conn, TABLENAME)){
            String createQuery =
               "create table "+TABLENAME+" ( " +
               "MARK varchar(5) NOT NULL, " +
               "THREAD_Name varchar(255) NOT NULL, " +
               "SERVICE_TIME bigint NOT NULL, " +
               "FREE_MEMORY bigint NULL, " +
               "TOTAL_MEMORY bigint NULL, " +
```

```
                "SERVICE_CODE smallint NULL, " +
                "SERVICE_STATUS smallint NULL, " +
                "SERVICE_PARAMETERS text NULL, " +
                "DB_SESSION_ID varchar(255) NULL, " +
                "EXECUTION_TIME bigint NULL, " +
                "CLIENT_IP varchar(50) NULL, " +
                "SESSION_INFO text NULL ) ";

            PreparedStatement pstmt = conn.prepareStatement(createQuery);

            pstmt.executeUpdate();

            createQuery = "CREATE INDEX " + TABLENAME + "_idx_service_time_id
ON "
                        + TABLENAME + "  (SERVICE_TIME)";

            pstmt.close();
            pstmt = conn.prepareStatement(createQuery);
            pstmt.executeUpdate();
            pstmt.close();
         }
      }
      catch(Exception ex){
         cat.error(ex.getMessage(), ex);
         throw new OZUserDefinedLogException(ex.getMessage());
      }
   }

   /**
    * 로그에 기록할 타겟 구현
    * @param target OZUserDefinedLogTarget
    * @param ozconn OZConnection
    * @param cat OZLog (서버로그에 trace되는 param 예)cat.debug(str);)
    * @throws OZUserDefinedLogException
    */
   public  void  traceLog(OZUserDefinedLogTarget  target,  OZConnection  ozconn,
OZLog cat) throws OZUserDefinedLogException{
      try{

         OZMonitorLogTarget ref = (OZMonitorLogTarget)target;
         Connection conn = ozconn.getConnection();
         PreparedStatement pstmt = conn.prepareStatement(QUERY);

         pstmt.setString(1, ref.getMark());
         pstmt.setString(2, ref.getThreadName());
         pstmt.setLong(3, ref.getServiceTime());
         pstmt.setLong(4, ref.getFreeMemory());
         pstmt.setLong(5, ref.getTotalMemory());
```

```
            pstmt.setInt(6, ref.getServiceCode());
            pstmt.setInt(7, ref.getServiceStatus());
            pstmt.setString(8, ref.getServiceParameter());
            pstmt.setString(9, ref.getDBSessionID());
            pstmt.setLong(10, ref.getExecuteTime());
            pstmt.setString(11, ref.getIPAddress());

            StringBuffer sb = new StringBuffer();
            HttpServletRequest request = ref.getHttpRequest();
            if (request != null) {
                sb.append("[Request Parameter=");

                Enumeration en = request.getParameterNames();
                while(en.hasMoreElements()) {
                    String key = (String)en.nextElement();
                    String value = request.getParameter(key);
                    sb.append("(");sb.append(key+","+value);sb.append(")");
                }
                sb.append("]");

                HttpSession session = request.getSession(false);
                if (session != null) {

                    sb.append("[Session Attribute=");

                    en = session.getAttributeNames();
                    while(en.hasMoreElements()) {
                        String key = (String)en.nextElement();
                        sb.append("(");
                        sb.append(key);
                        sb.append(",");
                        sb.append(session.getAttribute(key));
                        sb.append(")");
                    }
                    sb.append("]");
                }
                pstmt.setString(12, sb.toString());
            }else {
                pstmt.setString(12, null);
            }

            pstmt.executeUpdate();
            pstmt.close();
        }
        catch(Exception ex){
            cat.error(ex.getMessage(), ex);
            throw new OZUserDefinedLogException(ex.getMessage());
        }
```

```
    }

    /**
     * UDL을 DB로 할 경우 OZ Server에 Pool된 alias이름
     * @return String
     */
    public String getOZDBAlias(){
        return "Sample";
    }

    private SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS");

    /**
     * 포멧에 의한 현재시간을 가져오는 메소드
     * @return String
     */
    private String getDateTime() {
        return sdf.format(new java.util.Date(System.currentTimeMillis()));
    }

    /**
     * 테이블이 존재하는지 여부
     * @param tableName String
     * @throws SQLException
     * @return boolean
     */
    private boolean isExistsTable(Connection conn, String tableName) throws
SQLException
    {
        DatabaseMetaData meta = null;
        ResultSet rs = null;

        try {
            if (conn != null) {
                meta = conn.getMetaData();
            } else {
                throw new SQLException("Connection is null");
            }

            String[] tmpType = {"TABLE", "VIEW", "SYSTEM TABLE", "GLOBAL
TEMPORARY",
                                "LOCAL TEMPORARY", "ALIAS", "SYNONYM"};
            rs = meta.getTables(null, null, null, tmpType);

            while (rs.next()) {
                if (rs.getString(3).equals(tableName)) {
```

```
                return true;
             }
          }
       } catch (SQLException sqlEx) {
          throw sqlEx;
       } finally {
          if (rs != null)
             rs.close();
          rs = null;
          meta = null;
       }
       return false;
    }
}
```

■ 서버 환경 설정

- ozudl.jar 파일을 서버가 설치된 경로의 \lib 폴더에 복사한 후 OZ_HOME/bin/ozcntrl.bat 파일을 편집기로 열어 사용자 정의 로그에서 사용할 클래스 파일을 설정합니다.

```
rem ---------------------------------
rem    User Defined Log Class
rem ---------------------------------
set OZLIB=%OZLIB%;%OZLIBALL%\ozudl.jar
```

- 서버의 UDL 환경 설정 파일인 OZ_HOME/conf/uslmngr.properties 파일을 편집기로 열어 아래와 같이 수정합니다.

```
#---------------------------------
# configuraion of OZ User Defined log
#---------------------------------

OZ_UDL_MONITOR.Active=true
OZ_UDL_MONITOR.Class=oz.udl.db.OZUserDefinedMonitorLogger
```

- db.properties 파일에 oz.udl.db.OZUserDefinedLoggerForDB에서 사용할 데이터베이스 커넥션 풀을 추가합니다.

```
#
#  Sample
#
Sample.vendor=MSSQL
Sample.serverAddress=127.0.0.1
Sample.user=user
Sample.password=userpw
Sample.portNo=1433
Sample.dbName=DBName
```

```
Sample.maxconns=20
Sample.initconns=5
Sample.timeout=5
Sample.doConnectionCheck=false
```

■ 사용자 정의 모니터 로그 생성

- 서버와 애플리케이션 디자이너를 구동한 후 애플리케이션 디자이너에서 구동된 서버에 접속합니다.

- 서버 리파지토리에서 보고서를 열기하여 모니터 로그에 요청 사항을 기록합니다.

- 서버가 설치된 경로의 /log/monitor.log 파일에 보고서 요청 시 클라이언트 IP 및 서비스 상태 등이 로그로 기록되었는지 확인합니다.

```
start 120224 1184045287342130799136 133955584 null null null null null  null

end 120224 1184045287358130664648 133955584 142 9001  item name;Car.odi,item
type;10000,category name; /127.0.0.1 16

