

# JEUS 並行処理ユーティリティガイド

JEUS v8.0



Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

## Copyright Notice

Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

45, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13613, South Korea

## Restricted Rights Legend

All TmaxSoft Software (JEUS®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd.

Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features. This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

このソフトウェア(JEUS®)マニュアルの内容とプログラムは、日本国の著作権法および国際条約によって保護されています。マニュアルの内容とプログラムは、TmaxSoft Co., Ltd.との使用許諾契約書の下でのみ使用することができ、マニュアルは使用許諾契約で許可されている範囲を除いては、配布または複製することができません。TmaxSoftの書面による事前の承諾を得ることなく、このマニュアルの全部または一部を電子的または機械的な方法を問わず、転送、複製、配布したり、または二次的著作物を作成する等の行為を一切禁じます。

このソフトウェアのマニュアルとプログラムの使用許諾契約は、いかなる場合においても、マニュアル及びプログラムと関連する知的財産権(登録の有無を問わず)を譲渡するものと解釈されず、TmaxSoftのブランド、ロゴ、商標等の使用権限を与えるものではありません。マニュアルは、情報を提供する目的でのみ提供しており、これに伴う契約上の直接的ないしは間接的な責任を負わず、マニュアルの内容は法律上もしくは商業的な特定の条件が満たされることを保証しません。マニュアルの内容は、製品のアップグレード及び修正により、その内容が予告なく変更されることがあり、内容上の誤りがないことを保証しません。

## Trademarks

JEUS® is registered trademark of TmaxSoft Co., Ltd.

JEUS®は、TmaxSoft Co., Ltd.の登録商標です。

Java and Solaris are registered trademarks of Oracle Corporation and its subsidiaries and affiliates.

Java、Solarisは、Oracle Corporation及びその子会社、関連会社の登録商標です。

Microsoft, Windows, and Windows NT are registered trademarks or trademarks of Microsoft Corporation.

Microsoft、Windows、Windows NTは、Microsoft Corporationの登録商標または商標です。

HP-UX is a registered trademark of Hewlett Packard Enterprise Company.

HP-UXは、Hewlett Packard Enterprise Companyの登録商標です。

---

AIX is a registered trademark of International Business Machines Corporation.

AIXは、International Business Machines Corporationの登録商標です。

UNIX is a registered trademark of X/Open Company, Ltd.

UNIXは、X/Open Company, Ltd.の登録商標です。

Linux is a registered trademark of Linus Torvalds.

Linuxは、Linus Torvaldsの登録商標です。

Other products and company names are trademarks or registered trademarks of their respective owners.

その他、記載されている会社名、製品名などは、各社の商号、商標または登録商標です。

The names of companies, systems, and products mentioned in this manual may not necessarily be indicated with a trademark symbol (TM, ®).

本マニュアルに記載されている会社名、システム名、製品名などには必ずしも商標表示(TM、®)を付記しておりません。

### **Open Source Software Notice**

Some modules or files of this product are subject to the terms of the following licenses. : APACHE2.0, CDDL1.0, EDL1.0, OPEN SYMPHONY SOFTWARE1.1, TRILEAD-SSH2, Bouncy Castle, BSD, MIT, SIL OPEN FONT1.1

Detailed Information related to the license can be found in the following directory : \${INSTALL\_PATH}/lib/licenses

この製品の一部ファイルまたはモジュールは、APACHE2.0、CDDL1.0、EDL1.0、OPEN SYMPHONY SOFTWARE1.1、TRILEAD-SSH2、Bouncy Castle、BSD、MIT、SIL OPEN FONT1.1のライセンスに準拠します。

### **文書情報**

文書名: JEUS 並行処理ユーティリティガイド

発行日: 2016年10月14日

ソフトウェアバージョン: JEUS v8.0

ガイドバージョン: v2.1.1

---



# 目次

このガイドについて .....	xi
<b>第1章 Concurrency Utilities for Java EE .....</b>	<b>1</b>
1.1. 概要 .....	1
1.2. Managed Task .....	1
1.3. コンテナ・スレッド・コンテキスト .....	1
1.3.1. コンテキストを維持しながらタスクを実行するポイント .....	2
<b>第2章 Managed Objects .....</b>	<b>3</b>
2.1. ManagedExecutorServiceインターフェース .....	3
2.2. ManagedScheduledExecutorServiceインターフェース .....	6
2.3. ContextServiceインターフェース .....	9
2.4. ManagedThreadFactoryインターフェース .....	11



