```
// ============================================================================
//
// This source code has been automatically generated by_Talend Integration Suite Professional Edition
// / JobDesigner (CodeGenerator version 4.2.2.r63143)
// You can find more information about Talend products at www.talend.com.
//
// ============================================================================
package test.test1_0_1;
import routines.Mathematical;
import routines.DataOperation;
import routines.Relational;
import routines.TalendDate;
import routines.TalendDataGenerator;
import routines.Numeric;
import routines.SQLike;
import routines.TalendString;
import routines.StringHandling;
import routines.system.*;
import routines.system.api.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.math.BigDecimal;
import java.io.ByteArrayOutputStream;
import java.io.ByteArrayInputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.ObjectOutputStream;
import java.io.ObjectInputStream;
import java.io.IOException;
import java.util.Comparator;
/**
* Job: test1 Purpose: <br>
* Description: <br>
*
* @author Munirathinam, Sai
* @version 4.2.2.r63143
* @status
*/
public class test1 implements TalendJob {
public final Object obj = new Object();
// for transmiting parameters purpose
private Object valueObject = null;
public Object getValueObject() {
return this.valueObject;
}
public void setValueObject(Object valueObject) {
this.valueObject = valueObject;
}
private final static String defaultCharset = java.nio.charset.Charset
.defaultCharset().name();
private final static String utf8Charset = "UTF-8";
```

```
// create and load default properties
private java.util.Properties defaultProps = new java.util.Properties();
// create application properties with default
public class ContextProperties extends java.util.Properties {
public ContextProperties(java.util.Properties properties) {
super(properties);
}
public ContextProperties() {
super();
}
public void synchronizeContext() {
}
}
private ContextProperties context = new ContextProperties();
public ContextProperties getContext() {
return this.context;
}
private final String jobVersion = "0.1";
private final String jobName = "test1";
private final String projectName = "TEST";
public Integer errorCode = null;
private String currentComponent = "";
private final java.util.Map<String, Long> start_Hash = new java.util.HashMap<String, Long>();
private final java.util.Map<String, Long> end_Hash = new java.util.HashMap<String, Long>();
private final java.util.Map<String, Boolean> ok_Hash = new java.util.HashMap<String, Boolean>();
private final java.util.Map<String, Object> globalMap = new java.util.HashMap<String, Object>();
public final java.util.List<String[]> globalBuffer = new java.util.ArrayList<String[]>();
public boolean isExportedAsOSGI = false;
private final java.io.ByteArrayOutputStream baos = new java.io.ByteArrayOutputStream();
private final java.io.PrintStream errorMessagePS = new java.io.PrintStream(
new java.io.BufferedOutputStream(baos));
public String getExceptionStackTrace() {
if ("failure".equals(this.getStatus())) {
errorMessagePS.flush();
return baos.toString();
}
return null;
}
private Exception exception = null;
public Exception getException() {
if ("failure".equals(this.getStatus())) {
return this.exception;
}
return null;
}
private class TalendException extends Exception {
private java.util.Map<String, Object> globalMap = null;
private Exception e = null;
private String currentComponent = null;
private TalendException(Exception e, String errorComponent,
final java.util.Map<String, Object> globalMap) {
this.currentComponent = errorComponent;
this.globalMap = globalMap;
```

```
this.e = e;
}
@Override
public void printStackTrace() {
if (!(e instanceof TalendException || e instanceof TDieException)) {
globalMap.put(currentComponent + "_ERROR_MESSAGE", e
