

## TERASOLUNA 3.3.1 移行ガイド (Batch 版)

## 変更履歴

バージョン	日付	改訂箇所	改訂内容
3.3.1	2015/02/23	－	新規作成

## ■ 概要

本ドキュメントは、TERASOLUNA Batch Framework for Java 3.2.0 で作成したアプリケーションを、3.3.1 へ移行する際の手順を示すドキュメントである。

### ◆ 3.3.1 の変更点概要

- 依存ライブラリのバージョンアップ
- Spring Bean 定義ファイルの XML スキーマ定義(XSD)からバージョン指定を削除
- バグの修正

## ■ 移行手順

3.2.0 から 3.3.1 への移行手順を説明する。

### 前提条件

TERASOLUNA Batch Framework for Java 3.2.0 を使用したアプリケーションが正常に動作していること。

以後、本書ではこれをアプリケーションと呼称する。

#### ① blank プロジェクトのダウンロード

TERASOLUNA Batch Framework for Java 3.3.1 の blank プロジェクト (terasoluna-batch4j-blank\_3.3.1.zip) を以下の URL よりダウンロードし、任意のフォルダに展開する。

ダウンロードサイトの URL :

➤ SourceForge.jp <http://sourceforge.jp/projects/terasoluna/releases/>

## ② 依存ライブラリの変更

TERASOLUNA フレームワークや Spring の依存ライブラリの差し替えを行う。  
差し替えるライブラリは以下の通りである。

3.2.0のライブラリ	3.3.1のライブラリ	種別
cglib-nodep-2.1.3.jar	-	削除
spring-asm-3.1.3.RELEASE.jar	-	削除
aspectjweaver-1.6.12.jar	aspectjweaver-1.7.4.jar	差し替え
commons-collections-3.2.jar	commons-collections-3.2.1.jar	差し替え
commons-digester-1.8.jar	commons-digester-2.0.jar	差し替え
commons-logging-1.1.1.jar	commons-logging-1.1.3.jar	差し替え
spring-aop-3.1.3.RELEASE.jar	spring-aop-3.2.13.RELEASE.jar	差し替え
spring-beans-3.1.3.RELEASE.jar	spring-beans-3.2.13.RELEASE.jar	差し替え
spring-context-3.1.3.RELEASE.jar	spring-context-3.2.13.RELEASE.jar	差し替え
spring-core-3.1.3.RELEASE.jar	spring-core-3.2.13.RELEASE.jar	差し替え
spring-expression-3.1.3.RELEASE.jar	spring-expression-3.2.13.RELEASE.jar	差し替え
spring-jdbc-3.1.3.RELEASE.jar	spring-jdbc-3.2.13.RELEASE.jar	差し替え
spring-orm-3.1.3.RELEASE.jar	spring-orm-3.2.13.RELEASE.jar	差し替え
spring-tx-3.1.3.RELEASE.jar	spring-tx-3.2.13.RELEASE.jar	差し替え
terasoluna-batch-3.2.0.jar	terasoluna-batch-3.3.1.jar	差し替え
terasoluna-batch-update-1.1.2.jar	terasoluna-batch-update-3.3.1.jar	差し替え
terasoluna-collector-1.1.0.jar	terasoluna-collector-3.3.1.jar	差し替え
terasoluna-commons-2.0.5.0.jar	terasoluna-commons-3.3.1.jar	差し替え
terasoluna-dao-2.0.5.0.jar	terasoluna-dao-3.3.1.jar	差し替え
terasoluna-filedao-2.0.3.2.jar	terasoluna-filedao-3.3.1.jar	差し替え
terasoluna-ibatis-2.0.5.0.jar	terasoluna-ibatis-3.3.1.jar	差し替え
terasoluna-logger-1.0.0.jar	terasoluna-logger-3.3.1.jar	差し替え
terasoluna-validator-2.0.5.0.jar	terasoluna-validator-3.3.1.jar	差し替え
jakarta-oro-2.0.8.jar	oro-2.0.8.jar ※名称変更のみ	差し替え