start 4523271 1184045782686130517528 133955584 null null null null null null

end 4523271 1184045782717130399088 133955584 122 9001      item id;333960544
127.0.0.1 31
```

- 설정한 데이터 베이스 테이블에 보고서 요청 시 클라이언트 IP 및 서비스 상태 등이 로그로 기록되었는지 확인합니다.

	MARK	THREA...	SERVICE_TIME	FREE_MEMORY	TOTAL_MEMORY	SE...	SERV...	SERVICE_PARAMET...	D...	EXE...	CLIENT_IP	SESSION_INFO
	start	120224	1184045287342	130799136	133955584	-1	-1			-1	127.0.0.1	NULL
	end	120224	1184045287358	130664648	133955584	142	9001	:ar.odi,item type;10000;		16	127.0.0.1	NULL
	end	4523271	1184045782717	130399088	133955584	122	9001	item id;333960544		31	127.0.0.1	NULL
	end	3981922	1184045782748	130241848	133955584	174	9001	item name;DataActio...		0	127.0.0.1	NULL

오즈 스케줄러 서버에서 사용

OZSchedulerUserDefinedLogger 인터페이스

오즈 스케줄러에서 사용자 정의 로그 기능을 실행하기 위한 기본 인터페이스입니다.

■ **init**

Prototype	void init(org.apache.log4j.Category cat) throws OZSchedulerUDLException
Definition	스케줄러 서버 구동 시 호출됩니다.
Argument	*cat* 스케줄러 서버 구동 시 로그에 남길 내용

■ **release**

Prototype	void release(org.apache.log4j.Category cat) throws OZSchedulerUDLException
Definition	스케줄러 서버 종료 시 호출됩니다.
Argument	*cat* 스케줄러 서버 종료 시 로그에 남길 내용

■ **traceLog**

Prototype	void traceLog(OZSchedulerUDLTarget target, org.apache.log4j.Category cat) throws OZSchedulerUDLException
Definition	로그를 기록합니다.
Argument	*target* 사용자 정의 로그 타겟
	cat 로그에 남길 내용

OZSchedulerUDLTarget 인터페이스의 클래스 구조

oz.scheduler.udl.OZSchedulerUDLTarget



- **oz.scheduler.udl.OZRequestTarget**
- **oz.scheduler.udl.OZExecuteTarget**
- **oz.scheduler.udl.OZEndTarget**
- **oz.scheduler.udl.OZEndTarget**

■ **OZSchedulerUDLTarget 인터페이스 함수**

오즈 스케줄러 UDL에서 제공하는 기본 Target 인터페이스입니다.

- getConcurrentCount

Prototype	`int getConcurrentCount()`
Definition	실행 중인 태스크 개수를 가져옵니다.

- getWaitQueueCount

Prototype	`int getWaitQueueCount()`
Definition	대기 중인 태스크 개수를 가져옵니다.

- getTaskID

Prototype	`String getTaskID()`
Definition	태스크 ID를 가져옵니다.

- getMarkType

Prototype	`String getMarkType()`
Definition	실행 타입을 가져옵니다. "REQ", "EXE", "END" 중 하나의 값으로 리턴됩니다.

- getFreeMemory

Prototype	`long getFreeMemory()`
Definition	사용 가능한 메모리를 가져옵니다. (단위 : byte)

- getTotalMemory

Prototype	`long getTotalMemory()`

Definition	전체 메모리를 가져옵니다. (단위 : byte)

- getServiceTime

Prototype	`long getServiceTime()`
Definition	각 실행 시점의 시간(1970년 1월 1일 0시부터 현재까지의 시간)을 가져옵니다. (단위 : mesc)

■ **OZRequestTarget 인터페이스 함수**

태스크 요청 시 호출되는 Target 인터페이스입니다.

- getConfigType

Prototype	`String getConfigType()`
Definition	설정 타입을 가져옵니다. 새로 생성인 경우 "new"로, 수정인 경우 "edit"로 리턴됩니다.

■ **OZExecuteTarget 인터페이스 함수**

태스크 실행 시 호출되는 Target 인터페이스입니다.

- getReportFiles

Prototype	`String[] getReportFiles()`
Definition	카테고리 경로를 포함한 리포트 이름을 가져옵니다.

- getSchedulingType

Prototype	`String getSchedulingType()`
Definition	스케줄링 타입을 가져옵니다. 즉시 실행인 경우 "Immediately"로, 한 번만 실행인 경우 "Once at specific time"으로, 주기적 실행인 경우 "Periodically"로 리턴됩니다.

- getViewerParamString

Prototype	`String getViewerParamString()`
Definition	뷰어 패러미터를 가져옵니다.

■ **OZEndTarget 인터페이스 함수**

태스크 종료 시 호출되는 Target 인터페이스입니다.

- getExecuteTime

Prototype	`long getExecuteTime()`
Definition	수행 시간을 가져옵니다. (단위 : mesc)

- getExportedFileName

Prototype	`String[] getExportedFileName()`
Definition	익스포트된 파일 이름을 가져옵니다.

- isSuccessfulString

Prototype	`String isSuccessfulString()`
Definition	성공 여부 메시지를 가져옵니다.

- getErrorMsg

Prototype	`String getErrorMsg()`
Definition	에러 메시지를 가져옵니다.

■ **OZTaskInfoTarget 인터페이스 함수**

태스크 실행 시 태스크 정보가 저장되는 Target 인터페이스입니다.

- getTaskGroupName

Prototype	`String getTaskGroupName()`
Definition	태스크 그룹 이름을 전체 경로로 가져옵니다.

- getTaskName

Prototype	`String getTaskName()`
Definition	태스크 이름을 가져옵니다.

- getTaskProperty

Prototype	`HashMap getTaskProperty()`
Definition	태스크 정보를 HashMap 형태(key와 value)로 가져옵니다. 리턴되는 key는 task_type이며, task_type은 아래와 같은 value를 가집니다. • Basic : 별도 방식으로 처리한 태스크 • Direct : 뷰어 태그 방식으로 처리한 태스크 • SDM : SDM 생성 태스크 • Print : 프린트 생성 태스크

■ **OZSchedulerUDLException 함수**

UDL에서 발생되는 Exception입니다.

- getSchedulingType

Prototype	public OZSchedulerUDLException(String msg)
Definition	UDL에서 발생되는 Exception입니다.
Argument	*msg* 에러 메시지

UDL 구현 시 오즈 스케줄러 서버 설정 방법

■ 라이브러리 파일 복사

- UDL 기능이 구현된 jar 파일을 스케줄러 서버가 설치된 경로의 lib 폴더에 복사합니다.
- 로그 내용을 데이터베이스에 기록할 경우 해당 DBMS의 JDBC 드라이버 파일도 lib 폴더에 복사합니다.

■ 오즈 스케줄러 서버 실행 설정 파일 변경

- scheduler.bat/scheduler.sh 파일을 편집기로 열어 스케줄러 서버가 구동될 때 UDL 기능이 구현된 라이브러리 파일이 로딩되도록 설정합니다. 만일 로그 내용을 데이터베이스에 기록할 경우 해당 DBMS의 JDBC 드라이버 파일도 로딩되도록 설정합니다.

  ex) UDL 기능이 구현된 라이브러리 파일이 ozscheduler_udl.jar이고, MSSQL에 로그 내용을 기록하는 경우

```
...
rem ----------------------------------
rem Library for UDL
rem ----------------------------------
set OZSchLib=%OZSchLib%;%OZSCHEDULER_HOME%\lib\ozscheduler_udl.jar
set OZSchLib=%OZSchLib%;%OZSCHEDULER_HOME%\lib\msbase.jar
set OZSchLib=%OZSchLib%;%OZSCHEDULER_HOME%\lib\mssqlserver.jar
set OZSchLib=%OZSchLib%;%OZSCHEDULER_HOME%\lib\msutil.jar
...
```

- scheduler_server_log.properties 파일을 편집기로 열어 UDL 기능을 사용하도록 OZ_UDL.Active 옵션을 true로 설정하고 OZ_UDL.Class 옵션에 클래스 이름을 설정합니다.

```
...
OZ_UDL.Active=true
OZ_UDL.Class=oz.scheduler.udl.file.OZSchedulerUDLForFile
```

Ⅷ. User Defined post Execute

- UDE 인터페이스
- UDE 구현

UDE 인터페이스

UDE(User Defined post Execute)은 태스크 실행이 정상적으로 완료된 후 특정 기능을 실행할 수 있도록 하는 인터페이스입니다.

Interface Summary

■ **UDE 관련 인터페이스**

- **public interface class OZSchedulerUserDefinedPostTaskExecuteEx**

Interface Method

[OZSchedulerUserDefinedPostTaskExecuteEx Interface Method]

■ **postTaskExecuteSuccess**

Prototype	boolean postTaskExecuteSuccess(oz.util.SortProperties configMap, oz.util.SortProperties exportMap, OZUDEPostTarget target, org.apache.log4j.Category cat) throws OZSchedulerUDEException	
Definition	태스크 실행을 성공한 경우 그 다음에 실행시킬 후 처리 프로그램을 실행합니다.	
Argument	*configMap*	태스크 수행할 Config Properties postTaskExecute를 실행하려면 Config Properties에 ude.classname을 설정하여야 합니다.
	exportMap	태스크 수행할 Export Properties
	target	태스크 수행 후 결과 객체
	cat	수행 로그

■ **postTaskExecuteFail**

Prototype	boolean postTaskExecuteFail(oz.util.SortProperties configMap, oz.util.SortProperties exportMap, OZUDEPostTarget target, org.apache.log4j.Category cat) throws OZSchedulerUDEException

Definition	태스크를 실행하였으나 실패한 경우 그 다음에 실행시킬 후 처리 프로그램을 실행합니다.	
Argument	*configMap*	태스크 수행할 Config Properties postTaskExecute를 실행하려면 Config Properties에 ude.classname을 설정하여야 합니다.
	exportMap	태스크 수행할 Export Properties
	target	태스크 수행 후 결과 객체
	cat	수행 로그

UDE 구현

UDE 인터페이스 예제

다음은 UDE 인터페이스를 구현한 예제입니다.

```
package oz.ude;

import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.GregorianCalendar;

import com.forcs.zip.*;
import org.apache.log4j.*;

import oz.scheduler.SchedulerException;
import oz.scheduler.ude.OZSchedulerUDEException;
import oz.scheduler.ude.OZSchedulerUserDefinedPostTaskExecute;
import oz.scheduler.ude.OZUDEPostTarget;
import oz.util.SortProperties;

public class OZSchedulerUDE implements OZSchedulerUserDefinedPostTaskExecute {

   public OZSchedulerUDE() {
   }

  public boolean postTaskExecute(SortProperties configMap, SortProperties exportMap,
OZUDEPostTarget target, Category cat) throws OZSchedulerUDEException {

   try
   {
     cat.debug("[POST-TASK] init.");

     //String exportedMoveFolder = configMap.getProperty("ude.exportzipfolder");
     String zipfilename = configMap.getProperty("ude.zipfilename");
     String password = configMap.getProperty("ude.zippassword");

     if (zipfilename == null || "".equals(zipfilename))
     {
       throw new Exception("Zip file name is null.");
     }else
     {
       cat.debug("[POST-TASK] Zip file name is " +zipfilename);
```

```
    }

    if (target == null)
    {
      throw new Exception("target is null.");
    }

    zipfilename = convertDateFormat(zipfilename);

    if (!zipfilename.toLowerCase().endsWith(".zip"))
      {
      zipfilename = zipfilename + ".zip";
      }

    cat.debug("[POST-TASK] zip start.");

      // Create the ZIP file
      ZipOutputStream out = null;

      try {
        new File(zipfilename).getParentFile().mkdirs();

        out = new ZipOutputStream(new BufferedOutputStream(new
FileOutputStream(zipfilename)));

        if (password != null && !"".equals(password))
        {
          out.setPassword(password.getBytes());
        }

        String[] exportFolders = target.getExportFolder();
             cat.debug("[POST-TASK] exportFolder size = "+ exportFolders.length);

        for(int i=0; i<exportFolders.length; i++)
        {

          File[] fileList= new File(exportFolders[i]).listFiles();

          if (fileList != null && fileList.length > 0)
          {
            String filename = fileList[0].getName();
            int idx = filename.lastIndexOf(".");
            String ext = filename.substring(idx+1);

            out.putNextEntry(new ZipEntry(ext+"/"));
                  out.closeEntry();

                  for(int j=0; j<fileList.length; j++)
```

```
                    {
                      filename = fileList[j].getName();
                      filezip(ext+"/" + filename, new FileInputStream(fileList[j]),
out);
                    }
          }
              }