.getMessage());
System.err
.println("Exception in component " + currentComponent);
}
if (!(e instanceof TDieException)) {
if (e instanceof TalendException) {
e.printStackTrace();
} else {
e.printStackTrace();
e.printStackTrace(errorMessagePS);
test1.this.exception = e;
}
}
if (!(e instanceof TalendException)) {
try {
for (java.lang.reflect.Method m : this.getClass()
.getEnclosingClass().getMethods()) {
if (m.getName().compareTo(currentComponent + "_error") == 0) {
m.invoke(test1.this, new Object[] { e,
currentComponent, globalMap });
break;
}
}
if (!(e instanceof TDieException)) {
}
} catch (java.lang.SecurityException e) {
this.e.printStackTrace();
} catch (java.lang.IllegalArgumentException e) {
this.e.printStackTrace();
} catch (java.lang.IllegalAccessException e) {
this.e.printStackTrace();
} catch (java.lang.reflect.InvocationTargetException e) {
this.e.printStackTrace();
}
}
}
}
public void tRowGenerator_1_error(Exception exception,
String errorComponent, final java.util.Map<String, Object> globalMap)
throws TalendException {
end_Hash.put("tRowGenerator_1", System.currentTimeMillis());
tRowGenerator_1_onSubJobError(exception, errorComponent, globalMap);
}
public void tScramble_1_error(Exception exception, String errorComponent,
final java.util.Map<String, Object> globalMap)
throws TalendException {
end_Hash.put("tScramble_1", System.currentTimeMillis());
```

```
tRowGenerator_1_onSubJobError(exception, errorComponent, globalMap);
}
public void tLogRow_1_error(Exception exception, String errorComponent,
final java.util.Map<String, Object> globalMap)
throws TalendException {
end_Hash.put("tLogRow_1", System.currentTimeMillis());
tRowGenerator_1_onSubJobError(exception, errorComponent, globalMap);
}
public void tRowGenerator_1_onSubJobError(Exception exception,
String errorComponent, final java.util.Map<String, Object> globalMap)
throws TalendException {
resumeUtil.addLog("SYSTEM_LOG", "NODE:" + errorComponent, "", Thread
.currentThread().getId()
+ "", "FATAL", "", exception.getMessage(), ResumeUtil
.getExceptionStackTrace(exception), "");
}
interface ESBProviderCallbackTalendJobInner extends ESBProviderCallback {
void sendFault(Throwable e);
void sendBusinessFault(String faultString,
org.dom4j.Document faultDetail);
}
public static class row2Struct implements
routines.system.IPersistableRow<row2Struct> {
final static byte[] commonByteArrayLock = new byte;
static byte[] commonByteArray = new byte;
public Integer ssn;
public Integer getSsn() {
return this.ssn;
}
private int readInteger(ObjectInputStream dis) throws IOException {
Integer intReturn;
int length = 0;
length = dis.readByte();
if (length == -1) {
intReturn = null;
} else {
intReturn = dis.readInt();
}
return intReturn;
}
private void writeInteger(Integer intNum, ObjectOutputStream dos)
throws IOException {
if (intNum == null) {
dos.writeByte(-1);
} else {
dos.writeByte(0);
dos.writeInt(intNum);
}
}
public void readData(ObjectInputStream dis) {
synchronized (commonByteArrayLock) {
try {
int length = 0;
```

```
this.ssn = readInteger(dis);
} catch (IOException e) {
throw new RuntimeException(e);
}
}
}
public void writeData(ObjectOutputStream dos) {
try {
// Integer
writeInteger(this.ssn, dos);
} catch (IOException e) {
throw new RuntimeException(e);
}
}
public String toString() {
StringBuilder sb = new StringBuilder();
sb.append(super.toString());
sb.append("");
return sb.toString();
}
/**
* Compare keys
*/
public int compareTo(row2Struct other) {
int returnValue = -1;
return returnValue;
}
private int checkNullsAndCompare(Object object1, Object object2) {
int returnValue = 0;
if (object1 instanceof Comparable && object2 instanceof Comparable) {
returnValue = ((Comparable) object1).compareTo(object2);
} else if (object1 != null && object2 != null) {
returnValue = compareStrings(object1.toString(), object2
.toString());
