intra-mart WebPlatform/AppFramework Ver.7.0

ログ 設定ガイド

2011/06/30 第 8 版

<< 変更履歴 >>

変更年月日	変更内容
2008/07/07	初版
2008/09/30	第 2 版 「2.6 commons-logging、log4jの利用」の説明を追加・修正しました。
2008/10/31	第 3 版 「5.5 設定ファイルの新規作成」における「注意事項」を追記しました。
2009/02/27	第 4 版 「4.1 概要説明」のデータベースログとリクエストログの識別名、デフォルト値の誤植を修正しました。 「4.5.2.1 注意事項」の誤植を修正しました。
2009/06/30	第 5 版 「2.3.3.3.5.1 パラメータ」に、MinIndexとMaxIndexに指定可能な値の条件を追記しました。 「2.5 独自に作成したクラスの利用」を追記しました。 「2.6.2 Tips」に、SLF4Jブリッジ・ライブラリが配置されているディレクトリを追記しました。 「4.6 リクエストログ」に、リクエストを受け付けた際のログ出力方法を記載しました。 「4.8 画面遷移ログ」に、応答時間に関する説明を追記しました。 「5.3.1 概要説明」の設定ファイルについての記述説明（吹き出し）の欠落を修正しました。 「5.5.2 設定ファイルの作成」で、アペンダー名とロガー名が同一名になっている誤植を修正しました。
2009/10/30	第 6 版 「2.3.4.1 概要説明」 および 「2.3.4.5.2 Evaluatorの利用」の誤字を修正しました。 「4.5.3 データベースログの出力判定方式の設定」を追記しました。 「7.1.3 前提条件」 および 「7.2.3 前提条件」のログ出力先を訂正しました。
2010/11/30	第 7 版 「2.3.3.3.1.3 ExtendedTimeBasedRollingPolicyの設定と動作」の説明を修正しました。 「2.3.4 ContentTypeCharsetSMTPAppender」のjanino.jarをクラスパスに追加する方法を修正しました。 「5.6 設定ファイルのバックアップに関する注意点」を追記しました。
2011/06/30	第 8 版 「2.6.1 SLF4J経由でのログの出力」の記述を修正しました。

<< 目次 >>

1 はじめに 1
2 ログ共通機能 2
2.1 概要 2
2.1.1 SLF4J（Simple Logging Facade for Java） 2
2.1.2 Logback 2
2.2 Logger（ロガー） 3
2.2.1 ロガー名 3
2.3 Appender（アペンダー） 5
2.3.1 ConsoleAppender 5
2.3.2 FileAppender 5
2.3.3 RollingFileAppender 6
2.3.4 ContentTypeCharsetSMTPAppender 12
2.3.5 Tips 15
2.4 Layout（レイアウト） 16
2.4.1 PatternLayout 16
2.4.2 OutputStackTracePatternLayout 18
2.5 独自に作成したクラスの利用 19
2.5.1 クラスのデプロイ 19
2.6 commons-logging、log4jの利用 20
2.6.1 SLF4J経由でのログの出力 20
2.6.2 Tips 20
3 システムログ 21
3.1 概要説明 21
3.2 パラメータ 21
4 特定用途ログ 22
4.1 概要説明 22
4.2 詳細ログの出力 22
4.3 ネットワークログ 23
4.3.1 パラメータ 23
4.4 メモリログ 24
4.4.1 パラメータ 24
4.4.2 メモリ監視間隔（時間）の設定 24
4.5 データベースログ 25
4.5.1 パラメータ 25
4.5.2 パラメータ置換の設定 25
4.5.3 データベースログの出力判定方式の設定 26
4.6 リクエストログ 28
4.6.1 パラメータ 28
4.7 セキュリティログ 29
4.7.1 パラメータ 29
4.8 画面遷移ログ 30
4.8.1 パラメータ 30
5 設定ファイルと設定 32
5.1 設定ファイルの種類 32
5.2 設定ファイルの編集に関する注意点 32
5.3 im_logger.xml 33