       }catch(Throwable ex)
       {
         throw ex;
       }finally
       {
         if (out != null) try { out.close(); }catch(Exception ex){}
       }
       cat.debug("[POST-TASK] zip finish.");

/*
       cat.debug("[POST-TASK] move start.");

       String[] exportFolders = target.getExportFolder();

     for(int i=0; i<exportFolders.length; i++)
     {

       File[] fileList= new File(exportFolders[i]).listFiles();

       if (fileList != null && fileList.length > 0)
       {
         String filename = fileList[0].getName();
         int idx = filename.lastIndexOf(".");
         String ext = filename.substring(idx+1);


         String folder = exportedMoveFolder + "/" + ext;

         new File(folder).mkdirs();

                for(int j=0; j<fileList.length; j++)
                {
                  filename = fileList[j].getName();
                  fileCopy(new FileInputStream(fileList[j]), new
FileOutputStream(folder + "/" + filename));
                }
       }
          }
     cat.debug("[POST-TASK] move finish.");
*/
```

```
     cat.debug("[POST-TASK] End.");
   }catch(Throwable t)
   {
     cat.error(t.getLocalizedMessage(), t);
     throw new OZSchedulerUDEException(t.getLocalizedMessage());
   }

   return true;
 }

   private void filezip(String filename, FileInputStream in, ZipOutputStream out)
throws FileNotFoundException, IOException {

     // Add ZIP entry to output stream.
     out.putNextEntry(new ZipEntry(filename));

       try
       {
         // Transfer bytes from the file to the ZIP file
         // Create a buffer for reading the files
         byte[] buf = new byte[4096];
         int len;
         while ((len = in.read(buf)) > 0) {
           out.write(buf, 0, len);
         }

         // Complete the entry
         out.closeEntry();
       }finally
       {
         if (in != null) in.close();
       }
   }
/*
   private void fileCopy(FileInputStream is, FileOutputStream out) throws Exception
{

       BufferedInputStream bis = new BufferedInputStream(is);

       int bytesRead = 0;
       byte[] buffer = new byte[10000];
       try {
          while((bytesRead = bis.read(buffer)) >=0) {
            out.write(buffer,0, bytesRead);
          }
          bis.close();
       } catch(Exception e) {
          throw e;
```

```
      }finally
      {
        if (out != null)
        {
          out.close();
        }
      }
   }
*/

   private String convertDateFormat(String val) throws Exception {

      if (val == null) return null;

      int idx$1;
      idx$1 = val.indexOf("$OZ.#D/");
      if(idx$1 == -1) {
         return val;
      }

      int idxRP,idx$2;
      char opType;
      int addDay;
      String convertedDate;

      Calendar cc = new GregorianCalendar();

      if((idxRP = val.indexOf("/#D$",idx$1)) != -1) {
         idx$2 = idxRP + 3;
      }
      else {
         if((idxRP = val.indexOf("/#D+",idx$1)) != -1) {
            opType = 'P';
         }
         else if((idxRP = val.indexOf("/#D-",idx$1)) != -1) {
            opType = 'M';
         }
         else {
            throw new Exception(val + " : can't find '/#D $' or '/#D[op]'");
         }

         if((idx$2 = val.indexOf('$',idxRP)) == -1) {
            throw new SchedulerException(val + " : can't find '$' following
'/#D[op][num]'");
         }

         try {
            if(opType=='P') {
```

```
                addDay = Integer.parseInt(val.substring(idxRP+4,idx$2).trim());
             }
             else {
                addDay = -Integer.parseInt(val.substring(idxRP+4,idx$2).trim());
             }

             cc.add(Calendar.DATE,addDay); //오늘 날짜를 기준으로 새로운 날짜 생성.
          }
          catch (NumberFormatException e) {
             throw new SchedulerException(val+" : invalid number format in
'/#D[op][num]$'");
          }
       }

       SimpleDateFormat df = new SimpleDateFormat(val.substring(idx$1+7,idxRP));
       convertedDate = df.format(cc.getTime());

       return val.substring(0,idx$1)+convertedDate+val.substring(idx$2+1);
    }


}
```

Ⅸ. User Defined Resource

- UDR 인터페이스
- UDR 구현

UDR 인터페이스

UDR(User Defined Resource)은 보고서의 리소스 필드를 생성할 때 리소스 필드 값을 리포트 디자이너에서 일일이 설정하지 않고 데이터베이스에 입력된 리소스 정보로 한꺼번에 설정할 수 있도록 하는 인터페이스입니다.

관련 Interface

- **syncResource(oz.udr.IOZUserDefinedResource)**

Prototype	`public OZUDRResult syncResource(OZConnection ozconn, OZLog cat, String props, OZUDRInfo info) throws Exception`	
Definition	리소스를 동기화합니다.	
Argument	*ozconn*	DB의 Connection
	cat	서버 로그에 남을 카테고리
	props	사용자가 정의한 Property
	info	보고서에 있는 리소스 정보 객체
Return	OZUDRResult	DB 정보 동기화한 결과 객체

관련 Class - oz.udr.OZUDRInfo

- Constructor
  - OZUDRInfo

Prototype	`public OZUDRInfo(String category, String reportname, String[] language, Resource[] resource)`	
Argument	*category*	보고서 카테고리
	reportname	보고서 이름
	language	언어 리스트 ex) ko/kr, en/us
	resource	리소스 정보 객체

- Method
  - getCategory

Prototype	`public String getCategory()`
Definition	카테고리 이름을 가져옵니다.

- setCategory

Prototype	`public void setCategory(String category)`
Definition	카테고리 이름을 설정합니다.
Argument	*category* 카테고리 이름

- getReportname

Prototype	`public String getReportname()`
Definition	리포트 이름을 가져옵니다.

- setReportname

Prototype	`public void setReportname(String reportname)`
Definition	리포트 이름을 설정합니다.
Argument	*reportname* 리포트 이름

- getLanguage

Prototype	`public String[] getLanguage()`
Definition	언어 리스트를 가져옵니다.

- setLanguage

Prototype	`public void setLanguage(String[] language)`
Definition	언어 리스트를 설정합니다.
Argument	*language* 언어 리스트

- getResource

Prototype	`public Resource[] getResource()`
Definition	리소스를 가져옵니다.

- setResource

Prototype	`public void setResource(Resource[] resource)`
Definition	리소스를 설정합니다.

Argument	*resource*	리소스

관련 Class - oz.udr.Resource

■ Constructor

- Resource

Prototype	public Resource(String name, String desc, String[] value)	
Argument	*name*	리소스 필드 이름
	desc	리소스 설명
	value	언어별 리소스 값(OZUDRInfo language index와 같은 index 값이어야 함)

■ Method

- getName

Prototype	public String getName()
Definition	리소스 필드 이름을 가져옵니다.

- setName

Prototype	public void setName(String name)	
Definition	리소스 필드 이름을 설정합니다.	
Argument	*name*	리소스 필드 이름

- getDesc

Prototype	public String getDesc()
Definition	리소스 설명을 가져옵니다.

- setDesc

Prototype	public void setDesc(String desc)	
Definition	리소스 설명을 설정합니다.	
Argument	*desc*	리소스 설명

- getValue

Prototype	public String[] getValue()

Definition	언어별 리소스 값을 가져옵니다.

- setValue

Prototype	`public void setValue(String[] value)`
Definition	언어별 리소스 값을 설정합니다.
Argument	*value* 언어별 리소스 값

- getDescHash

Prototype	`public HashMap getDescHash()`
Definition	리소스 설명을 파싱할 Hash에 key, value 형태로 가지고 있는 객체를 가져옵니다.

관련 Class - oz.udr.OZUDRResult

■ Constructor

- OZUDRResult

Prototype	`public OZUDRResult(boolean isUpdate, OZUDRInfo info)`	
Argument	*isUpdate*	업데이트 여부
	info	사용자가 정의한 리소스 정보를 다시 저장한 객체

■ Method

- isUpdate

Prototype	`public boolean isUpdate()`
Definition	업데이트 여부를 가져옵니다.

- getUDRInfo

Prototype	`public OZUDRInfo getUDRInfo()`
Definition	사용자가 정의한 리소스 정보를 다시 저장한 객체를 가져옵니다.

UDR 구현

UDR 인터페이스 예제

예를 들어, 리소스 정보를 저장한 테이블의 데이터 구조가 아래와 같고,

데이터 필드 이름	데이터 필드 타입	설명
ProgramID	varchar(40)	OZR 파일 전체 경로
ObjectID	varchar(50)	리소스 필드 이름
DD_KO	varchar(400)	한국어 리소스 필드 값
DD_EN	varchar(400)	영어 리소스 필드 값
DD_JP	varchar(400)	일본어 리소스 필드 값
DD_CN	varchar(400)	중국어 리소스 필드 값

필드 이름과 한국어, 영어, 일본어, 중국어에 대한 리소스 필드의 필드 이름과 필드 값을 동기화할 경우 다음과 같이 구현합니다.