} else if (object1 == null && object2 != null) {
returnValue = 1;
} else if (object1 != null && object2 == null) {
returnValue = -1;
} else {
returnValue = 0;
}
return returnValue;
}
private int compareStrings(String string1, String string2) {
return string1.compareTo(string2);
}
}
public static class row1Struct implements
routines.system.IPersistableRow<row1Struct> {
final static byte[] commonByteArrayLock = new byte;
static byte[] commonByteArray = new byte;
public Integer ssn;
public Integer getSsn() {
```

```
return this.ssn;
}
private int readInteger(ObjectInputStream dis) throws IOException {
Integer intReturn;
int length = 0;
length = dis.readByte();
if (length == -1) {
intReturn = null;
} else {
intReturn = dis.readInt();
}
return intReturn;
}
private void writeInteger(Integer intNum, ObjectOutputStream dos)
throws IOException {
if (intNum == null) {
dos.writeByte(-1);
} else {
dos.writeByte(0);
dos.writeInt(intNum);
}
}
public void readData(ObjectInputStream dis) {
synchronized (commonByteArrayLock) {
try {
int length = 0;
this.ssn = readInteger(dis);
} catch (IOException e) {
throw new RuntimeException(e);
}
}
}
public void writeData(ObjectOutputStream dos) {
try {
// Integer
writeInteger(this.ssn, dos);
} catch (IOException e) {
throw new RuntimeException(e);
}
}
public String toString() {
StringBuilder sb = new StringBuilder();
sb.append(super.toString());
sb.append("");
return sb.toString();
}
/**
* Compare keys
*/
public int compareTo(row1Struct other) {
int returnValue = -1;
return returnValue;
}
```

```
private int checkNullsAndCompare(Object object1, Object object2) {
int returnValue = 0;
if (object1 instanceof Comparable && object2 instanceof Comparable) {
returnValue = ((Comparable) object1).compareTo(object2);
} else if (object1 != null && object2 != null) {
returnValue = compareStrings(object1.toString(), object2
.toString());
} else if (object1 == null && object2 != null) {
returnValue = 1;
} else if (object1 != null && object2 == null) {
returnValue = -1;
} else {
returnValue = 0;
}
return returnValue;
}
private int compareStrings(String string1, String string2) {
return string1.compareTo(string2);
}
}
public void tRowGenerator_1Process(
final java.util.Map<String, Object> globalMap)
throws TalendException {
globalMap.put("tRowGenerator_1_SUBPROCESS_STATE", 0);
final boolean execStat = this.execStat;
String iterateId = "";
String currentComponent = "";
try {
String currentMethodName = new Exception().getStackTrace()
.getMethodName();
boolean resumeIt = currentMethodName.equals(resumeEntryMethodName);
if (resumeEntryMethodName == null || resumeIt || globalResumeTicket) {// start
// the
// resume
globalResumeTicket = true;
row1Struct row1 = new row1Struct();
row1Struct row2 = row1;
/**
* start
*/
ok_Hash.put("tLogRow_1", false);
start_Hash.put("tLogRow_1", System.currentTimeMillis());
currentComponent = "tLogRow_1";
int tos_count_tLogRow_1 = 0;
// ////////////////////
final String OUTPUT_FIELD_SEPARATOR_tLogRow_1 = "|";
java.io.PrintStream consoleOut_tLogRow_1 = null;
int nb_line_tLogRow_1 = 0;
// ////////////////////
/**
* stop
*/
/**
```

```
* start
*/
ok_Hash.put("tScramble_1", false);
start_Hash.put("tScramble_1", System.currentTimeMillis());
currentComponent = "tScramble_1";
int tos_count_tScramble_1 = 0;
/**
* stop
*/
/**
* start
*/
ok_Hash.put("tRowGenerator_1", false);
start_Hash.put("tRowGenerator_1", System.currentTimeMillis());
currentComponent = "tRowGenerator_1";
int tos_count_tRowGenerator_1 = 0;
int nb_line_tRowGenerator_1 = 0;
int nb_max_row_tRowGenerator_1 = 100;
class tRowGenerator_1Randomizer {
public Integer getRandomssn() {
return Numeric.sequence("s1", 111111111, 1);
}
}
tRowGenerator_1Randomizer randtRowGenerator_1 = new tRowGenerator_1Randomizer();
for (int itRowGenerator_1 = 0; itRowGenerator_1 < nb_max_row_tRowGenerator_1; itRowGenerator_1++) {