## 図目次

[図 2.1]	WebAdminを利用したManagedExecutorServiceの設定 .....	5
[図 2.2]	WebAdminを利用したManagedScheduledExecutorServiceの設定 .....	8
[図 2.3]	WebAdminを利用したContextServiceの設定 .....	10
[図 2.4]	WebAdminを利用したManagedThreadFactoryの設定 .....	12





## 例目次

[例 2.1]	ManagedExecutorServiceをリソースとして定義する例：<<domain.xml>> .....	3
[例 2.2]	ManagedExecutorServiceを利用したアプリケーションの例 .....	4
[例 2.3]	ManagedScheduledExecutorServiceをリソースとして定義する例：<<domain.xml>> .....	6
[例 2.4]	ManagedScheduledExecutorServiceを利用したアプリケーションの例 .....	7
[例 2.5]	ContextServiceをリソースとして定義する例：<<domain.xml>> .....	9
[例 2.6]	ContextServiceを利用したアプリケーションの例 .....	9
[例 2.7]	ManagedThreadFactoryをリソースとして定義する例：<<domain.xml>> .....	11
[例 2.8]	ManagedThreadFactoryを利用したアプリケーションの例 .....	11



# このガイドについて

## 対象読者

Concurrency Utilities for Java EE (JSR-236)は、並行処理を行う非同期タスクをアプリケーション・サーバーがコンテキストを維持して実行・管理する標準技術です。

本書は、JEUS<sup>®</sup>(以下、JEUS)でConcurrency Utilities for Java EEを利用して開発を行う開発者を対象としています。

## 前提知識

JEUSの基本的な使用方法と製品を理解するには、以下のガイドについてあらかじめ熟知することをお勧めします。

- 『JEUS 紹介ガイド』
- 『JEUS インストール & スタートガイド』

本書のすべてのサンプルと環境構成は、UNIXスタイルに準拠します。Microsoft Windows<sup>™</sup>(以下、Windows)など他の環境で作業を行う場合は、次のような事項を考慮してください。

たとえば、Windowsプラットフォームでは、ディレクトリー区切り子をUNIXスタイルのスラッシュ(/)からWindowsスタイルのバックスラッシュ(\)に変えて使用してください。また、環境変数もWindowsスタイル(%%)に変更して使用してください。

本書で触れているJEUS\_HOMEは、JEUSがインストールされているディレクトリーです。

## 制限事項

本書の内容は、Java標準に準拠して作成されていますが、本書で触れているJava EEやJava仕様については詳しく取り上げていません。関連内容についてはJava関連ドキュメントを参照してください。

## 本書の構成

本書は、計2章で構成されています。

- [「第1章 Concurrency Utilities for Java EE」](#)

Concurrency Utilities for Java EEについて説明します。

- [「第2章 Managed Objects」](#)

Managed Objectsについて説明し、JEUSでManaged Objectsを利用した非同期タスクの例を説明します。

## 表記上の規則

表記	意味
<<AaBbCc123>>	プログラム・ソースコードのファイル名
<Ctrl>+C	CtrlキーとCキーを同時に押す
[Button]	GUIのボタン、メニュー名
太字	強調
「」、『』（鍵カッコ）	関連文書、あるいはガイド内の他の章および節の表示
「入力項目」	画面UI上の入力項目
ハイパーリンク	メール・アカウント、Webサイト
>	メニューの実行順
+----	下位ディレクトリー/ファイル有り
----	下位ディレクトリー/ファイル無し
<div>参考</div>	参照/注意事項
<div>注</div>	注意事項
[図 1.1]	図の名称
[表 1.1]	表の名称
AaBbCc123	Javaコード、XMLドキュメント
[ <i>command argument</i> ]	オプション・パラメータ
< xyz >	「<」と「>」の間の内容は実際に使用される特定の名称または値で置き換えられる
	構文の中の相互に排他的な選択項目の選択肢を示す 例) A B: AとBのいずれかを選択
...	パラメータ、値、または他の情報が繰り返される
\${ }	環境変数