5.3.1 概要説明 33
5.3.2 注意事項 33
5.4 特定用途ログの設定ファイル 34
5.4.1 概要説明 34
5.5 設定ファイルの新規作成 36
5.5.1 概要説明 36
5.5.2 設定ファイルの作成 37
5.5.3 注意事項 40
5.5.4 substitutionPropertyの利用 41
5.5.5 JMX経由による設定 42
5.6 設定ファイルのバックアップに関する注意点 44
6 ログのディレクトリ構造 45
6.1 Server Managerログディレクトリ構成 45
6.2 Application Runtimeログディレクトリ構成 46
6.3 Shared Memory Serviceログディレクトリ構成 47
6.4 Permanent Data Serviceログディレクトリ構成 48
6.5 Storage Serviceログディレクトリ構成 49
6.6 Resource Serviceログディレクトリ構成 50
6.7 Serialization Serviceログディレクトリ構成 51
6.8 Schedule Serviceログディレクトリ構成 52
7 サンプル 53
7.1 JavaEE開発モデル 53
7.1.1 作成するもの 53
7.1.2 使用するAPI 53
7.1.3 前提条件 53
7.1.4 ソースコード 53
7.1.5 設定ファイル 54
7.1.6 出力結果 54
7.2 スクリプト開発モデル 55
7.2.1 作成するもの 55
7.2.2 使用するAPI 55
7.2.3 前提条件 55
7.2.4 ソースコード 55
7.2.5 設定ファイル 55
7.2.6 出力結果 56
8 ログの解析 57
8.1 httpログ 57
8.2 サーバ実行ログ 57
8.3 ログ解析手順の例 57
8.4 注意事項 58
9 Ver6.1 ログ機能との相違点 59
9.1 ログ種別の変更 59
9.1.1 システムログへ統一 59
9.1.2 特定の用途を持つログを特定用途ログに変更 59
9.2 ログ出力に関する設定ファイルの変更 59
9.3 ログレベルの変更 60
9.4 ロガー単位のログ出力 60

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

9.5 ログの出力先変更 61
9.6 例外（Exception）ログの出力 61
9.7 操作ログの廃止 61
9.8 ログパラメータ名称の変更 61
9.8.1 ネットワークログの比較 61
9.8.2 メモリログの比較 61
9.8.3 データベースログの比較 62
9.8.4 リクエストログの比較 62
9.8.5 セキュリティログの比較 62
9.8.6 画面遷移ログの比較 63
10 サポート 64
11 索引 65

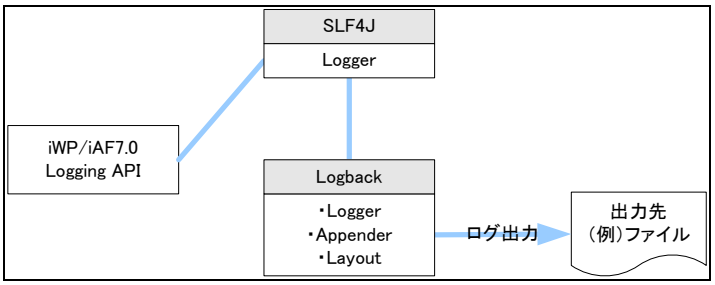
1 はじめに

本ドキュメントは、intra-mart WebPlatform/AppFramework のログ機能に関する設定手引書です。
ログの機能と種類ならびに、各ログに共通な設定項目、設定について解説します。

2 ログ共通機能

2.1 概要

intra-mart WebPlatform / intra-mart AppFramework Ver.7.0 では、
「Logback（http://logback.qos.ch/）」と「SLF4J（http://www.slf4j.org/）」を利用してログの出力機構を実現しています。



2.1.1 SLF4J（Simple Logging Facade for Java）

Logback のロギング実装をラップし、共通のロギング API として利用されます。

ライブラリ	概要
slf4j-api-1.5.0.jar	SLF4J の本体

※ バージョンは、2008 年 06 月 30 日現在

2.1.2 Logback

SLF4J の実装（ログ出力機能）として利用されます。

ライブラリ	概要
logback-core-0.9.9.jar	Logback の本体
logback-classic-0.9.9.jar	Logback と slf4j のブリッジライブラリ