```
package oz.udr;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Arrays;
import java.util.Vector;

import oz.framework.db.OZConnection;
import com.forcs.log4oz.OZLog;

public class OZUserDefinedResourceImpl implements IOZUserDefinedResource {
  OZLog cat = null;
  private boolean isUpdate = false;
  private int languageLength = 0;
  private String languges = "";

  public OZUDRResult syncResource(OZConnection ozconn, OZLog cat, OZUDRInfo
info) throws Exception
  {
    this.cat = cat;
    cat.info("OZUserDefinedResourceImpl sync begin...");
    if (ozconn == null) throw new Exception("Connection is null.");
    String programID = getProgramID(info);
```

```
    languageLength = info.getLanguage().length;
    languges = OZStringTokenJoin(Delim, info.getLanguage());
    int size = info.getResource().length;
    for(int i=0; i<size; i++)
    {
      Resource resource = info.getResource()[i];
      String objectID = resource.getName();
      if (objectID != null &&  !"".equals(objectID) )
        condition(ozconn.getConnection(), programID, objectID, info, i);
      }
      OZUDRResult result = new OZUDRResult(isUpdate, info);
      cat.info("OZUserDefinedResourceImpl sync end...");
      return result;
    }

    private String getProgramID(OZUDRInfo info)
    {
      String reportname = info.getReportname();
      int idx = reportname.toLowerCase().lastIndexOf(".ozr");
      if (idx >= 0)
      {
        reportname = reportname.substring(0, idx);
      }
      return info.getCategory() + reportname;
    }

    public void condition(Connection conn, String programID, String objectID,
OZUDRInfo info, int index) throws Exception
    {
      PreparedStatement pstmt = null;
      ResultSet rs = null;
      boolean isInsert = false;
      try {
        String query = "select * from DD where ProgramID = ? and ObjectID
= ?";
        pstmt = conn.prepareStatement(query);
        pstmt.setString(1, programID);
        pstmt.setString(2, objectID);
        rs = pstmt.executeQuery();
        Vector vec = null;
        Vector vecValue = null;
        String[] targetLanguage = null;
        String[] targetValue = null;
        if (rs.next())
        {
          vec = new Vector();
          vecValue= new Vector();
          String ko_name = rs.getString("DD_KO");
```

```
            if (ko_name != null)
            {
              vec.add("ko/kr");
              vecValue.add(ko_name);
            }
            String en_name = rs.getString("DD_EN");
            if (en_name != null)
            {
              vec.add("en/us");
              vecValue.add(en_name);
            }
            String cn_name = rs.getString("DD_CN");
            if (cn_name != null)
            {
              vec.add("zh/cn");
              vecValue.add(cn_name);
            }
            String jp_name = rs.getString("DD_JP");
            if (jp_name != null)
            {
              vec.add("ja/jp");
              vecValue.add(jp_name);
            }
            targetLanguage = new String[vec.size()];
            targetValue = new String[vec.size()];
            vec.copyInto(targetLanguage);
            vecValue.copyInto(targetValue);
            String compare1 = OZStringTokenJoin(Delim, targetLanguage);
            if(!isUpdate)
            {
              if(!compare1.equals(languges))
              {
                isUpdate = true;
              }
            }
            compare1 = OZStringTokenJoin(Delim,targetValue);
            String                          compare2                          =
OZStringTokenJoin(Delim,info.getResource()[index].getValue());
            if(!isUpdate)
            {
              if(!compare1.equals(compare2))
              {
                isUpdate = true;
              }
            }
            info.setLanguage(targetLanguage);
            info.getResource()[index].setValue(targetValue);
            isInsert = false;
```

```
            }else
            {
               isInsert = true;
            }
         } catch(Exception e) {
            cat.error(e.getMessage(), e);
            throw e;
         } finally {
            try{
               if (rs != null) rs.close();
               if(pstmt != null) pstmt.close();
            } catch(Exception e) {
         }
         if (isInsert)
         {
            insert(conn, programID, objectID, info, index);
         }
       }
    }

  public void insert(Connection conn, String programID, String objectID,
OZUDRInfo info, int index) throws Exception {
  PreparedStatement pstmt = null;
    try {
       conn.setAutoCommit(false);
       String            query            =            "insert            into
DD(ProgramID,ObjectID,DD_KO,DD_EN,DD_CN,DD_JP) values(?,?,?,?,?,?)";
       pstmt = conn.prepareStatement(query);
       pstmt.setString(1, programID);
       pstmt.setString(2, objectID);
       boolean[] flag = new boolean[4];
       Arrays.fill(flag, false);
       String dd_ko = "";
       for(int i=0; i<info.getLanguage().length; i++)
       {
         if (info.getLanguage()[i].equalsIgnoreCase("ko/kr"))
         {
            dd_ko = info.getResource()[index].getValue()[i];
            pstmt.setString(3, dd_ko);
            flag[0] = true;
         }else if (info.getLanguage()[i].equalsIgnoreCase("en/us"))
         {
            pstmt.setString(4, info.getResource()[index].getValue()[i]);
            flag[1] = true;
         }else if (info.getLanguage()[i].equalsIgnoreCase("zh/cn"))
         {
            pstmt.setString(5, info.getResource()[index].getValue()[i]);
            flag[2] = true;
```

```
        }else if (info.getLanguage()[i].equalsIgnoreCase("ja/jp"))
        {
          pstmt.setString(6, info.getResource()[index].getValue()[i]);
          flag[3] = true;
        }
      }
      for(int i=0; i<flag.length; i++)
      {
        if (!!!flag[i])
        {
          pstmt.setString(i+3, dd_ko);
        }
      }
      pstmt.execute();
      conn.commit();
      cat.info("[DD] table successed insert.");
    }
    catch (Exception e) {
      cat.error("UDR Error : " + e.getMessage(), e);
      conn.rollback();
      throw e;
    }
    finally {
      if (pstmt != null) {
        pstmt.close();
      }
    }
  }

  public static final String Delim = "#%$oz*&^";
    private String OZStringTokenJoin(String token, String[] strings)
    {
      StringBuffer buffer = new StringBuffer();
      int count = strings.length - 1;
      for(int i = 0; i < count; i++)
      {
        buffer.append(strings[i]);
        buffer.append(token);
      }
      if(count >= 0)
      {
        buffer.append(strings[count]);
      }
      return buffer.toString();
    }
  }
```

X. 사용자 컴포넌트

- 연동 방법
- 사용자 컴포넌트 C 구현 함수

연동 방법

사용자 컴포넌트는 라벨의 그리기 타입 중의 하나입니다.
라벨의 그리기 타입을 사용자 컴포넌트로 하면 해당 모듈을 로딩하여 보고서 디자인시 사용하며, 실제 서비스할 때는 오즈 뷰어에서 사용합니다.

본 매뉴얼에서는 사용자 컴포넌트의 이름을 "PDF417"라고 가정하고 설명합니다.

사용자 컴포넌트 DLL 명명 규칙

명명규칙 : ozc_컴포넌트명.dll
만일 사용자 컴포넌트 이름을 " PDF 417"로 한다면 "ozc_pdf417.dll"로 명명하여야 합니다.

오즈 리포트 디자이너와의 연동

사용자 컴포넌트를 launch.cfg의 classpath에 추가한 후 사용합니다.

뷰어와 연동

뷰어는 사용자 컴포넌트를 사용할 경우(화면 보기, 인쇄, 속성 편집 등)에 ozc_pdf417.dll를 WIN32 API의 LoadLibrary를 사용하여 로드해서 합니다.

DLL을 클라이언트에 배포시에는 ozuser.zip과 같이 dll을 zip으로 압축한 후 웹서버의 오즈 뷰어 탑재 위치(예: ozrviewer.idf 파일이 있는 위치)로 옮긴 후 ZTransfer에서 사용하는 설치 정보 파일(*.idf)에서 다운로드하여 설치합니다. 자세한 내용은 "오즈 리포트 뷰어 매뉴얼의 Ⅱ. 오즈 뷰어 설치 및 제거"를 참조하시기 바랍니다.

※ 주의사항 : 익스포트시 PDF와 SVG는 사용자 컴포넌트가 배경과 테두리만 저장되고 내용은 저장되지 않습니다. Excel, Word, Powerpoint, HTML 익스포트의 경우에는 이미지 형태로 저장됩니다.

사용자 컴포넌트 C 구현 함수

함수

■ 생성, 복사, 삭제 함수

- GetNewInstance

Prototype	DWORD GetNewInstance()
Definition	사용자 컴포넌트 객체 생성 함수입니다.
Return	사용자 컴포넌트 객체의 핸들을 반환, 에러 발생 시 0(NULL)을 반환

- GetCopyInstance

Prototype	DWORD GetCopyInstance(DWORD src)	
Definition	사용자 컴포넌트 객체 복사 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
Return	복사된 객체의 핸들을 반환, 에러 발생 시 0(NULL)을 반환	

- DeleteInstance

Prototype	void DeleteInstance(DWORD src)	
Definition	사용자 컴포넌트 객체 삭제 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들

■ 속성 **Serialize** 함수

- getAttrListLength

Prototype	int getAttrListLength(DWORD src)	
Definition	사용자 컴포넌트가 사용하는 속성 개수를 반환하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
Return	속성 개수를 반환 반환, 에러 발생 시 -1을 반환	

- getAttrList

Prototype	BOOL getAttrList(DWORD src, char** attrs, const int length)

Definition	사용자 컴포넌트에서 사용하는 속성명들을 배열에 넣어서 attrs로 돌려줍니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	attrs	속성명이 반환되는 버퍼의 Pointer
	length	getAttrListLength를 호출하여 나온 속성 개수
Return	성공하면 true를 반환, 에러 발생 시 false를 반환	

- getAttrLength

Prototype	`int getAttrLength(DWORD src, const char* name)`	
Definition	입력된 속성 값의 길이를 반환하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	name	속성명
Return	속성 값의 길이를 반환, 에러 발생 시(자신의 속성이 아닌 경우) -1을 반환	

- getAttr

Prototype	`BOOL getAttr(DWORD src, const char* name, char* value, const int value_length)`	
Definition	name에 해당하는 속성 값을 value로 반환하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	name	속성명
	value	속성의 값을 저장하는 장소
	value_length	getAttrLength를 호출하여 나온 값
Return	성공하면 true를 반환, 에러 발생 시(자신의 속성이 아닌 경우) false를 반환	

- setAttr

Prototype	`BOOL setAttr(DWORD src, const char* name, const char* value)`	
Definition	입력된 속성명의 속성 값을 변경하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	name	속성명
	value	변경하려는 속성의 값
Return	속성 값의 길이를 반환, 에러 발생 시(자신의 속성이 아닌 경우) -1을 반환	

- writeMe

Prototype	`char* writeMe(DWORD src, int * length)`	
Definition	사용자 컴포넌트의 속성을 serialize하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	length	serialize된 binary data의 길이를 반환하기 위한 int형 변수의 pointer
Return	serialize된 binary data의 pointer를 반환, 에러 발생 시 NULL을 반환	

- readMe

Prototype	`void readMe(DWORD src, const char* pData, const int length)`	
Definition	사용자 컴포넌트의 속성을 serialize된 binary data로부터 읽는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	pData	속성 값을 읽어들이기 위한 binary data 저장소
	length	읽어들일 데이터의 길이

■ 기타 함수

- paint

Prototype	`void paint(DWORD src, HDC hDC, LPCTSTR data, const float x, const float y, const float w, const float h, const float scale)`	
Definition	화면에 그릴 때 호출되는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	hDC	그려질 화면의 핸들(WIN32 API 참조)
	data	라벨의 Caption
	x	컴포넌트의 x 좌표
	y	컴포넌트의 y 좌표
	w	컴포넌트의 너비
	h	컴포넌트의 높이
	scale	확대 값

- paintEX

Prototype	`void paintEX(__cdecl *paintEX)(DWORD src, HDC hDC, LPCTSTR data, const float x, const float y, const float w, const float h, const float scale const float x_offset, const float y_offset)`	
Definition	화면에 그릴 때 호출되는 함수로 실제 이미지가 저장될 offset 값을 전달할 수 있습니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	hDC	그려질 화면의 핸들(WIN32 API 참조)
	data	라벨의 Caption
	x	컴포넌트의 x 좌표
	y	컴포넌트의 y 좌표
	w	컴포넌트의 너비
	h	컴포넌트의 높이
	scale	확대 값
	x_offset	실제 이미지가 저장될 x offset
	y_offset	실제 이미지가 저장될 y offset

- print

Prototype	`void print(DWORD src, HDC hDC, LPCTSTR data, const float x, const float y, const float w, const float h, const float scale, const float x_offset, const float y_offset)`	
Definition	인쇄할 때 호출되는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	hDC	그려질 화면의 핸들(WIN32 API 참조)
	data	라벨의 Caption
	x	컴포넌트의 x 좌표
	y	컴포넌트의 y 좌표
	w	컴포넌트의 너비
	h	컴포넌트의 높이
	scale	확대 값
	x_offset	프린트 x Offset
	y_offset	프린트 x Offset

- getAutosize

Prototype	void getAutosize(DWORD src, HDC hDC, LPCTSTR data, float * w, float * h)	
Definition	자동크기조정을 사용할 때 사용자 컴포넌트의 가로, 세로 크기를 설정하는 함수입니다.	
Argument	*src*	사용자 컴포넌트 객체 핸들
	hDC	그려질 화면의 핸들(WIN32 API 참조)
	data	라벨의 Caption
	w	컴포넌트의 원래 너비의 pointer
	h	컴포넌트의 원래 높이의 pointer
Return	w, h에 자동크기 조정된 가로와 세로 설정값을 반환, 에러 발생 시 반환값 없음	

예제

다음의 예제들은 C++를 이용한 정규 dll로서 PDF 417 사용자 컴포넌트 예제입니다.

- **PDF417.h**