「種別」に従って下記の修正を行う。

※「3.3.1 のライブラリ」は terasoluna-batch4j-blank\_3.3.1.zip を展開したフォルダの lib フォルダ直下に格納されている。

### 削除：

アプリケーションの lib フォルダ直下から「3.2.0 のライブラリ」に記載した jar ファイルを削除する。

### 差し替え：

アプリケーションの lib フォルダ直下から「3.2.0 のライブラリ」に記載した jar ファイルを削除し、「3.3.1 のライブラリ」に記載した jar ファイルを追加する。

### ③ build.xml の修正

②依存ライブラリの変更に従って、build.xml のクラスパスの設定を修正する。  
アプリケーション直下の/ant/build.xml を②依存ライブラリの変更で使用した依存ライブラリ表を参照し、「種別」に従って下記の修正を行う。

#### 削除：

「3.2.0 のライブラリ」に記載した jar ファイルのクラスパス設定を削除する。

#### 差し替え：

「3.2.0 のライブラリ」に記載した jar ファイルから「3.3.1 のライブラリ」に記載した jar ファイルにクラスパス設定の jar ファイル名を変更する。

### ●修正例●

#### 【3.2.0 の build.xml】

```
<!-- クラスパスの設定 -->
<property name="classpath.lib" value="
    ${lib.dir}/aspectjweaver-1.6.12.jar;
    ${lib.dir}/cglib-nodep-2.1_3.jar;
    ${lib.dir}/commons-beanutils-1.8.3.jar;
    ${lib.dir}/commons-collections-3.2.jar;
    * * * (中略) * * *
"/>
```

#### 【3.3.1 の build.xml】

```
<!-- クラスパスの設定 -->
<property name="classpath.lib" value="
    ${lib.dir}/aspectjweaver-1.7.4.jar;
    ${lib.dir}/commons-beanutils-1.8.3.jar;
    ${lib.dir}/commons-collections-3.2.1.jar;
    * * * (中略) * * *
"/>
```

#### ④ classpath.bat の修正

②依存ライブラリの変更に従って、classpath.bat の設定を修正する。  
アプリケーション直下の/scripts/classpath.bat を②依存ライブラリの変更で使用した依存ライブラリ表を参照し、「種別」に従って下記の修正を行う。

##### 削除：

「3.2.0 のライブラリ」に記載した jar ファイルのクラスパス設定を削除する。

##### 差し替え：

「3.2.0 のライブラリ」に記載した jar ファイルから「3.3.1 のライブラリ」に記載した jar ファイルにクラスパス設定の jar ファイル名を変更する。

#### ●修正例●

##### 【3.2.0 の classpath.bat】

```
SET CLASSPATH=%OUTPUT_FOLDER%;%LIB_PATH%\terasoluna-batch-3.2.0.jar

SET CLASSPATH=%CLASSPATH%;%LIB_PATH%\aspectjweaver-1.6.12.jar
SET CLASSPATH=%CLASSPATH%;%LIB_PATH%\cglib-nodep-2.1.3.jar
SET CLASSPATH=%CLASSPATH%;%LIB_PATH%\commons-beanutils-1.8.3.jar

* * * (中略) * * *
```

##### 【3.3.1 の classpath.bat】

```
SET CLASSPATH=%OUTPUT_FOLDER%;%LIB_PATH%\terasoluna-batch-3.3.1.jar

SET CLASSPATH=%CLASSPATH%;%LIB_PATH%\aspectjweaver-1.7.4.jar
SET CLASSPATH=%CLASSPATH%;%LIB_PATH%\commons-beanutils-1.8.3.jar

* * * (中略) * * *
```

#### ⑤ Spring Bean 定義ファイルの XML スキーマ定義(XSD)からバージョン指定を削除

Spring プロファイル機能など、Spring3.1 またはそれ以降から登場した新しい Bean 定義の記述を使用する場合は、XML スキーマ定義(XSD)のバージョン指定を削除する。Spring 3.1 またはそれ以降に登場する Bean 定義の記述を使用しない場合は、XSD のバージョン指定を削除する必要はない。