※ バージョンは、2008 年 06 月 30 日現在

Logback には、主に Logger、Appender、Layout の 3 つのタイプがあり、それぞれの特徴について記載します。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

2.2 Logger（ロガー）

Logger は以下の役割を担います。

- ログの出力を行います。
- メソッドに引渡すパラメータを利用したログの出力を可能とします。

（例）

```
String parm = "パラメータ A";
String result = "false";
Object [] arg = { parm, value };
logger.info( "{} の値は {} です。", arg );
```

- 以下5段階のログレベルを指定したログの出力を可能とします。

ログレベル	出力範囲	重要度
ERROR	ERROR	↑
WARN	ERROR、WARN	高い
INFO	ERROR、WARN、INFO	
DEBUG	ERROR、WARN、INFO、DEBUG	低い
TRACE	ERROR、WARN、INFO、DEBUG、TRACE	↓

- ドット「.」で区切られた階層構造でロガー（logger）名を指定することで、ログレベルの継承を可能とします。

2.2.1 ロガー名

ロガー（logger）名とは、logger を特定（識別）するための名称となります。
ロガー名は、階層構造を持たせることが可能となっており、ドット「.」で区切られたパッケージ名やクラス名を指定することで、親階層からログレベルを継承するこができます。

（例）

「X.Y」という名称の Logger があり、その下の階層に「Z」という名前の Logger を作成したい場合、
「X.Y.Z」という名称の Logger となります。

＜階層構造を持つ logger とその出力範囲＞

（例 1）

Logger 名	割り当てられたログレベル	有効となるログレベル
root	DEBUG	DEBUG
X	-	DEBUG
X.Y	-	DEBUG
X.Y.Z	-	DEBUG

解説

親となる root にのみログレベルが割り当てられた場合、
その子にあたる「X」、「X.Y」、「X.Y.Z」は root に割り当てられたログレベルが有効になります。

（例 2）

Logger 名	割り当てられたログレベル	有効となるログレベル
root	ERROR	ERROR

X	INFO	INFO
X.Y	DEBUG	DEBUG
X.Y.Z	WARN	WARN

解説

全てのロガーにログレベルが割り当てられた場合、
それぞれに割り当てられたログレベルが有効になります。

（例 3）

Logger 名	割り当てられたログレベル	有効となるログレベル
root	DEBUG	DEBUG
X	INFO	INFO
X.Y	-	INFO
X.Y.Z	ERROR	ERROR

解説

「X.Y」にのみログレベルが割り当てられなかった場合、
「X.Y」の直近の親階層で、且つログレベルが割り当てられている「X」のログレベルが有効になります。

（例 4）

Logger 名	割り当てられたログレベル	有効となるログレベル
root	DEBUG	DEBUG
X	INFO	INFO
X.Y	-	INFO
X.Y.Z	-	INFO

解説

「X.Y」、「X.Y.Z」にログレベルが割り当てられなかった場合、
「X.Y」、「X.Y.Z」の直近の親階層で、且つログレベルが割り当てられている「X」のログレベルが有効になります。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

2.3 Appender（アペンダー）

Appender は以下の役割を担います。

- ログの出力先を決定します。

アペンダーには以下のものが提供されています。

アペンダーの種類	出力先
ConsoleAppender	コンソール
FileAppender / RollingFileAppender	ファイル
ContentTypeCharsetSMTPAppender	メール

- Logger に紐付き、その Logger の出力先を指定することが可能となります。

これにより、Logger 単位で出力先の指定が可能となります。

2.3.1 ConsoleAppender

2.3.1.1 概要説明

ConsoleAppender はログをコンソールに出力するためのアペンダーです。

2.3.1.2 Appenderクラス

クラス名	ch.qos.logback.core.ConsoleAppender

2.3.1.3 パラメータ

パラメータ名称	形式	説明	デフォルト値
Encoding	String	ログ出力で使用するエンコーディング	システムエンコーディング
ImmediateFlush	boolean	書き込みバッファの内容を即時に反映（出力）させるかどうかの設定	true
Target	String	出力方式を設定します。 System.out：標準出力 System.err：標準エラー出力	System.out