```
// hanja.h : main header file for the HANJA DLL
//

#if !defined(AFX_CPdf417App_H__7FDF11DA_5BF8_4C07_9BCF_BB1C08A763FA__INCLUDED_)
#define AFX_CPdf417App_H__7FDF11DA_5BF8_4C07_9BCF_BB1C08A763FA__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#ifndef __AFXWIN_H__
      #error include 'stdafx.h' before including this file for PCH
#endif

#include <resource.h>                                                    // main symbols
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////////////////
```

```
// CPdf417App
// See CPdf417App.cpp for the implementation of this class
//

class CPdf417App : public CWinApp
{
public:
   CPdf417App();

// Overrides
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CPdf417AppaApp)
//}}AFX_VIRTUAL
//{{AFX_MSG(CPdf417App)
// NOTE - the ClassWizard will add and remove member functions here.
// DO NOT EDIT what you see in these blocks of generated code !
//}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately before
the previous line.

#endif
// !defined(AFX_CPdf417App_H__7FDF11DA_5BF8_4C07_9BCF_BB1C08A763FA__INCLUDED_)
```

- **PDF417.cpp**

```
// OZPdf417Comp.cpp : Defines the initialization routines for the DLL.
//

#include <stdafx.h>
#include <pdf417.h>
#include <CUniString.h>

#include "pdfapi.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
//
// Note!
//
// If this DLL is dynamically linked against the MFC
// DLLs, any functions exported from this DLL which
// call into MFC must have the AFX_MANAGE_STATE macro
// added at the very beginning of the function.
//
// For example:
//
// extern "C" BOOL PASCAL EXPORT ExportedFunction()
// {
//   AFX_MANAGE_STATE(AfxGetStaticModuleState());
//
//   normal function body here
//}
//
//   It is very important that this macro appear in each
//   function, prior to any calls into MFC.  This means that
//   it must appear as the first statement within the
//   function, even before any object variable declarations
//   as their constructors may generate calls into the MFC
//   DLL.
//
//   Please see MFC Technical Notes 33 and 58 for additional
//   details.
//

/////////////////////////////////////////////////////////////////////////////
// CHanjaApp

BEGIN_MESSAGE_MAP(CPdf417App, CWinApp)
//{{AFX_MSG_MAP(CPdf417App)
// NOTE - the ClassWizard will add and remove mapping macros here.
// DO NOT EDIT what you see in these blocks of generated code!
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CPdf417App construction

CPdf417App::CPdf417App()
{
  // TODO: add construction code here,
  // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
```

```
// The one and only CPdf417App object

CPdf417App theApp;

class OZPdf417Comp{
public:
   OZPdf417Comp();
   OZPdf417Comp(OZPdf417Comp & cp);
   virtual ~OZPdf417Comp();
   int getAttrListLength();
   BOOL getAttrList(WCHAR** attrs, const int length);
   int getAttrLength(CString name);
   BOOL getAttr(CString name, WCHAR* value, const int value_length);
   BOOL setAttr(CString name, CString value);
   char* writeMe(int * length);
   void readMe(const char* pData, const int length);
   void paint(HDC hDC, CUniString data, const float x, const float y, const
float w, const float h, const float scale);
void print(HDC hDC, CUniString data, const float x, const float y, const float
w, const float h, const float scale, const float x_offset, const float
y_offset);
   void getAutosize(HDC hDC, CUniString data, float * w, float * h);
private:
   int rowNum;
   int colNum;
   int ecc;
   int xScale;
   int yScale;
};


//////////////////////////////////////////////////////////////////////////
#ifdef __cplusplus
extern "C"
{
#endif

__declspec( dllexport ) DWORD __cdecl GetNewInstance()
{
   return (DWORD)(void*)new OZPdf417Comp();
}
__declspec( dllexport ) DWORD __cdecl GetCopyInstance(DWORD src)
{
 return (DWORD)(void*)new OZPdf417Comp(*((OZPdf417Comp *)(void *)src));
}
__declspec( dllexport ) void __cdecl DeleteInstance(DWORD src)
{
   delete ((OZPdf417Comp *)(void *)src);
```

```
}
////////////////////////////////////////////////////////////////////////
__declspec( dllexport ) int __cdecl getAttrListLength(DWORD src)
{
   return ((OZPdf417Comp *)(void *)src)->getAttrListLength();
}
__declspec( dllexport ) BOOL __cdecl getAttrList(DWORD src, WCHAR** attrs,
const int length){
   return ((OZPdf417Comp *)(void *)src)->getAttrList(attrs, length);
}
__declspec( dllexport ) int __cdecl getAttrLength(DWORD src, const WCHAR* name)
{
  return ((OZPdf417Comp *)(void *)src)->getAttrLength(CString(name));
}
__declspec( dllexport ) BOOL __cdecl getAttr(DWORD src, const WCHAR* name,
WCHAR* value, const int value_length){
return    ((OZPdf417Comp    *)(void    *)src)->getAttr(CString(name),    value,
value_length);
}
__declspec( dllexport ) BOOL __cdecl setAttr(DWORD src, const WCHAR* name,
const WCHAR* value)
{
return ((OZPdf417Comp *)(void *)src)->setAttr(CString(name), CString(value));
}
__declspec( dllexport ) char* __cdecl writeMe(DWORD src, int * length)
{
        return ((OZPdf417Comp *)(void *)src)->writeMe(length);
}
__declspec( dllexport ) void __cdecl readMe(DWORD src, const char* pData, const
int length)
{
   ((OZPdf417Comp *)(void *)src)->readMe(pData, length);
}
__declspec( dllexport ) void __cdecl paint(DWORD src, HDC hDC, LPCWSTR data,
const float x, const float y, const float w, const float h, const float scale)
{
  ((OZPdf417Comp *)(void *)src)->paint(hDC, data, x, y, w, h, scale);
}
__declspec( dllexport ) void __cdecl print(DWORD src, HDC hDC, LPCWSTR data,
const float x, const float y, const float w, const float h, const float scale,
const float x_offset, const float y_offset)
{
((OZPdf417Comp *)(void *)src)->print(hDC, data, x, y, w, h, scale, x_offset,
y_offset);
}
__declspec( dllexport ) void __cdecl getAutosize(DWORD src, HDC hDC, LPCWSTR
data, float * w, float * h)
{
```

```
    ((OZPdf417Comp *)(void *)src)->getAutosize(hDC, data, w, h);
}
#ifdef __cplusplus
}
#endif
static PDFObject g_MacroObjIn, g_MacroObjOut;

PDFSIZET Encode(CMBString dataStr, unsigned short rowNum, unsigned short colNum,
unsigned short ecc, UINT xScale, UINT yScale)
{
    PDFSIZET nInputLen = 0;
    UINT nDPI = 300;