バージョン指定を削除すると、プロパティファイルによるプレースホルダ置換機能(<context:property-placeholder>)の動作が変わり、OS の環境変数と同名のプロパティが定義されていると、プロパティの値が OS の環境変数の値で上書きされるようになる。そのため、OS の環境変数と同名のプロパティのキーがある場合は、プロパティのキー（プロパティファイル、Bean 定義ファイルの両方）を変更する必要がある。

blank プロジェクトで使用しているプロパティのなかでは、データソース定義ファイル (dataSource.xml、AdminDataSource.xml) で使用している `${username}` の変更が必要になる。Windows 環境では環境変数の `%USERNAME%` (ログオンユーザ名) で上書きされるため、データソース関連のプロパティにすべて接頭辞 `"jdbc."` を追加し、環境変数と衝突しないように変更する。

### ●修正例●

#### 【3.2.0 の AdminDataSource.xml】

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:util="http://www.springframework.org/schema/util"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/util
    http://www.springframework.org/schema/util/spring-util-2.5.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-2.5.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-2.5.xsd">

  <!-- DBCP のデータソースを設定する。 -->
  <context:property-placeholder location="SqlMapAdminConfig/jdbc.properties" />
  <bean id="adminDataSource" destroy-method="close"
    class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName" value="${driver}" />
    <property name="url" value="${url}" />
    <property name="username" value="${username}" />
    <property name="password" value="${password}" />
    <property name="maxActive" value="5" />
    <property name="maxIdle" value="1" />
    <property name="maxWait" value="5000" />
  </bean>

  * * * (中略) * * *
```

#### 【3.2.0 の jdbc.properties】

```
driver=org.postgresql.Driver
url=jdbc:postgresql://127.0.0.1:5432/postgres
username=postgres
password=postgres
```

## 【3.3.1 の AdminDataSource.xml】

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:util="http://www.springframework.org/schema/util"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/util
    http://www.springframework.org/schema/util/spring-util.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

  <!-- DBCP のデータソースを設定する。 -->
  <context:property-placeholder location="SqlMapAdminConfig/jdbc.properties" />
  <bean id="adminDataSource" destroy-method="close"
    class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName" value="${jdbc.driver}" />
    <property name="url" value="${jdbc.url}" />
    <property name="username" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
    <property name="maxActive" value="5" />
    <property name="maxIdle" value="1" />
    <property name="maxWait" value="5000" />
  </bean>

  * * * (中略) * * *
```

## 【3.3.1 の jdbc.properties】

```
jdbc.driver=org.postgresql.Driver
jdbc.url=jdbc:postgresql://127.0.0.1:5432/postgres
jdbc.username=postgres
jdbc.password=postgres
```



## ⑥ バグの修正に対する影響調査

3.3.1 で修正されたバグに対して影響調査を実施する。

以下に、アプリケーションの設定や実装に影響がある項目について説明する。なお、各項目で説明している修正例は一例であり、実際の修正内容は個々のアプリケーションにより異なる可能性がある。

- 機能説明書「BL-06 データベースアクセス機能」の「備考」にて、複数データソースの利用時は transactionManager の Bean 定義で transactionSynchronization の値を 2 に設定すると説明していた。本設定はフレームワークの誤った実装が原因で必要だったが、3.3.1 でフレームワークの実装を修正したため、本設定は不要になる。

そのため、transactionSynchronization のコメントアウト箇所を削除する。ただし、既に複数データソースを利用して transactionSynchronization の値を 2 に設定している場合は、削除すると BLogic の修正が必要になる可能性がある<sup>1</sup>。本設定を削除しなくても動作するため、無理に削除する必要はない。