row1.ssn = randtRowGenerator_1.getRandomssn();
nb_line_tRowGenerator_1++;
/**
* stop
*/
/**
* start
*/
currentComponent = "tRowGenerator_1";
tos_count_tRowGenerator_1++;
/**
* stop
*/
/**
* start
*/
currentComponent = "tScramble_1";
javax.crypto.Cipher eciphertScramble_1;
javax.crypto.Cipher dciphertScramble_1;
// 8-byte Salt
byte[] salttScramble_1 = { (byte) 0xA9, (byte) 0x9B,
(byte) 0xC8, (byte) 0x32, (byte) 0x56, (byte) 0x35,
(byte) 0xE3, (byte) 0x03 };
byte[] saltaestScramble_1 = { (byte) 0xA9, (byte) 0x9B,
(byte) 0xC8, (byte) 0x32, (byte) 0x56, (byte) 0x35,
(byte) 0xE3, (byte) 0x03 };
// Iteration count
int iterationCounttScramble_1 = 19;
```

```
java.security.spec.AlgorithmParameterSpec paramSpectScramble_1 = new javax.crypto.spec.PBEParameterSpec(
salttScramble_1, iterationCounttScramble_1);
java.security.spec.KeySpec keySpectScramble_1;
javax.crypto.SecretKey keytScramble_1;
javax.crypto.KeyGenerator kgentScramble_1 = javax.crypto.KeyGenerator
.getInstance("AES");
keySpectScramble_1 = kgentScramble_1.generateKey();
saltaestScramble_1 = skey.getEncoded();
javax.crypto.spec.SecretKeySpec skeySpectScramble_1 = new SecretKeySpec(
saltaestScramble_1, "AES");
String resulttScramble_1 = "";
byte[] utf8tScramble_1;
byte[] enctScramble_1;
byte[] dectScramble_1;
kgen.init(128);
if ("AES".equals("DES")) {
keySpectScramble_1 = new javax.crypto.spec.PBEKeySpec(
"SocialSecurity".toCharArray(),
salttScramble_1, iterationCounttScramble_1);
keytScramble_1 = javax.crypto.SecretKeyFactory
.getInstance("PBEWithMD5AndDES")
.generateSecret(keySpectScramble_1);
if ("Encrypt".equals("Encrypt")) {
// Encode the string into bytes using utf-8
utf8tScramble_1 = row1.ssn.getBytes("UTF8");
// Encrypt
eciphertScramble_1 = javax.crypto.Cipher
.getInstance(keytScramble_1.getAlgorithm());
// Create the ciphers
eciphertScramble_1.init(
javax.crypto.Cipher.ENCRYPT_MODE,
keytScramble_1, paramSpectScramble_1);
enctScramble_1 = eciphertScramble_1
.doFinal(utf8tScramble_1);
// Encode bytes to base64 to get a string
resulttScramble_1 = new sun.misc.BASE64Encoder()
.encode(enctScramble_1);
} else {
dectScramble_1 = new sun.misc.BASE64Decoder()
.decodeBuffer(row1.ssn);
dciphertScramble_1 = javax.crypto.Cipher
.getInstance(keytScramble_1.getAlgorithm());
dciphertScramble_1.init(
javax.crypto.Cipher.DECRYPT_MODE,
keytScramble_1, paramSpectScramble_1);
// Decrypt
utf8tScramble_1 = dciphertScramble_1
.doFinal(dectScramble_1);
// Decode using utf-8
resulttScramble_1 = new String(utf8tScramble_1,
"UTF8");
}
} else {
```

```
keytScramble_1 key = new javax.crypto.spec.SecretKeySpec(
"SocialSecurity".toCharArray(), "AES");
if ("Encrypt".equals("Encrypt")) {
// Instantiate the cipher
Cipher c = Cipher.getInstance(ALGORITHM);
c.init(Cipher.ENCRYPT_MODE, key);
byte[] encValue = c.doFinal(valueToEnc.getBytes());
String encryptedValue = new BASE64Encoder()
.encode(encValue);
return encryptedValue;
eciphertScramble_1 = javax.crypto.Cipher
.getInstance("AES");
eciphertScramble_1.init(
javax.crypto.Cipher.ENCRYPT_MODE,
skeySpectScramble_1);
enctScramble_1 = eciphertScramble_1
.doFinal(row1.ssn.getBytes());
resulttScramble_1 = new String(enctScramble_1,
"UTF8");
} else {
dciphertScramble_1 = javax.crypto.Cipher
.getInstance("AES");
dciphertScramble_1.init(
javax.crypto.Cipher.DECRYPT_MODE,
skeySpectScramble_1);
dectScramble_1 = new sun.misc.BASE64Decoder()
.decodeBuffer(row1.ssn);
byte[] original = dciphertScramble_1
.doFinal(dectScramble_1);
resulttScramble_1 = new String(original);
}
}
row1.ssn = resulttScramble_1;
row2.ssn = row1.ssn;
row2 = row1;
tos_count_tScramble_1++;
/**
* stop
*/
/**
* start
*/
currentComponent = "tLogRow_1";
// ////////////////////
StringBuilder strBuffer_tLogRow_1 = new StringBuilder();
if (row2.ssn != null) { //
strBuffer_tLogRow_1.append(String.valueOf(row2.ssn));
} //
if (globalMap.get("tLogRow_CONSOLE") != null) {