    PDFSetDefaults();
    PDFBinaryMode( false );
    PDFSetSymbolStyle( STYLE_NORMAL );
    PDFSetECCLevel( ecc );
    if(rowNum == 0 && colNum == 0){
    PDFSetAspect( "1:2" );
}else{
    PDFSetDimType( USE_FIXED );
    PDFSetRowCol( rowNum, colNum );
}
    CString szAspect;
    szAspect.Format(_T("%d:%d"), xScale+yScale-1, xScale );
    PDFSetRowHeight((LPCSTR)(CMBString)szAspect );

    MPDFDisable();
    nInputLen = (PDFSIZET)dataStr.GetLength();
    MakeMemoryObject( &g_MacroObjIn, PDFINPUT, (LPSTR)(LPCSTR)dataStr );
    uint16 wParam = 0;
    uint32 lParam = MakeLParam( nDPI, xScale );
    wParam |= DIB_ADDFILEHEADER;
/*
    if ( pSettings->nWShave )
        wParam |= ShaveWidthOption( pSettings->nWShave );
    if ( pSettings->nHShave )
        wParam |= ShaveHeightOption( pSettings->nHShave );
*/
    PDFOutputAsDIB( lParam, wParam );
    return nInputLen;
}
int EncodeData(CMBString dataStr, BYTE * rt, unsigned short rowNum, unsigned
short colNum, unsigned short ecc, UINT xScale, UINT yScale)
{
PDFSIZET nInputLen = Encode(dataStr, rowNum, colNum, ecc, xScale, yScale);
BOOL fStatus;
PDFSIZET rtsize = -1;
```

```
// MakeFilenameObject( &g_MacroObjOut, PDFOUTPUT,  fileStr);
   MakeMemoryObject( &g_MacroObjOut, PDFOUTPUT, (LPVOID) rt);


  if ( PDFEncode( &g_MacroObjIn, nInputLen ) >= 0 )  {
    fStatus = (PDFMakeImage( &g_MacroObjOut, &rtsize ) == 0);
  }else{
    fStatus = FALSE;
  }
  if(!fStatus){
      rtsize = -1;
      PDFAbort( &g_MacroObjIn );
      PDFAbort( &g_MacroObjOut);
  }
// printf(dataStr);
  return rtsize;
}

int EncodeFile(CMBString dataStr, CMBString fileStr, unsigned short rowNum,
unsigned short colNum, unsigned short ecc, UINT xScale, UINT yScale)
{
PDFSIZET nInputLen = Encode(dataStr, rowNum, colNum, ecc, xScale, yScale);
BOOL fStatus;
PDFSIZET rtsize = -1;
MakeFilenameObject( &g_MacroObjOut, PDFOUTPUT,  fileStr);
// MakeMemoryObject( &g_MacroObjOut, PDFOUTPUT, (LPVOID) rt);

  if ( PDFEncode( &g_MacroObjIn, nInputLen ) >= 0 )  {
    fStatus = (PDFMakeImage( &g_MacroObjOut, &rtsize ) == 0);
  }else{
    fStatus = FALSE;
  }
  if(!fStatus){
      rtsize = -1;
      PDFAbort( &g_MacroObjIn );
      PDFAbort( &g_MacroObjOut);
  }
// printf(dataStr);
// printf(fileStr);
  return rtsize;
}

LPPICTURE EncodePrint(CMBString dataStr, unsigned short rowNum, unsigned short
colNum, unsigned short ecc, UINT xScale, UINT yScale)
{
PDFSIZET nInputLen = Encode(dataStr, rowNum, colNum, ecc, xScale, yScale);
BOOL fStatus;
PDFSIZET rtsize = -1;
```

```
HGLOBAL hGlobal = NULL;
DWORD dwFileSize = 1024*10;

LPVOID pvData = NULL;
// alloc memory based on file size
hGlobal = GlobalAlloc(GMEM_MOVEABLE, dwFileSize);
if(NULL == hGlobal){
    PDFAbort( &g_MacroObjIn );
    PDFAbort( &g_MacroObjOut);
    return NULL;
}

pvData = GlobalLock(hGlobal);
if(NULL == pvData){
    PDFAbort( &g_MacroObjIn );
    PDFAbort( &g_MacroObjOut);
    return NULL;
}
    MakeMemoryObject( &g_MacroObjOut, PDFOUTPUT, pvData);

   if ( PDFEncode( &g_MacroObjIn, nInputLen ) >= 0 )  {
     fStatus = (PDFMakeImage( &g_MacroObjOut, &rtsize ) == 0);
   }else{
     fStatus = FALSE;
   }
   if(!fStatus){
       PDFAbort( &g_MacroObjIn );
       PDFAbort( &g_MacroObjOut);
       GlobalUnlock(hGlobal);
       return NULL;
   }

    GlobalUnlock(hGlobal);

    LPSTREAM pstm = NULL;
    // create IStream* from global memory
    HRESULT hr = CreateStreamOnHGlobal(hGlobal, TRUE, &pstm);
    if(!SUCCEEDED(hr)  || pstm == NULL){
       PDFAbort( &g_MacroObjIn );
       PDFAbort( &g_MacroObjOut);
       return NULL;
    }

// Create IPicture from image file
LPPICTURE gpPicture = NULL;
hr   =   ::OleLoadPicture(pstm,   dwFileSize,   FALSE,   IID_IPicture,   (LPVOID
*)&gpPicture);
```

```
    if(!SUCCEEDED(hr)  || gpPicture == NULL){
        BOOL b1 = hr == E_POINTER;
        BOOL b2 = hr == E_NOINTERFACE;
        BOOL b3 = hr == E_OUTOFMEMORY;
        BOOL b4 = hr == E_UNEXPECTED;
        pstm->Release();
        PDFAbort( &g_MacroObjIn );
        PDFAbort( &g_MacroObjOut);
        return NULL;
    }
    pstm->Release();

   return gpPicture;
}
CUniString _toString(int i){
    CString rst;
    rst.Format(_T("%d"), i);
    return (CUniString)rst;
}

////////////////////////////////////////////////////////////////////////
OZPdf417Comp::OZPdf417Comp(){
    rowNum = 0;
    colNum = 8;
    ecc = 0;
    xScale = 1;
    yScale = 1;
}
OZPdf417Comp::OZPdf417Comp(OZPdf417Comp & comp){
    rowNum = comp.rowNum;
    colNum = comp.colNum;
    ecc = comp.ecc;
    xScale = comp.xScale;
    yScale = comp.yScale;
}
OZPdf417Comp::~OZPdf417Comp(){
}
int OZPdf417Comp::getAttrListLength(){
    return 5;
}
BOOL OZPdf417Comp::getAttrList(WCHAR** attrs, const int length){
    if(length < 0)return FALSE;
    int nIndex = 0;
    if(length == nIndex)return TRUE;
    attrs[nIndex++] = L"PDF417_ROWNUM";
    if(length == nIndex)return TRUE;
    attrs[nIndex++] = L"PDF417_COLNUM";
```

```
    if(length == nIndex)return TRUE;
    attrs[nIndex++] = L"PDF417_ECC";
    if(length == nIndex)return TRUE;
    attrs[nIndex++] = L"PDF417_XSCALE";
    if(length == nIndex)return TRUE;
    attrs[nIndex++] = L"PDF417_YSCALE";
    return TRUE;
}

int OZPdf417Comp::getAttrLength(CString name){
    CUniString value;
    if(name == _T("PDF417_ROWNUM")){
        value = _toString(rowNum);
    }else if(name == _T("PDF417_COLNUM")){
        value = _toString(colNum);
    }else if(name == _T("PDF417_ECC")){
        value = _toString(ecc);
    }else if(name == _T("PDF417_XSCALE")){
        value = _toString(xScale);
    }else if(name == _T("PDF417_YSCALE")){
        value = _toString(yScale);
////////////////////////////////////////////////////
    }else{
        return -1;
}
    return value.GetLength()+1;
}

BOOL   OZPdf417Comp::getAttr(CString   name,   WCHAR*   valueBuffer,   const   int
value_length){
    CUniString value;
    if(name == _T("PDF417_ROWNUM")){
        value = _toString(rowNum);
    }else if(name == _T("PDF417_COLNUM")){
        value = _toString(colNum);
    }else if(name == _T("PDF417_ECC")){
        value = _toString(ecc);
    }else if(name == _T("PDF417_XSCALE")){
        value = _toString(xScale);
    }else if(name == _T("PDF417_YSCALE")){
        value = _toString(yScale);
////////////////////////////////////////////////////
    }else{
        return FALSE;
    }
    if(value.GetLength() >= value_length){
        return FALSE;
    }
```

```
    wcscpy(valueBuffer, value);
    return TRUE;
}
BOOL OZPdf417Comp::setAttr(CString name, CString value){
    if(name == _T("PDF417_ROWNUM")){
        rowNum = _ttoi(value);
    }else if(name == _T("PDF417_COLNUM")){
        colNum = _ttoi(value);
    }else if(name == _T("PDF417_ECC")){
        ecc = _ttoi(value);
    }else if(name == _T("PDF417_XSCALE")){
        xScale = _ttoi(value);
    }else if(name == _T("PDF417_YSCALE")){
        yScale = _ttoi(value);
    }else{
        return FALSE;
    }
    return TRUE;
}
char* OZPdf417Comp::writeMe(int * length){
    *length = 20;
    char* pData = new char[*length];
    int i = 0;
    pData[i++] = (char)((rowNum & 0xFF000000) >> 24);
    pData[i++] = (char)((rowNum & 0x00FF0000) >> 16);
    pData[i++] = (char)((rowNum & 0x0000FF00) >> 8);
    pData[i++] = (char)((rowNum & 0x000000FF));

    pData[i++] = (char)((colNum & 0xFF000000) >> 24);
    pData[i++] = (char)((colNum & 0x00FF0000) >> 16);
    pData[i++] = (char)((colNum & 0x0000FF00) >> 8);
    pData[i++] = (char)((colNum & 0x000000FF));

    pData[i++] = (char)((ecc & 0xFF000000) >> 24);
    pData[i++] = (char)((ecc & 0x00FF0000) >> 16);
    pData[i++] = (char)((ecc & 0x0000FF00) >> 8);
    pData[i++] = (char)((ecc & 0x000000FF));

    pData[i++] = (char)((xScale & 0xFF000000) >> 24);
    pData[i++] = (char)((xScale & 0x00FF0000) >> 16);
    pData[i++] = (char)((xScale & 0x0000FF00) >> 8);
    pData[i++] = (char)((xScale & 0x000000FF));

    pData[i++] = (char)((yScale & 0xFF000000) >> 24);
    pData[i++] = (char)((yScale & 0x00FF0000) >> 16);
    pData[i++] = (char)((yScale & 0x0000FF00) >> 8);
    pData[i++] = (char)((yScale & 0x000000FF));
```