## システム要件

	要求事項
プラットフォーム	Solaris 9, 10, 11
	HP-UX 11.x, 11i, 11iV2
	IBM AIX 5L, 6L, AIX 7L
	MS Windows 2008, 2012, Vista, 7, 8
ハードウェア	最小2GB以上、推奨20GBのハードディスク容量
	推奨1GB以上のメモリー容量
JDK	JDK 7, JDK 8

## 関連文書

ガイド	説明
JEUS 紹介ガイド	JEUSサーバーについて全般的に紹介し、JEUSのアーキテクチャーを含む各構成要素について記述しています
JEUS インストール&スタートガイド	JEUSについて紹介し、JEUSのインストールおよび開始方法について記述しています
JEUS サーバガイド	JEUSシステムおよびサーバーの概要とシステムの管理方法について記述しています
JEUS WebAdminガイド	JEUSのWeb管理ツールであるWebAdminを利用したJEUSの設定および制御、モニタリング、クラスタリング、リソースの設定および管理について記述しています

## 参考文献

- Concurrency Utilities for Java EE(JSR-236)仕様

## お問合せ先

### Korea

TmaxSoft Co., Ltd.  
45, Jeongjail-ro, Bundang-gu,  
Seongnam-si, Gyeonggi-do, 13613  
South Korea  
Tel: +82-31-8018-1000  
Fax: +82-31-8018-1115  
Email: [info@tmax.co.kr](mailto:info@tmax.co.kr)  
Web (Korean): <http://www.tmaxsoft.com>  
TechNet: <http://technet.tmaxsoft.com>

### USA

TmaxSoft Inc.  
101 North Wacker Drive, Suite 2014,  
Chicago, IL 60606  
U.S.A  
Tel: +1-312-525-8330  
Email: [info@tmaxsoft.com](mailto:info@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/us\\_en/home](http://www.tmaxsoft.com/us_en/home)

### Japan

TmaxSoft Japan Co., Ltd.  
5F Sanko Bldg, 3-12-16 Mita, Minato-Ku, Tokyo, 108-0073  
Japan  
Tel: +81-3-5765-2550  
Fax: +81-3-5765-2567  
Email: [info@tmaxsoft.co.jp](mailto:info@tmaxsoft.co.jp)  
Web (Japanese): <http://www.tmaxsoft.co.jp>

## China

Beijing TmaxSoft System Software Co., Ltd.  
Room103, No.2 Huizhong Building, Seven Street Shangdi,  
Haidian District, Beijing, 100085  
P.R.China  
Tel: +86-10-6298-8827  
Email: [info@tmaxsoft.com.cn](mailto:info@tmaxsoft.com.cn)  
Web (Chinese): [http://www.tmaxsoft.com/cn\\_en/home\\_cn\\_en](http://www.tmaxsoft.com/cn_en/home_cn_en)

## Brazil

Tmax Brasil Sistemas e Serviços Ltda.  
Av. Copacabana, 177, sala 32~35 Empresarial 18 do Fortel  
Alphaville Barueri, Sao Paulo, 06472-001  
Brazil  
Tel: +55-11-4191-3100  
Fax: +55(11) 4191-3705 (extension#112)  
Email: [info.bra@tmaxsoft.com](mailto:info.bra@tmaxsoft.com)  
Web (Portuguese): [http://www.tmaxsoft.com/br\\_en/home\\_br\\_en](http://www.tmaxsoft.com/br_en/home_br_en)

## Russia

Tmax Rus L.L.C.  
Leninsky prospekt, 113/1 (Park Place Moscow),  
Office 318e, Moscow, 117198  
Russia  
Tel: +7(495)970-01-35  
Email: [info.rus@tmaxsoft.com](mailto:info.rus@tmaxsoft.com)  
Web (Russian): [http://www.tmaxsoft.com/ru\\_ru/home\\_ru\\_ru](http://www.tmaxsoft.com/ru_ru/home_ru_ru)



## Singapore

Tmax Singapore Pte. Ltd.  
430 Lorong 6, Toa Payoh #10-02,  
OrangeTee Building, 319402  
Singapore  
Tel: +65-6259-7223  
Fax: +65-6258-7112  
Email: [info.sg@tmaxsoft.com](mailto:info.sg@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/sg\\_en/home\\_sg\\_en](http://www.tmaxsoft.com/sg_en/home_sg_en)