consoleOut_tLogRow_1 = (java.io.PrintStream) globalMap
.get("tLogRow_CONSOLE");
} else {
consoleOut_tLogRow_1 = new java.io.PrintStream(
```

```
new java.io.BufferedOutputStream(System.out));
globalMap.put("tLogRow_CONSOLE", consoleOut_tLogRow_1);
}
consoleOut_tLogRow_1
.println(strBuffer_tLogRow_1.toString());
consoleOut_tLogRow_1.flush();
nb_line_tLogRow_1++;
// ////
// ////
// //////////////////////
tos_count_tLogRow_1++;
/**
* stop
*/
/**
* start
*/
currentComponent = "tRowGenerator_1";
}
globalMap.put("tRowGenerator_1_NB_LINE",
nb_line_tRowGenerator_1);
ok_Hash.put("tRowGenerator_1", true);
end_Hash.put("tRowGenerator_1", System.currentTimeMillis());
/**
* stop
*/
/**
* start
*/
currentComponent = "tScramble_1";
ok_Hash.put("tScramble_1", true);
end_Hash.put("tScramble_1", System.currentTimeMillis());
/**
* stop
*/
/**
* start
*/
currentComponent = "tLogRow_1";
// ////
// ////
globalMap.put("tLogRow_1_NB_LINE", nb_line_tLogRow_1);
// //////////////////////
ok_Hash.put("tLogRow_1", true);
end_Hash.put("tLogRow_1", System.currentTimeMillis());
/**
* stop
*/// end the resume
} catch (Exception e) {
throw new TalendException(e, currentComponent, globalMap);
} catch (Error error) {
throw new Error(error);
```

```
}
globalMap.put("tRowGenerator_1_SUBPROCESS_STATE", 1);
}
public String resuming_logs_dir_path = null;
public String resuming_checkpoint_path = null;
public String parent_part_launcher = null;
private String resumeEntryMethodName = null;
private boolean globalResumeTicket = false;
public boolean watch = false;
// portStats is null, it means don't execute the statistics
public Integer portStats = null;
public int portTraces = 4334;
public String clientHost;
public String defaultClientHost = "localhost";
public String contextStr = "Default";
public String pid = "0";
public String rootPid = null;
public String fatherPid = null;
public String fatherNode = null;
public long startTime = 0;
public boolean isChildJob = false;
private boolean execStat = true;
private ThreadLocal threadLocal = new ThreadLocal();
{
java.util.Map threadRunResultMap = new java.util.HashMap();
threadRunResultMap.put("errorCode", null);
threadRunResultMap.put("status", "");
threadLocal.set(threadRunResultMap);
}
private java.util.Properties context_param = new java.util.Properties();
public java.util.Map<String, Object> parentContextMap = new java.util.HashMap<String, Object>();
public String status = "";
public static void main(String[] args) {
final test1 test1Class = new test1();
int exitCode = test1Class.runJobInTOS(args);
System.exit(exitCode);
}
public String[][] runJob(String[] args) {
int exitCode = runJobInTOS(args);
String[][] bufferValue = new String[][] { { Integer.toString(exitCode) } };
return bufferValue;
}
public int runJobInTOS(String[] args) {
String lastStr = "";
for (String arg : args) {
if (arg.equalsIgnoreCase("--context_param")) {
lastStr = arg;
} else if (lastStr.equals("")) {
evalParam(arg);
} else {
evalParam(lastStr + " " + arg);
lastStr = "";
}
```

```
}
if (clientHost == null) {
clientHost = defaultClientHost;
}
if (pid == null || "0".equals(pid)) {
pid = TalendString.getAsciiRandomString(6);
}
if (rootPid == null) {
rootPid = pid;
}
if (fatherPid == null) {
fatherPid = pid;
} else {
isChildJob = true;
}
try {
// call job/subjob with an existing context, like:
// --context=production. if without this parameter, there will use
// the default context instead.
java.io.InputStream inContext = test1.class.getClassLoader()
.getResourceAsStream(
"test/test1_0_1/contexts/" + contextStr
+ ".properties");
if (inContext != null) {
// defaultProps is in order to keep the original context value
defaultProps.load(inContext);
inContext.close();
context = new ContextProperties(defaultProps);
} else {
// print info and job continue to run, for case: context_param
// is not empty.