### ●修正例●

#### 【3.2.0 の dataSource.xml】

```
<!-- トランザクションマネージャの定義 -->
<bean id="transactionManager"
      class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" />
  <!-- 複数 DB 接続を行うためにトランザクション同期を行わない設定 -->
  <!-- 複数の DataSource 定義が使用される場合、下のコメントアウトを削除してください。 -->
  <!-- <property name="transactionSynchronization" value="2"/> -->
</bean>
```

#### 【3.3.1 の dataSource.xml】

```
<!-- トランザクションマネージャの定義 -->
<bean id="transactionManager"
      class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" />
</bean>
```

---

<sup>1</sup>本設定を削除する場合は、プログラマティックなトランザクション管理を行っている BLogic にて、トランザクションを開始した順序と逆の順序で終了しているか確認し、なっていない場合は修正する必要がある。

- 入力データ取得機能（コレクタ）を使用し、入力チェックエラーが発生した際にスキップするように拡張入力チェックエラーハンドラクラスを実装していた場合、または、データ取得時に例外が発生した際にスキップするように拡張例外ハンドラクラスを実装していた場合、3.2.0 ではスキップしたデータを抜き取る際に、キューに空きができたことを、空き待ちを行っているスレッドに通知していなかったためにデッドロックを起こしてしまっていた。3.3.1 では修正されたため、本事象の対処方法として AbstractCollector クラスの createQueue メソッドを拡張して ArrayBlockingQueueEx の処理を修正していた場合は、バージョンアップに伴い削除する必要がある。

### ●修正例●

#### 【3.2.0 の BLogic】

```
Collector<Data> collector = new FileValidateCollector<Data>(...) {
    @Override
    protected BlockingQueue<DataValueObject> createQueue() {
        super.createQueue();
        return new ArrayBlockingQueueEx<DataValueObject>(this.queueSize) {
            @Override
            public DataValueObject poll() {
                queueLock.lock();
                try {
                    DataValueObject elm = super.poll();
                    if (elm != null) {
                        notFull.signal();
                    }
                    return elm;
                } finally {
                    queueLock.unlock();
                }
            }
        };
    }
};
```

#### 【3.3.1 の BLogic】

```
Collector<Data> collector = new FileValidateCollector<Data>(...);
```

- 非同期バッチ処理において、ジョブの実行結果をデータベースに書き込む際のエラーメッセージが誤っていた。本事象の対処方法として、terasoluna-batch-3.2.0.jar よりも先に通っているクラスパス上に META-INF/log-

messages-AL025.properties を配置してメッセージを修正していた場合、3.3.1 では本メッセージの修正に加え、その他メッセージの追加や修正があるため、terasoluna-batch-3.3.1.jar に含まれる META-INF/log-messages-AL025.properties との差分をマージする必要がある。

なお、プロジェクトで配置している log-messages-AL025.properties と terasoluna-batch-3.2.0.jar に含まれる META-INF/log-messages-AL025.properties を比較し、修正箇所がメッセージ ID「EAL025025」のみであった場合は、差分をマージせずにファイルを削除するだけで良い。

- 入力データ取得機能 (コレクタ) の Collector インタフェースで公開している getCurrent メソッド、getPrevious メソッド、getNext メソッドの仕様が修正された。そのため、コントロールブレイク機能を用いている処理、および、各公開メソッドを使用している処理において、異常データ (DTO に変換できないデータや、入力チェックエラーとなるデータ) を入力した際の挙動に問題がないか十分に確認する。

コントロールブレイク機能の ControlBreakChecker クラスは Collector インタフェースで公開している各メソッドを以下のように使用している。

- ✓ isPreBreak メソッドを使用した前ブレイク判定では、Collector インタフェースの getCurrent メソッドを使用して取得する「現在取得している入力データ」と、Collector インタフェースの getPrevious メソッドを使用して取得する「1 つ前に取得した入力データ」を用いて判定する。
- ✓ isBreak メソッドを使用した後ブレイク判定では、Collector インタフェースの getCurrent メソッドを使用して取得する「現在取得している入力データ」と、Collector インタフェースの getNext メソッドを使用して取得する「次に取得できる入力データ」を用いて判定する。