## United Kingdom

TmaxSoft UK Ltd.  
215 Knyvett House, Watermans Business Park,  
The Causeway, Staines TW18 3BAB  
United Kingdom  
Tel: +44-1784-895005  
Email: [info.uk@tmaxsoft.com](mailto:info.uk@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/gb\\_en/home\\_gb\\_en](http://www.tmaxsoft.com/gb_en/home_gb_en)

## Canada

TmaxSoft Canada, Inc.  
2425 Matheson Blvd East, 8th floor,  
Unit 824 Mississauga, ON, L4W 5K4  
Canada  
Tel: +1-905-361-2888  
Email: [info.canada@tmaxsoft.com](mailto:info.canada@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/ca\\_en/home\\_ca\\_en](http://www.tmaxsoft.com/ca_en/home_ca_en)

## Australia

TmaxSoft Proprietary Limited  
L32, 101 Miller Street, North Sydney 2060  
Australia  
Tel: +91-9845-330-704  
Email: [info.aus@tmaxsoft.com](mailto:info.aus@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/au\\_en/home\\_au\\_en](http://www.tmaxsoft.com/au_en/home_au_en)

## India

TmaxSoft Technologies Private Limited  
Sobha Alexander Plaza, 3rd Floor,  
16/2 Commissariat Road, Bangalore-560025  
India  
Tel: +91-9845-330-704  
Email: [info.india@tmaxsoft.com](mailto:info.india@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/in\\_en/home\\_in\\_en](http://www.tmaxsoft.com/in_en/home_in_en)

## Turkey

TmaxSoft Co., Ltd. Turkey Liaison Office  
Windowist Tower. Eski Buyukdere Cad. No:26,  
Maslak 34467 Istanbul  
Turkey  
Tel: +90-544-553-6045  
Email: [cslee@tmaxsoft.com](mailto:cslee@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/tr\\_en/home\\_tr\\_en](http://www.tmaxsoft.com/tr_en/home_tr_en)

# 第1章 Concurrency Utilities for Java EE

本章では、Concurrency Utilities for Java EEの登場背景および技術について簡単に説明します。

## 1.1. 概要

アプリケーション・サーバーのEJBやWebコンポーネントでJava SEが提供するConcurrency API (Thread、Timer、ExecutorService、...)を利用することは推奨しません。Java EEアプリケーションのコンポーネント (Servlet、EJBなど)は、アプリケーション・サーバーが管理するスレッドで実行され、同じスレッドでコンテナが提供する機能が動作することを仮定します。

このような理由で、Appコンポーネントはコンテナが管理しないスレッドではJava EEサービスを安全に使用できません。また、コンテナが管理しないスレッドでリソースを利用する場合、Java EEでユーザービリティ、セキュリティ、スケーラビリティの問題を潜在的に起こす可能性があります。

このように管理されない(unmanaged)スレッドによって発生する問題を解決するために、従来のJava SEのConcurrency Utilitiesを拡張したConcurrency Utilities for Java EE(JSR-236)が提供されています。

この標準化された技術により、Java EE環境でコンテナの整合性を損なわずにアプリケーションが動作することを保証できます。

## 1.2. Managed Task

コンテナは不要なリソースの消費を減らすために、リソースをプーリングしてライフサイクルを管理します。ところが、コンポーネント内でJava SEが提供するConcurrency APIを利用して非同期タスクを実行すると、コンテナがそのリソースを検知できないため管理することができません。

Concurrency Utilities for Java EEでは既存のJava SEの`java.util.concurrent`に定義されているタスクを拡張してManaged Task(管理されるタスク)を定義しています。これにより、コンテナがその一般タスクをManaged Taskとして管理することができ、非同期でタスクが実行されたときに、実行コンテキストを維持して実行されるようにします。

## 1.3. コンテナ・スレッド・コンテキスト

Java EE環境ではサービスを実行時に各サービスのコンテキスト情報を持っています。たとえば、JDBCのデータソースやJMSのプロバイダーとEJBなどがあります。

コンポーネント内でJava SEのConcurrency APIを利用する場合、コンテナが新規作成されたスレッドでサービスのコンテキスト情報を維持するためには、アプリケーション開発者は次のような方式でコンテキストを伝播する必要があります。