```
    return pData;
}
void OZPdf417Comp::readMe(const char* pData, const int length){
    if(length != 20){
        AfxThrowUserException();
    }
    int i = 0;
    rowNum = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
(pData[i++] << 0));
    colNum = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
(pData[i++] << 0));
ecc = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
(pData[i++] << 0));
    xScale = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
(pData[i++] << 0));
    yScale = ((pData[i++] << 24) + (pData[i++] << 16) + (pData[i++] << 8) +
(pData[i++] << 0));
}
void OZPdf417Comp::paint(HDC hDC, CUniString data, const float x, const float y,
const float w, const float h, const float scale){
    LPPICTURE gpPicture = EncodePrint((CMBString)data,
    rowNum,colNum,ecc,xScale,yScale);//함수호출
    if(gpPicture == NULL)return;
    long hmWidth;
    long hmHeight;
    gpPicture->get_Width(&hmWidth);
    gpPicture->get_Height(&hmHeight);
    // convert himetric to pixelsA
    int imageWidth;
    int imageHeight;
    imageWidth= MulDiv(hmWidth*72, GetDeviceCaps(hDC, LOGPIXELSX), 2540*96);
    imageHeight= MulDiv(hmHeight*72, GetDeviceCaps(hDC, LOGPIXELSY), 2540*96);
    imageWidth = (int)(imageWidth*scale);
    imageHeight = (int)(imageHeight*scale);
    RECT rc = { 0, 0, imageWidth, imageHeight };
    HRESULT hesult = gpPicture->Render(hDC, (int)(x*scale), (int)(y*scale) ,
imageWidth, imageHeight, 0, hmHeight, hmWidth, -hmHeight, &rc);

    gpPicture->Release();
}
void OZPdf417Comp::print(HDC hDC, CUniString data, const float x, const float y,
const float w, const float h, const float scale, const float x_offset, const
float y_offset){
    LPPICTURE gpPicture = EncodePrint((CMBString)data,
        rowNum,colNum,ecc,xScale,yScale);//함수호출
    if(gpPicture == NULL)return;
    long hmWidth;
    long hmHeight;
```

```
   gpPicture->get_Width(&hmWidth);
   gpPicture->get_Height(&hmHeight);
   // convert himetric to pixelsA
   int imageWidth;
   int imageHeight;
   imageWidth= MulDiv(hmWidth, GetDeviceCaps(hDC, LOGPIXELSX), 2540);
   imageHeight= MulDiv(hmHeight, GetDeviceCaps(hDC, LOGPIXELSY), 2540);

   RECT rc = { 0, 0, imageWidth, imageHeight };
HRESULT     hesult     =     gpPicture->Render(hDC,     (int)(x*scale+x_offset),
(int)(y*scale+y_offset)  ,  imageWidth,  imageHeight,  0,  hmHeight,  hmWidth,  -
hmHeight, &rc);

   gpPicture->Release();
}
void OZPdf417Comp::getAutosize(HDC hDC, CUniString data, float * w, float * h){
   LPPICTURE gpPicture = EncodePrint((CMBString)data,
      rowNum,colNum,ecc,xScale,yScale);//함수호출
   if(gpPicture == NULL)return;
   long hmWidth;
   long hmHeight;
   gpPicture->get_Width(&hmWidth);
   gpPicture->get_Height(&hmHeight);
   // convert himetric to pixelsA
   *w= (float)MulDiv(hmWidth*72, GetDeviceCaps(hDC, LOGPIXELSX), 2540*96);
   *h= (float)MulDiv(hmHeight*72, GetDeviceCaps(hDC, LOGPIXELSY), 2540*96);
   gpPicture->Release();
}
```

Appendix 1. SchedulerCom 활용

본 장에서는 **Scheduler API** 중 **makePDF** 함수와 **export** 함수를 **ASP**에서 사용할 수 있도록 하는 **SchedulerCom**의 함수와 **makePDF**와 **export** 함수를 사용하는 **ASP** 샘플에 대해 설명합니다.

SchedulerCom 함수

■ **Com** 초기화 및 해제 함수

- .Init

Prototype	`.Init()`		
Definition	COM을 초기화합니다.		

- .Clean

Prototype	`.Clean()`		
Definition	COM의 설정 정보를 해제시킵니다.		

■ 오즈 서버 설정 함수

- .SetServerType

Prototype	`.SetServerType String ServerType`		
Definition	오즈 서버의 타입을 설정합니다.		
Argument	*ServerType*	*"TCP"*	오즈 서버가 데몬 타입일 경우 "TCP"로 설정
		"Servlet"	오즈 서버가 서블릿 타입일 경우 "Servlet"으로 설정 (기본값)

- .SetServerIP

Prototype	`.SetServerIP String IP`	
Definition	오즈 서버가 데몬 타입일 경우 오즈 서버의 IP를 설정합니다.	
Argument	*IP*	오즈 서버의 IP ex) "127.0.0.1"

- .SetServerPort

Prototype	`.SetServerPort String Port`	
Definition	오즈 서버가 데몬 타입일 경우 오즈 서버의 Port 번호를 설정합니다.	
Argument	*Port*	오즈 서버의 Port 번호 ex) "8003"

- .SetServerURL

Prototype	`.SetServerURL String URL`	
Definition	오즈 서버가 서블릿 타입일 경우 오즈 서버의 URL을 설정합니다.	
Argument	*URL*	오즈 서버 URL ex) "http://localhost:7001/oz/server"

■ 오즈 스케줄러 서버 설정 함수

- .SetSchedulerIP

Prototype	`.SetSchedulerIP String IP`	
Definition	오즈 스케줄러 서버의 IP를 설정합니다.	
Argument	*IP*	오즈 스케줄러 서버의 IP ex) "127.0.0.1"

- .SetSchedulerPort

Prototype	`.SetSchedulerPort String Port`	
Definition	오즈 스케줄러 서버의 Port 번호를 설정합니다.	
Argument	*Port*	오즈 스케줄러 서버의 Port 번호 ex) "9521"

■ 사용자 정보 설정 함수

- .SetUser

Prototype	`.SetUser String UserID`	
Definition	접속할 사용자의 아이디를 설정합니다.	
Argument	*UserID*	사용자 아이디 ex) "admin"

- .SetPassword

Prototype	`.SetPassword String Password`	
Definition	접속할 사용자의 패스워드를 설정합니다.	
Argument	*Password*	사용자 패스워드 ex) "admin"

■ **Key 설정 함수**

- .SetProperty

Prototype	`.SetProperty String Key, String Value`

Definition	보고서 파일명, 패러미터 등을 설정합니다. ※ 주의사항 : ".SetProperty" 함수를 이용하여 설정되는 Key 중 "launch_type", "report_name", "category_name", "export.confirmsave", "parameter_count"은 반드시 설정하여야 제대로 동작됩니다.	
Argument	*Key*	설정할 Key
	Value	설정할 값

※ 참고사항 : ".SetProperty"에서 사용할 수 있는 Key와 Value

Key	Value	설명
"launch_type"	"Immediately"	Scheduler에서 Task로 인식하기 위해 필요하며, 값은 항상 "Immediately"로 설정하여야 합니다.
"report_name"	"파일명"	익스포트할 보고서(*.ozr) 파일명을 설정합니다.
"category_name"	"/<Category>"	익스포트할 보고서의 파일 카테고리를 설정합니다.
"export.confirmsave"	"false"	항상 "false"로 설정합니다.
"parameter_count"	"패러미터 개수"	폼 패러미터 개수와 ODI 패러미터 개수를 더한 전체 패러미터 개수를 설정합니다. ※ 주의사항 : 패러미터가 없을 경우 "0"으로 설정합니다.
"parameter_name_<index>"	"패러미터명"	패러미터명을 설정합니다. 폼 패러미터의 경우에는 "[FORM].패러미터명"으로 설정하고 ODI 패러미터의 경우에는 "ODI명.패러미터명"으로 설정합니다.
"parameter_value_<index>"	"패러미터 값"	패러미터 값을 설정합니다.

- .SetExportProperty

Prototype	`.SetExportProperty String Key, String Value`

Definition	익스포트할 보고서 정보 또는 익스포트 파일 등 에 관한 정보를 설정합니다.	
Argument	*Key*	설정할 Key
	Value	설정할 값

※ 참고사항 : ".SetExportProperty"에서 사용할 수 있는 Key와 Value

Key	Value	설명
"connection.server"	"서버 IP"	오즈 서버가 데몬 타입일 경우 오즈 서버의 IP를 설정합니다.
"connection.port"	"서버 Port"	오즈 서버가 데몬 타입일 경우 오즈 서버의 Port 번호를 설정합니다.
"connection.servlet"	"서버 URL"	오즈 서버가 서블릿 타입일 경우 오즈 서버의 URL을 설정합니다.
"connection.reportName"	"보고서 파일명"	익스포트할 보고서의 파일명을 설정합니다.
"connection.fetchtype"	"BATCH"	패치 타입을 설정합니다.
"connection.pcount"	"패러미터 개수"	폼 패러미터 개수를 설정합니다.
"connection.args<index>=패러미터명"	"패러미터 값"	폼 패러미터 값을 설정합니다.
"odi.odinames"	"ODI명,ODI명,.."	사용되는 ODI명을 설정합니다. 여러 개의 ODI인 경우 각각의 ODI명을 콤마(",")로 구별하여 나열합니다.
"odi.ODI명.pcount",	"패러미터 개수"	ODI 패러미터 개수를 설정합니다.
"odi.ODI명.args<index>"	"패러미터명=패러미터값"	패러미터명과 패러미터 값을 설정합니다.
"export.format"	"익스포트할 파일 확장자"	익스포트할 파일의 확장자를 "/"로 구분하여 나열합니다. "ozd/html/jpg/xls/doc/svg/txt/ppt/tif/csv"

"<ozd/html/jpg/xls/doc/svg/ txt/ppt/tif/csv>.filename"	"파일명"	익스포트할 파일명을 설정합니다. ex).setExportProperty "ozd.filename", "test.ozd"
"viewer.childcount"	"자식 개수"	멀티 보고서의 경우 자식 보고서의 개수를 설정합니다.

※ 참고사항 : 자식 보고서의 경우 부모 보고서에서 설정할 수 있는 Key에 "child<index>."을 조합하여 사용할 수 있습니다.

ex) "child1.connection.server"

ex) "child1.connection.port"

■ 익스포트 함수

- .MakePDF

Prototype	`.MakePDF(String ExportType)`		
Definition	pdf 파일로 익스포트합니다. ※ 주의사항 : 서버에 설정된 스케줄러의 환경 설정 중 "ViewType"을 "None"으로 설정한 경우에는 사용할 수 없습니다.		
Argument	*ExportType*	*"SHOW"*	사용자 PC에 설치되어 있는 PDF Reader에 익스포트한 pdf 파일을 보여줍니다.
		"ATTACH"	사용자가 pdf 파일을 다운로드 받을 수 있는 창이 뜹니다.
		"NONE"	pdf 파일을 익스포트만 하고, 사용자에게는 아무런 액션을 보내지 않습니다.

- .Export

Prototype	`.Export()`
Definition	pdf, ozd, html, jpg, xls, doc, svg, txt, ppt, tif, csv 파일로 익스포트합니다. ※ 주의사항 : 서버에 설정된 스케줄러의 환경 설정 중 "ViewType"을 "None"으로 설정한 경우에는 사용할 수 없습니다.

- .IsExportSuccessed

Prototype	`.IsExportSuccessed()`	
Definition	익스포트 결과를 리턴합니다.	
Return	*true*	익스포트 성공

	false	익스포트 실패

■ 기타 함수

- .ShowMessage

Prototype	.ShowMessage(String Msg)	
Definition	설정한 메시지를 표시합니다.	
Argument	*Msg*	표시할 메시지

SchedulerCom 주의사항

■ 사용 가능 서버

SchedulerCom은 MS 계열 서버에서만 사용할 수 있습니다.

■ **SchedulerCOM.dll 파일 레지스트리에 등록**

"SchedulerCom"을 사용하기 위해서는 "SchedulerCOM.dll" 파일을 서버의 레지스트리에 등록하여야 합니다.

만일 "SchedulerCOM.dll" 파일이 "C:\OZServer\Scheduler\"에 있을 경우 실행 창에