2.3.2 FileAppender

2.3.2.1 概要説明

FileAppender はログをファイルに出力するためのアペンダーです。

このアペンダーではログファイルのローテイトは行えません。

2.3.2.2 Appenderクラス

クラス名	ch.qos.logback.core.FileAppender

2.3.2.3 パラメータ

パラメータ名称	形式	説明	デフォルト値
Append	boolean	追記モード	true

		true：追加モード false：既存のログを上書きして出力	
Encoding	String	ログ出力で使用するエンコーディング	システムエンコーディング
BufferedIO	boolean	バッファリング	false
BufferSize	int	バッファリングをする場合のバッファサイズ	8192 バイト
File	String	出力するログファイル名	---
ImmediateFlush	boolean	書き込みバッファの内容を即時に反映（出力）させるかどうかの設定	true

2.3.3 RollingFileAppender

2.3.3.1 概要説明

RollingFileAppender は FileAppender を拡張したアペンダーです。
ローテイト条件を指定することで、ログファイルのローテイトを行うことが可能となります。

2.3.3.2 Appenderクラス

クラス名	ch.qos.logback.core.rolling.RollingFileAppender

2.3.3.3 RollingPolicy

ローテイト方法の設定です。ローテイトの契機は、時間ベースでのローテイト（ExtendedTimeBasedRollingPolicy）と、ファイルサイズでのローテイト（SizeBasedTriggeringPolicy）の 2 種類が用意されています。

*ローテイト方法に時間ベースでのローテイト（**ExtendedTimeBasedRollingPolicy**）とファイルサイズでのローテイト（**SizeBasedTriggeringPolicy**）を同時に指定することはできません。*
どちらか一方を選択してください。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

2.3.3.3.1 ExtendedTimeBasedRollingPolicy

時間ベースでローテイトする設定です。

ExtendedTimeBasedRollingPolicy は、Logback 標準の TimeBasedRollingPolicy を拡張したローテイトクラスです。

Logback 標準の TimeBasedRollingPolicy は、intra-mar サーバが起動した状態で、ローテイト条件満たした場合のみ実行されます。
そのため、システム日付を変更して、intra-mart サーバの再起動を行った直後にはローテイトは行われず、
次回ローテイト条件と一致した際にローテイトが行われることになります。
ExtendedTimeBasedRollingPolicy を利用した場合、再起動時にローテイト条件を満たしていた場合には、次回のローテイト条件を待たずにログのローテイトが行われます。

(例)サーバの再起動とローテイトの関係 （ローテイト条件：毎時）

システム時間	処理	TimeBasedRollingPolicy	ExtendedTimeBasedRollingPolicy
17:15	*サーバを起動*	-	-
17:30	ログ出力	*foo.log* ・・・ 17:30 のログ	*foo.log* ・・・ 17:30 のログ
17:45	*システム時間を 20:00 に変更* *サーバ再起動*	-	-
20:15	ログ出力	*foo.log* ・・・ 17:30 と 20:15 のログ ローテイトは行われない。	*foo.log* ・・・20:15 のログ *foo-yyyy-MM-dd_17.log* ・・・ 17:30 のログ ローテイトが行われる。
21:15	ログ出力	*foo.log* ・・・ *21:15* のログ *foo-yyyy-MM-dd_20.log* ・・・ 17:30 と 20:15 のログ ローテイトが行われる。	*foo.log* ・・・ *21:15* のログ *foo-yyyy-MM-dd_17.log* ・・・ 17:30 のログ *foo-yyyy-MM-dd_20.log* ・・・ 20:15 のログ ローテイトが行われる。

2.3.3.3.1.1 クラス

クラス名	jp.co.intra_mart.common.platform.log.rolling.ExtendedTimeBasedRollingPolicy

2.3.3.3.1.2 パラメータ

パラメータ名称	形式	説明	デフォルト値
FileNamePattern	String	ローテイト時のファイルパターン ファイルパターンには%d{日付フォーマット}を指定することが必須となります。	---