1. アプリケーション・コンポーネント・スレッドのコンテナ・コンテキストを保存します。
2. どのコンテナ・コンテキストを保存して伝播するかを判断します。
3. 新規作成されたスレッドにコンテナ・コンテキストを適用します。
4. 元のコンポーネント・スレッドのコンテキストを回復します。

このような方式でJava SEのConcurrency APIでコンテキストを維持しながらタスクを実行することができますが、Concurrency Utilities for Java EEサービスを利用すると、ユーザーが定義したタスクをManaged Objects(管理対象オブジェクト)に転送するだけで、内部でコンテキストを自動で維持・回復するので便利かつ安全です。

### 1.3.1. コンテキストを維持しながらタスクを実行するポイント

- `java.util.concurrent.Callable`
  - `call()`
- `java.lang.Runnable`
  - `run()`
- `javax.enterprise.concurrent.ManagedTaskListener`
  - `taskAborted`
  - `taskSubmitted`
  - `taskStarting`
- `javax.enterprise.concurrent.ManagedTaskListener`
  - `taskAborted`
  - `taskSubmitted`
  - `taskStarting`
- `javax.enterprise.concurrent.Trigger`
  - `getNextRuntime()`
  - `skipRun()`

## 第2章 Managed Objects

本章では、Concurrency Utilities for Java EEが提供するManaged Objects(管理対象オブジェクト)とその使用例を説明します。

### 2.1. ManagedExecutorServiceインターフェース

javax.enterprise.concurrent.ManagedExecutorServiceインターフェースは、Java SEの java.util.concurrent.ExecutorServiceインターフェースを継承します。ExecutorServiceと同様に、非同期タスクの実行のために利用されます。アプリケーション・サーバーは非同期で実行されるタスクのコンテキスト情報を維持します。

### リソースの定義例

以下は、ManagedExecutorServiceをリソースとして定義する例です。

**[例 2.1] ManagedExecutorServiceをリソースとして定義する例 : <<domain.xml>>**

```
<domain>
  ...
  <server>
    <data-sources>
      <data-source>testdb</data-source>
    </data-sources>
    <managed-executor-service>mes1</managed-executor-service>
  </server>

  <resources>
    <managed-executor-service>
      <export-name>mes1</export-name>
      <long-running-task>true</long-running-task>
      <thread-pool>
        <min>10</min>
        <max>20</max>
        <keep-alive-time>60000</keep-alive-time>
        <queue-size>4096</queue-size>
      </thread-pool>
    </managed-executor-service>
  </resources>
```

```
...
</domain>
```

## アプリケーションの例

以下は、ManagedExecutorServiceを利用したアプリケーションの例です。

### [例 2.2] ManagedExecutorServiceを利用したアプリケーションの例

```
public class AppServlet extends HttpServlet implements Servlet {
    // Retrieve our executor instance.
    @Resource(name="mes1")
    ManagedExecutorService mes;

    protected void doPost(HttpServletRequest req, HttpServletResponse
        resp) throws ServletException, IOException {
        ArrayList<Callable> builderTasks = new ArrayList<Callable>();
        builderTasks.add(new AccountTask(reqID, accountID));
        builderTasks.add(new InsuranceTask(reqID, accountID));

        // Submit the tasks and wait.
        List<Future<Object>> results = mes.invokeAll(builderTasks);

        AccountInfo accountInfo = (AccountInfo) results.get(0).get();
        InsuranceInfo insInfo = (InsuranceInfo) results.get(1).get();
        // Process the results
    }
}
```

# WebAdminを利用した設定例

WebAdminで左側の[Resources]メニューの「Concurrency Utilities Resource」項目で ManagedExecutorServiceリソース設定を行えます。

[図 2.1] WebAdminを利用したManagedExecutorServiceの設定

Managed Executor Service

HISTORY

ヘルプ

アプリケーションで使用できるManagedExecutorService(管理対象エグゼキュータサービス)を定義します。タスクがワーカースレッドで実行されるときも、コンテキスト情報(セキュリティ、トランザクション、実行コンテキストなど)を取得できるようにします。