System.err.println("Could not find the context " + contextStr);
}
if (!context_param.isEmpty()) {
context.putAll(context_param);
}
} catch (java.io.IOException ie) {
System.err.println("Could not load context " + contextStr);
ie.printStackTrace();
}
// get context value from parent directly
if (parentContextMap != null && !parentContextMap.isEmpty()) {
}
// Resume: init the resumeUtil
resumeEntryMethodName = ResumeUtil
.getResumeEntryMethodName(resuming_checkpoint_path);
resumeUtil = new ResumeUtil(resuming_logs_dir_path, isChildJob, rootPid);
resumeUtil.initCommonInfo(pid, rootPid, fatherPid, projectName,
jobName, contextStr, jobVersion);
// Resume: jobStart
resumeUtil.addLog("JOB_STARTED", "JOB:" + jobName,
parent_part_launcher, Thread.currentThread().getId() + "", "",
"", "", "", resumeUtil.convertToJsonText(context));
```

```
long startUsedMemory = Runtime.getRuntime().totalMemory()
- Runtime.getRuntime().freeMemory();
long endUsedMemory = 0;
long end = 0;
startTime = System.currentTimeMillis();
this.globalResumeTicket = true;// to run tPreJob
this.globalResumeTicket = false;// to run others jobs
try {
errorCode = null;
tRowGenerator_1Process(globalMap);
status = "end";
} catch (TalendException e_tRowGenerator_1) {
status = "failure";
e_tRowGenerator_1.printStackTrace();
globalMap.put("tRowGenerator_1_SUBPROCESS_STATE", -1);
} finally {
}
this.globalResumeTicket = true;// to run tPostJob
end = System.currentTimeMillis();
if (watch) {
System.out.println((end - startTime) + " milliseconds");
}
endUsedMemory = Runtime.getRuntime().totalMemory()
- Runtime.getRuntime().freeMemory();
if (false) {
System.out.println((endUsedMemory - startUsedMemory)
+ " bytes memory increase when running : test1");
}
int returnCode = 0;
if (errorCode == null) {
returnCode = status != null && status.equals("failure") ? 1 : 0;
} else {
returnCode = errorCode.intValue();
}
resumeUtil.addLog("JOB_ENDED", "JOB:" + jobName, parent_part_launcher,
Thread.currentThread().getId() + "", "", "" + returnCode, "",
"", "");
return returnCode;
}
private void evalParam(String arg) {
if (arg.startsWith("--resuming_logs_dir_path")) {
resuming_logs_dir_path = arg.substring(25);
} else if (arg.startsWith("--resuming_checkpoint_path")) {
resuming_checkpoint_path = arg.substring(27);
} else if (arg.startsWith("--parent_part_launcher")) {
parent_part_launcher = arg.substring(23);
} else if (arg.startsWith("--watch")) {
watch = true;
} else if (arg.startsWith("--stat_port=")) {
String portStatsStr = arg.substring(12);
if (portStatsStr != null && !portStatsStr.equals("null")) {
portStats = Integer.parseInt(portStatsStr);
}
```

```
} else if (arg.startsWith("--trace_port=")) {
portTraces = Integer.parseInt(arg.substring(13));
} else if (arg.startsWith("--client_host=")) {
clientHost = arg.substring(14);
} else if (arg.startsWith("--context=")) {
contextStr = arg.substring(10);
} else if (arg.startsWith("--father_pid=")) {
fatherPid = arg.substring(13);
} else if (arg.startsWith("--root_pid=")) {
rootPid = arg.substring(11);
} else if (arg.startsWith("--father_node=")) {
fatherNode = arg.substring(14);
} else if (arg.startsWith("--pid=")) {
pid = arg.substring(6);
} else if (arg.startsWith("--context_param")) {
String keyValue = arg.substring(16);
int index = -1;
if (keyValue != null && (index = keyValue.indexOf('=')) > -1) {
context_param.put(keyValue.substring(0, index), keyValue
.substring(index + 1));
}
}
}
public Integer getErrorCode() {
return errorCode;
}
public String getStatus() {
return status;
}
ResumeUtil resumeUtil = null;
}
/************************************************************************************************
* 28142 characters generated by Talend Integration Suite Professional Edition
* on the January 9, 2012 9:21:51 AM EST
************************************************************************************************/
```