この前後のデータが異常データ (入力チェック時にエラーが発生するデータ、または、データの入力時に例外が発生するデータ) だった場合の動作の仕様が 3.2.0 では未定義であったため、不正な動作となっていた場合があった。しかし、3.3.1 では仕様が定義され、不正な動作となっていた箇所が正常な動作となるよう修正された。

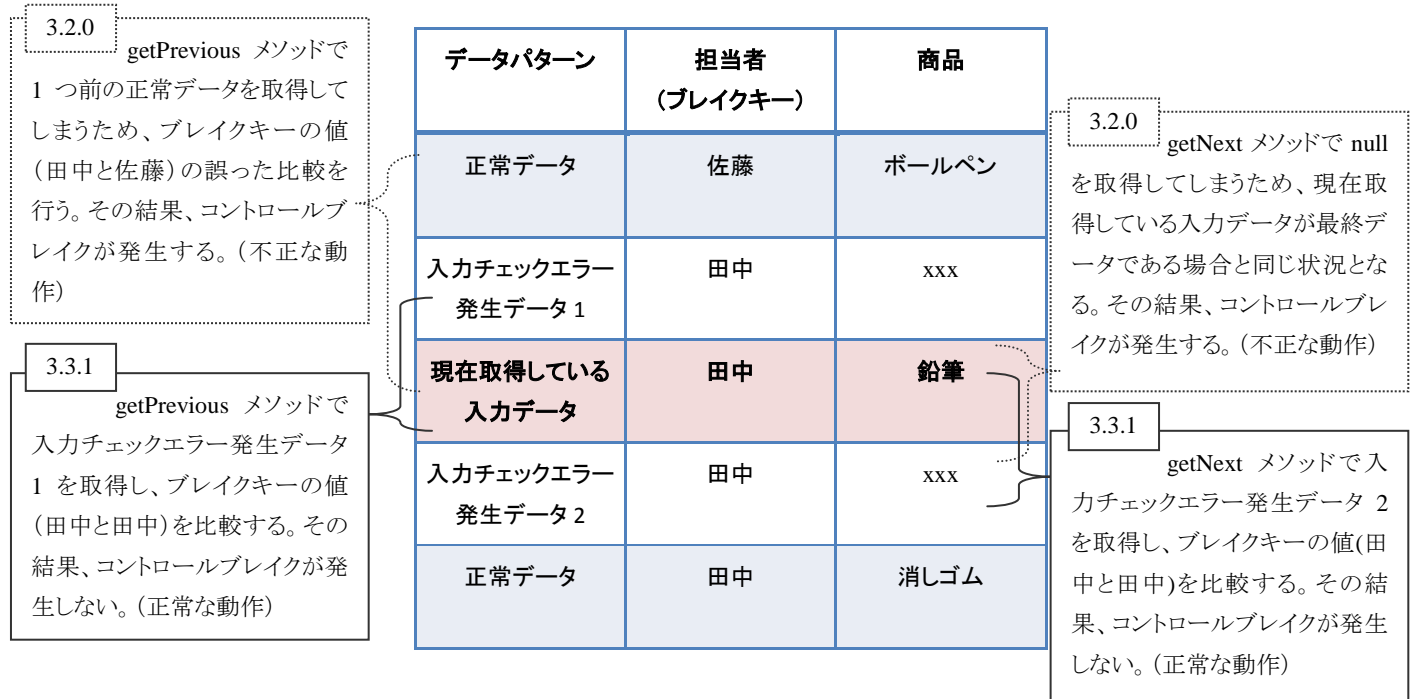
以下に、BLogic の実装バリエーション (拡張入力チェックエラーハンドラの実装と拡張例外ハンドラの実装の組み合わせ) による異常データ検出時の振る舞いを表で示す。BLogic がどの実装になっているかを確認し、3.2.0 から 3.3.1 への移行で生じる前ブレイク判定/後ブレイク判定の動作の違いによる影響を確認する。