動的設定 必須項目

確認 再設定

Export Name

ネーミングサーバに登録するManaged Executor Serviceの名前を設定します。

Long Running Task

☐ [デフォルト: false] Managed Executor Serviceで動作するタスクが長時間実行されるタスクかどうかをブーリアン値で設定します。

Thread Pool

Managed Executor Service内部で使われるスレッドプールを設定します。

Min

[デフォルト: 0] スレッドプールで管理するスレッドの最小数です。

Max

[デフォルト: 10] スレッドプールで管理するスレッドの最大数です。

Keep Alive Time

ms [デフォルト: 60000] 最小値(Min)以下のスレッドに対し、設定された時間内に使用されない場合は自動的にスレッドプールから削除されます。0の場合は削除しません。

Queue Size

[デフォルト: 4096] スレッドプールで処理するタスクを蓄積するキューのサイズを指定します。

Stuck Thread Handling

スレッドが特定のタスクによって一定時間以上継続して占有されている場合、当該スレッドに対して適切なアクションを取るための設定です。

Max Stuck Thread Time

ms [デフォルト: 3600000] スレッドをスタックスレッドと判断する基準になる値を設定します。設定された時間以上継続して占有されている状態であれば、当該スレッドをスタックスレッドと見なします。

Action On Stuck Thread

[デフォルト: None] スタックスレッドと判断された場合、そのスレッドに対して適切なアクションを取るための設定です。

Stuck Thread Check Period

[デフォルト: 300000] スタックスレッドの状態をチェックする周期を設定します。

User Warning Class

Action On Stuck ThreadがWarningに設定された場合、デフォルトではスレッドダンプを取りますが、希望する作業を行うために直接クラスを作成したい場合は、この設定を使用します。作成するクラスは、jeus.util.pool.Warningを実装する必要があり、jeus.util.pool.Warningインターフェースはjclient.jarで見つけることができます。作成したクラスは、使用する対象サーバのSERVER\_HOME/lib/applicationに置きます。

確認 再設定

## 2.2. ManagedScheduledExecutorServiceインターフェース

javax.enterprise.concurrent.ManagedScheduledExecutorServiceインターフェースは、ManagedExecutorServiceの全機能を継承すると共に、Java SEのjava.util.concurrent.ScheduledExecutorServiceの機能を継承し、タスクの遅延実行と周期実行機能を提供します。さらに、トリガーとManagedTaskListenerインターフェースにより、タスクの実行を制御できるようにします。

### リソースの定義例

以下は、ManagedScheduledExecutorServiceをリソースとして定義する例です。

**[例 2.3] ManagedScheduledExecutorServiceをリソースとして定義する例：<<domain.xml>>**

```
<domain>
  ...
  <server>
    <data-sources>
      <data-source>testdb</data-source>
    </data-sources>

    <managed-scheduled-executor-service>mse1</managed-scheduled-executor-service>
  </server>
  <resources>
    <managed-scheduled-executor-service>
      <export-name>mse1</export-name>
      <long-running-task>true</long-running-task>
      <thread-pool>
        <min>10</min>
        <max>20</max>
        <keep-alive-time>60000</keep-alive-time>
        <queue-size>4096</queue-size>
      </thread-pool>
    </managed-scheduled-executor-service>
  </resources>
  ...
</domain>
```



## アプリケーションの例

以下は、ManagedScheduledExecutorServiceを利用したアプリケーションの例です。

### [例 2.4] ManagedScheduledExecutorServiceを利用したアプリケーションの例

```
public class AppServlet extends HttpServlet implements Servlet {
    @Resource(name="mses1")
    ManagedScheduledExecutorService mses;

    protected void doPost(HttpServletRequest req, HttpServletResponse
        resp) throws ServletException, IOException {
        Runnable printTask = new Runnable() {
            @Override
            public void run() {
                System.out.println(System.currentTimeMillis());
            }
        };

        // printTaskが5秒ごとに実行される
        mses.schedule(printTask, 5, TimeUnit.SECONDS);
    }
}
```

## WebAdminを利用した設定例

WebAdminで左側の[Resources]メニューの「Concurrency Utilities Resource」項目で ManagedScheduledExecutorServiceリソース設定を行えます。