2.3.3.3.1.2.1 バックアップファイルの圧縮

FileNamePattern には以下圧縮形式を指定することは可能です。

ファイルの圧縮形式
zip
gz

（例）

```
<FileNamePattern>tests. %d..log.zip</FileNamePattern>
```

2.3.3.3.1.3 ExtendedTimeBasedRollingPolicyの設定と動作

（ローテイト条件：毎分の場合）

im_logger.xml

```
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>${im.home}/log/platform/foo.log</file>
    <rollingPolicy  class="jp.co.intra_mart.common.platform.log.rolling.ExtendedTimeBasedRollingPolicy">
        <FileNamePattern>
            ${im.home}/log/platform/foo-%d{yyyy-MM-dd-HH-mm}.log
        </FileNamePattern>
    </rollingPolicy>
…
```

最新のログは *foo.log* ファイルに出力され、古いログは新たな *foo.log* が生成されるタイミングで *foo*- yyyy-MM-dd-HH-mm.*log* といった日付（毎分）を名称に持つバックアップファイルに移されます。

	実行時間	ログファイル	ログの内容
1	2008/06/01｜12：00	*foo.log*	*foo.log* が生成され、Step1 のログが出力。
2	2008/06/01｜12：01	*foo.log* ***foo-2008-06-01-12-00.log***	Step1 の *foo.log* が *foo-2008-06-01-12-00.log* にリネーム。 *foo.log* が生成され、Step2 のログが出力。
3	2008/06/01｜12：02	*foo.log* *foo-2008-06-01-12-00.log* ***foo-2008-06-01-12-01.log***	Step2 の *foo.log* が *foo-2008-06-01-12-01.log* にリネーム。 *foo.log* が生成され、Steps3 のログが出力。
4	2008/06/01｜12：03	*foo.log* *foo-2008-06-01-12-00.log* *foo-2008-06-01-12-01.log* ***foo-2008-06-01-12-02.log***	Step3 の *foo.log* が *foo-2008-06-01-12-02.log* にリネーム。 *foo.log* が生成され、Step4 のログが出力。
5	2008/06/01｜12：10	*foo.log* *foo-2008-06-01-12-00.log* *foo-2008-06-01-12-01.log* *foo-2008-06-01-12-02.log* ***foo-2008-06-01-12-03.log***	Step4 の *foo.log* が *foo-2008-06-01-12-03.log* にリネーム。 *foo.log* が生成され、Step5 のログが出力。