なお、拡張入力チェックエラーハンドラの実装方法については機能説明書「AL-043 入力チェック機能」を、拡張例外ハンドラの実装方法については機能説明書「AL-041 入力データ取得機能」を参照。

		1	2	3	4	5	6	7	8	9
拡張入力チェックエラーハンドラの実装	例外をスローする(拡張入力チェックエラーハンドラを実装しない場合を含む)	○	○	○						
	当該データの取得をスキップする				○	○	○			
	入力データの取得を終了する							○	○	○
拡張例外ハンドラの実装	発生した例外をそのままスローする(拡張例外ハンドラを実装しない場合を含む)	○			○			○		
	当該データの取得をスキップする		○			○			○	
	入力データの取得を終了する			○			○			○
異常データの種類	入力チェックエラーが発生	○	○	○	○	○	○	○	○	○
	データの入力時に例外が発生	○	○	○	○	○	○	○	○	○
異常データ検出時の振る舞い	パターン1	○								
	パターン2		○			○			○	
	パターン3		○	○		○	○	○		○
	パターン4			○	○			○	○	○

※入力チェックエラー発生時に当該データを取得する場合は、3.2.0 と 3.3.1 で動作に違いがないため考慮していない。

### ●パターン 1



## ● パターン 2

3.2.0

getPrevious メソッドで 1 つ前の正常データが取得されてしまうため、ブレイクキーの値 (田中と佐藤) の誤った比較を行う。その結果、コントロールブレイクが発生する。(不正な動作)

3.3.1

getPrevious メソッドで例外発生データ 1 を取得する際、例外が発生する。その結果、コントロールブレイクの判定が行われない。(正常な動作)

データパターン	担当者 (ブレイクキー)	商品
正常データ	佐藤	ボールペン
例外発生データ 1	田中 DTO に変換できないため参照不能	xxx
現在取得している 入力データ	田中	鉛筆
例外発生データ 2	田中 DTO に変換できないため参照不能	xxx
正常データ	田中	消しゴム

3.2.0

getNext メソッドで null を取得してしまうため、現在取得している入力データが最終データである場合と同じ状況となる。その結果、コントロールブレイクが発生する。(不正な動作)

3.3.1

getNext メソッドで例外発生データ 2 を取得する際、例外が発生する。その結果、コントロールブレイクの判定が行われない。(正常な動作)

## ● パターン 3

3.2.0

getPrevious メソッドで 1 つ前の正常データを取得し、ブレイクキーの値 (田中と佐藤) を比較する。その結果、コントロールブレイクが発生する。(正常な動作)

3.3.1

3.2.0 と同じ動作となる。(変更なし)

データパターン	担当者 (ブレイクキー)	商品
正常データ	佐藤	ボールペン
入力チェックエラー ／例外発生データ 1	田中	xxx
現在取得している 入力データ	田中	鉛筆
入力チェックエラー ／例外発生データ 2	田中	xxx
正常データ	田中	消しゴム

3.2.0

getNext メソッドで null を取得してしまうため、現在取得している入力データが最終データである場合と同じ状況となる。その結果、コントロールブレイクが発生する。(不正な動作)

3.3.1

getNext メソッドで次に取得できる正常データを取得し、ブレイクキーの値 (田中と田中) を比較する。その結果、コントロールブレイクが発生しない。(正常な動作)

## ●パターン 4

※入力チェックエラー／例外発生データ、および、入力チェックエラー／例外発生後のデータは取得できないため、後ブレイク判定のみを説明する。

データパターン	担当者 (ブレイクキー)	商品
現在取得している 入力データ	田中	鉛筆
入力チェックエラー ／例外発生データ 1	田中	xxx
正常データ	田中	消しゴム

3.2.0

現在取得している入力データが最終データ扱いとなり、getNext メソッドで null を取得する。その結果、コントロールブレイクが発生する。(正常な動作)

3.3.1

3.2.0 と同じ動作となる。(変更なし)

⑦ 動作確認

アプリケーションを起動させて、問題なく動作することを回帰試験等により確認すること。