【図 2.2】 WebAdminを利用したManagedScheduledExecutorServiceの設定

Managed Scheduled Executor Service

アプリーケーションで使用できるManagedExecutorService(管理対象エグゼキュータサービス)を定義します。タスクがワーカスレッドで実行される時も、コンテキスト情報(セキュリティ、トランザクション、実行コンテキストなど)を取得できるようにします。選択事項として、javax.enterprise.concurrent.Triggerの実装をタスクのサブミット時に一緒に渡せば、タスクの実行時点を制御することができます。

動的設定 必須項目

Export Name

ネーミングサーバに登録するManaged Scheduled Executor Serviceの名前を設定します。

Long Running Task

☐ [デフォルト: false] Managed Scheduled Executor Serviceで動作するタスクが長時間実行されるタスクかどうかをブーリアン値で設定します。

Thread Pool

Managed Scheduled Executor Service内部で使われるスレッドプールを設定します。

Min

[デフォルト: 0] スレッドプールで管理するスレッドの最小数です。

Max

[デフォルト: 10] スレッドプールで管理するスレッドの最大数です。

Keep Alive Time

[デフォルト: 60000] 最小値(Min)以下のスレッドに対し、設定された時間内に使用されない場合は自動的にスレッドプールから削除されます。0の場合は削除しません。

Queue Size

[デフォルト: 4096] スレッドプールで処理するタスクを蓄積するキューのサイズを指定します。

Stuck Thread Handling

スレッドが特定のタスクによって一定時間以上継続して占有されている場合、当該スレッドに対して適切なアクションを取るための設定です。

Max Stuck Thread Time

[デフォルト: 3600000] スレッドをスタックスレッドと判断する基準になる値を設定します。設定された時間以上継続して占有されている状態であれば、当該スレッドをスタックスレッドと見なします。

Action On Stuck Thread

[デフォルト: None] スタックスレッドと判断された場合、そのスレッドに対して適切なアクションを取るための設定です。

Stuck Thread Check Period

[デフォルト: 300000] スタックスレッドの状態をチェックする間隔を設定します。

User Warning Class

Action On Stuck ThreadがWarningに設定された場合、デフォルトではスレッドダンプを取りますが、希望する作業を行うために直接クラスを作成したい場合は、この設定を使用します。作成するクラスは、jeus.util.pool.Warningを実装する必要があり、jeus.util.pool.Warningインターフェースはjclient.jarで見つけることができます。作成したクラスは、使用する対象サーバのSERVER\_HOME/lib/applicationに置きます。

確認 再設定

## 2.3. ContextServiceインターフェース

ExecutorServiceを利用せずにContextServiceによりManaged Taskを作成する方法を提供します。本機能を利用すると、ユーザーがタスクを作成するときにコンテキストに気を配らなくても、アプリケーション・サーバー内部でタスクの実行時にコンテキストを維持します。より詳細には、動的プロキシを利用して現在のタスクを実行する前に、該当するスレッドにコンテキストを設定してタスクを実行します。すべてのタスクが完了したら、コンテキストを回復するタスクを代わりに実行します。

## リソースの定義例

以下は、ContextServiceをリソースとして定義する例です。

**[例 2.5] ContextServiceをリソースとして定義する例 : <<domain.xml>>**

```
<domain>
    ...
    <server>
        <data-sources>
            <data-source>testdb</data-source>
        </data-sources>
        <context-service>cs1</context-service>
    </server>

    <resources>
        <context-service>
            <export-name>cs1</export-name>
        </context-service>
    </resources>
    ...
</domain>
```

## アプリケーションの例

以下は、ContextServiceを利用したアプリケーションの例です。

**[例 2.6] ContextServiceを利用したアプリケーションの例**

```
public class AppServlet extends HTTPServlet implements Servlet {
    @Resource(name="cs1")
    ContextService cs;

    protected void doPost(HttpServletRequest req, HttpServletResponse
        resp) throws ServletException, IOException {
        // 一般的な実行可能なタスク
        Runnable simpleTask = new Runnable() {
```

```

@Override
public void run() {
    int sum = 0;
    for (int i = 0; i < 10; i++) { sum += i; }
    System.out.println(sum);
}
};

// 一般のタスクをContextServiceによりコンテキストチュアル・タスク(Contextual task)
// として作成
cs.createContextualProxy(simpleTask, Runnable.class);

// SEが提供するExecutorにコンテキストチュアル・タスクを渡す
ExecutorService es = Executors.newFixedThreadPool(1);
es.submit(simpleTask);
}
}