```
regsvr32  " C:\OZServer\Scheduler\SchedulerCOM.dll"
```

로 입력하여 등록합니다.

■ 익스포트 파일 경로

MakePDF 또는 Export 함수를 이용하여 보고서 파일을 익스포트하면 익스포트된 파일의 이름은 설정할 수 있으나 익스포트 경로는 패러미터로 설정할 수 없고, 스케줄러에 설정되어 있는 리파지토리 경로로 자동 지정됩니다. 스케줄러에 설정되어 있는 리파지토리 경로는 "scheduler_server.properties"의 "RepositoryFileRootPath"에 설정된 값을 참조하시기 바랍니다.

MakePDF 함수 사용 ASP 샘플

```
<%
Dim Com
Set Com = Server.CreateObject("SchedulerCOM.CSchedulerCall.1")
```

```
If Not IsObject (Com) Then

   Response.Write("PDF File이 정상적으로 생성되지 않았습니다.")
   Response.End

Else

   With Com

      'Init. COM
         .Init()
      ' TCP-Daemon Type Server
         .SetServerType "TCP"
         .SetServerIP "127.0.0.1"
         .SetServerPort "8003"

      ' Servlet Type Server
      '   .SetServerType "Servlet"
      '   .SetServerURL "http://localhost:7001/oz/server"

      ' Set Scheduler Info.
         .SetSchedulerIP "127.0.0.1"
         .SetSchedulerPort "9521"

      ' set User Info.
         .SetUser "admin"
         .SetPassword "admin"

      ' set Launch Type
         .SetProperty "launch_type", "Immediately"

      ' set Report Info.
         .SetProperty "report_name", "parameter_test.ozr"
         .SetProperty "category_name", "/"
         .SetProperty "export.confirmsave", "false"

      ' with NO parameter.
         .setProperty "parameter_count", "0"

      ' with parameters.
      ' .setProperty "parameter_count", "4"
      ' .setProperty "parameter_name_1", "[FORM].formparam1"
      ' .setProperty "parameter_value_1", "PARAM 1"
```

```
      ' .setProperty "parameter_name_2", "[FORM].formparam2"
      ' .setProperty "parameter_value_2", "PARAM 2"
      ' .setProperty "parameter_name_3", "parameter_test.odiparam1"
      ' .setProperty "parameter_value_3", "PARAM 3"
      ' .setProperty "parameter_name_4", "parameter_test.odiparam2"
      ' .setProperty "parameter_value_4", "PARAM 4"

       ' set Export Info.
          .setExportProperty "pdf.filename", "PDF_TEST.pdf"

      .MakePDF("NONE")

      res = .IsExportSuccessed()
      if res = "true" then
         .ShowMessage ("successed...")
      else
         .ShowMessage ("failed...")
      end if

      .Clean()

   End With

   Set Com = Nothing

End If
%>
```

Export 함수 사용 ASP 샘플

```
<%
Dim Com
Set Com = Server.CreateObject("SchedulerCOM.CSchedulerCall.1")

If Not IsObject (Com) Then
   Response.Write("File이 정상적으로 생성되지 않았습니다.")
   Response.End

Else
   With Com
      .Init()
      .SetServerType "TCP"
```

```
.SetServerIP "127.0.0.1"
.SetServerPort "8003"
.SetSchedulerIP "127.0.0.1"
.SetSchedulerPort "9521"
.SetUser "admin"
.SetPassword "admin"
.SetProperty "launch_type", "Immediately"
.setProperty "cfg.type","new"

.setExportProperty "connection.server", "127.0.0.1"
.setExportProperty "connection.port", "8003"
.setExportProperty "connection.reportName", "/parameter_test.ozr"
.setExportProperty "connection.fetchtype", "BATCH"
.setExportProperty "connection.pcount", "2"
.setExportProperty "connection.args1=formparam1", "form1"
.setExportProperty "connection.args2=formparam2", "form2"

.setExportProperty "odi.parameter_test.args1", "odiparam1=odi1"
.setExportProperty "odi.parameter_test.args2", "odiparam2=odi2"
.setExportProperty "odi.parameter_test.pcount", "2"
.setExportProperty "odi.odinames", "parameter_test"

.setExportProperty "export.format",
                    "ozd/html/jpg/xls/doc/svg/txt/ppt/tif/csv"
.setExportProperty "ozd.filename", "test.ozd"
.setExportProperty "html.filename", "test.html"
.setExportProperty "jpg.filename", "test.jpg"
.setExportProperty "excel.filename", "test.xls"
.setExportProperty "word.filename", "test.doc"
.setExportProperty "svg.filename", "test.svg"
.setExportProperty "text.filename", "test.txt"
.setExportProperty "ppt.filename", "test.ppt"
.setExportProperty "tiff.filename", "test.tif"
.setExportProperty "csv.filename", "test.csv"

.setExportProperty "viewer.childcount", "1"
.setExportProperty "child1.connection.server", "127.0.0.1"
.setExportProperty "child1.connection.port", "8003"
.setExportProperty "child1.connection.reportName",
                   "/parameter_test.ozr"
.setExportProperty "child1.connection.fetchtype", "BATCH"
.setExportProperty "child1.connection.pcount", "2"
.setExportProperty "child1.connection.args1=formparam1", "form1"
.setExportProperty "child1.connection.args2=formparam2", "form2"
```

```
      .setExportProperty "child1.odi.parameter_test.args1", "odiparam1=odi1"
      .setExportProperty "child1.odi.parameter_test.args2", "odiparam2=odi2"
      .setExportProperty "child1.odi.parameter_test.pcount", "2"
      .setExportProperty "child1.odi.odinames", "parameter_test"

      .setExportProperty "child1.export.format",
                         "ozd/html/jpg/xls/doc/svg/txt/ppt/tif/csv"
      .setExportProperty "child1.ozd.filename", "child_test.ozd"
      .setExportProperty "child1.html.filename", "child_test.html"
      .setExportProperty "child1.jpg.filename", "child_test.jpg"
      .setExportProperty "child1.excel.filename", "child_test.xls"
      .setExportProperty "child1.word.filename", "child_test.doc"
      .setExportProperty "child1.svg.filename", "child_test.svg"
      .setExportProperty "child1.text.filename", "child_test.txt"
      .setExportProperty "child1.ppt.filename", "child_test.ppt"
      .setExportProperty "child1.tiff.filename", "child_test.tif"
      .setExportProperty "child1.csv.filename", "child_test.csv"

      .Export()

      Response.Write(.IsExportSuccessed())

      .Clean()

   End With

   Set Com = Nothing

End If
%>
```