```

## WebAdminを利用した設定例

WebAdminで左側の[Resources]メニューの「**Concurrency Utilities Resource**」項目でContextServiceリソース設定を行えます。

[図 2.3] WebAdminを利用したContextServiceの設定

**Context Service** HISTORY

アプリケーションで利用できるContextService(コンテキストサービス)を定義します。タスクがJava SEで提供される並列処理ユーティリティ(例: ExecutorServiceなど)で実行される場合でも、タスクのコンテキストを維持させます。定義されたタスクをContextServiceを介して提供されるプロキシオブジェクトに変換して実行する必要があります。

ヘルプ ?

動的設定 \* 必須項目

確認 再設定

Export Name \*

ネーミングサーバに登録するContext Serviceの名前を設定します。

確認 再設定

## 2.4. ManagedThreadFactoryインターフェース

javax.enterprise.concurrent.ManagedThreadFactoryインターフェースは、Java SEの java.util.concurrent.ThreadFactory機能を継承してスレッドの作成機能を提供します。一般的にThreadFactoryは、ThreadPoolExecutorを作成するときに選択的に設定することができ、スレッドの作成を制御する機能を行います。したがって、Java SEのConcurrency APIでもワーカー・スレッドがタスクを実行時にタスクのコンテキストを維持することができます。

### リソースの定義例

以下は、ManagedThreadFactoryをリソースとして定義する例です。

**[例 2.7] ManagedThreadFactoryをリソースとして定義する例 : <<domain.xml>>**

```
<domain>
  ...
  <server>
    <data-sources>
      <data-source>testdb</data-source>
    </data-sources>
    <managed-thread-factory>mtf1</managed-thread-factory>
  </server>

  <resources>
    <managed-thread-factory>
      <export-name>mtf1</export-name>
      <thread-priority>5</thread-priority>
    </managed-thread-factory>
  </resources>
  ...
</domain>
```

### アプリケーションの例

以下は、ManagedThreadFactoryを利用したアプリケーションの例です。

**[例 2.8] ManagedThreadFactoryを利用したアプリケーションの例**

```
public class AppServlet extends HTTPServlet implements Servlet {
    // Retrieve our executor instance.
    @Resource(name="mtf1")
    ManagedThreadFactory mtf;

    protected void doPost(HttpServletRequest req, HttpServletResponse
```

```

        resp) throws ServletException, IOException {
// 一般的な実行可能なタスク
Runnable simpleTask = new Runnable() {
    @Override
    public void run() {
        int sum = 0;
        for (int i = 0; i < 10; i++) { sum += i; }
        System.out.println(sum);
    }
};

// simpleTaskをManagedThreadFactoryで提供されるスレッドで実行
mtf.newThread(simpleTask).start();

// あるいはThreadPoolExecutorのパラメータでThreadFactoryを渡す
Executor e = new ThreadPoolExecutor(5, 10, 6L, TimeUnit.MINUTES, new
ArrayBlockingQueue<Runnable>(4096), mtf);
e.execute(new SimpleTask());
}
}

```

## WebAdminを利用した設定例

WebAdminで左側の[Resources]メニューの「Concurrency Utilities Resource」項目で ManagedThreadFactoryリソース設定を行えます。

【図 2.4】 WebAdminを利用したManagedThreadFactoryの設定

**Managed Thread Factory**

アプリケーションで使用できるManagedThreadFactory(管理対象スレッドファクトリ)を定義します。タスクがJava SEで提供される並列処理ユーティリティ(例: ExecutorServiceなど)で実行される場合でも、タスクのコンテキストを維持させます。並列処理ユーティリティを利用するときは、ManagedThreadFactoryを使用してスレッドを作成する必要があります。(例: java.util.ThreadPoolExecutorのコンストラクタパラメータでスレッドファクトリを渡すことが可能)

動的設定 \* 必須項目

Export Name *	<input type="text"/> ネーミングサーバに登録するManaged Thread Factoryの名前を設定します。
Thread Priority	<input type="text"/> [デフォルト: 5] スレッドの優先順位を設定します。(デフォルト値: 5)

確認 再設定