（ローテイト：毎日の場合）

im_logger.xml

```
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${im.home}/log/platform/foo.log</file>

        <rollingPolicy class="jp.co.intra_mart.common.platform.log.rolling.ExtendedTimeBasedRollingPolicy">
                <FileNamePattern>
                    ${im.home}/log/platform/foo-%d{yyyy-MM-dd}.log
                </FileNamePattern>
        </rollingPolicy>
…
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

最新のログは *foo.log* ファイルに出力され、古いログは新たな *foo.log* が生成されるタイミングで
foo- yyyy-MM-dd.*log* といった日付（毎日）を名称に持つバックアップファイルに移されます。

	実行した日付	ログファイル	ログの内容
1	2008/06/01｜01:00	*foo.log*	*foo.log* が生成され、Step1 のログが出力。
2	2008/06/02｜01:00	*foo.log* ***foo-2008-06-01.log***	Step1 の *foo.log* が *foo-2008-06-01.log* にリネーム。 *foo.log* が生成され、Steps2 のログが出力。
3	2008/06/03｜01:00	*foo.log* *foo-2008-06-01.log* ***foo-2008-06-02.log***	Step2 の *foo.log* が *foo-2008-06-02.log* にリネーム。 *foo.log* が生成され、Step3 のログが出力。
4	2008/06/04｜01:00	*foo.log* *foo-2008-06-01.log* *foo-2008-06-02.log* ***foo-2008-06-03.log***	Step3 の *foo.log* が *foo-2008-06-03.log* にリネーム。 *foo.log* が生成され、Step4 のログが出力。
5	2008/06/10｜01:00	*foo.log* *foo-2008-06-01.log* *foo-2008-06-02.log* *foo-2008-06-03.log* ***foo-2008-06-04.log***	Step4 の *foo.log* が *foo-2008-06-04.log* にリネーム。 *foo.log* が生成され、Step5 のログが出力。

2.3.3.3.2 SizeBasedTriggeringPolicy

指定したファイルサイズでローテイトする設定です。以下にその動作を解説します。

ファイルサイズでローテイトを行う場合には、ローテイト時のファイルパターンを定義する「FixedWindowRollingPolicy」オプションも併せて設定する必要があります。

2.3.3.3.3 クラス

クラス名	ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy

2.3.3.3.3.1 パラメータ

パラメータ名称	形式	説明	デフォルト値
MaxFileSize	int	ローテイトサイズ	10MB

ローテイト条件となるファイルサイズに指定可能な単位

単位	KB （キロバイト）
	MB （メガバイト）
	GB （ギガバイト）

2.3.3.3.4 FixedWindowRollingPolicy

ローテイトの際にインデックスベースのバックアップファイル名を生成する設定です。

SizeBasedTriggeringPolicy を利用する場合に、併せて設定します。

2.3.3.3.5 クラス

クラス名	ch.qos.logback.core.rolling.FixedWindowRollingPolicy

2.3.3.3.5.1 パラメータ

パラメータ名称	形式	説明	デフォルト値
MinIndex	int	ファイルインデックスの最小値	---
MaxIndex	int	ファイルインデックスの最大値	---
FileNamePattern	String	ローテイト時のファイルパターン ファイルパターンには「%i」を指定することが必須となります。	---

生成可能なバックアップファイル数は、***最大 12 ファイルまで***となります。それ以上バックアップファイルを生成することはできません。

そのため、パラメータ「MinIndex」と「MaxIndex」には、以下条件を満たす値を設定して下さい。

MaxIndex － MinIndex ＜ 12

※ 上記の条件を満たさない値が設定された場合には、MinIndex + 12 の値が MaxIndex として適用されます。

（例）

```
<rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
        <FileNamePattern>${im.home}/log/platform/foo%i.log</FileNamePattern>
                <MinIndex>5</MinIndex>
                <MaxIndex>30</MaxIndex>
</rollingPolicy>
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

上記の設定の場合、バックアップファイル数の上限により、実際に生成されるバックアップファイルは以下のとおりとなります。

File Name	
foo.log	カレントのログファイル
foo5.log	↑
foo6.log	*new*
foo7.log	
foo8.log	
foo9.log	
foo10.log	
foo11.log	
foo12.log	
foo13.log	
foo14.log	
foo15.log	*old*
foo16.log	↓

2.3.3.3.5.1.1 バックアップファイルの圧縮

FileNamePattern には以下圧縮形式を指定することは可能です。

ファイルの圧縮形式
zip
gz

（例）

```
<FileNamePattern>tests.%i.log.zip</FileNamePattern>
```

2.3.3.3.5.2 SizeBasedTriggeringPolicy とFixedWindowRollingPolicyの設定例

（ローテイト:ファイルサイズ 10M でのローテイト）

```
<rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
        <FileNamePattern>${im.home}/log/platform/foo%i.log</FileNamePattern>
                <MinIndex>0</MinIndex>
                <MaxIndex>5</MaxIndex>
</rollingPolicy>

<triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
        <MaxFileSize>10MB</MaxFileSize>
</triggeringPolicy>
・・・
```

サイズローテイトによりリネームされたされたファイル

File Name	
foo.log	カレントのログファイル
foo0.log	↑
foo1.log	*new*
foo2.log	
foo3.log	
foo4.log	*old*
foo5.log	↓

2.3.4 ContentTypeCharsetSMTPAppender

2.3.4.1 概要説明

ContentTypeCharsetSMTPAppender は、Logback 標準の SMTPAppender を拡張したアペンダーです。
このアペンダーを利用することで、ログを電子メールで受け取ることができるようになります。

※ デフォルトは**ログレベルが ERROR の場合のみメールを送信します**。

Logback 標準の SMTPAppender との違いは、ContentTypeCharsetSMTPAppender を利用した場合には、ヘッダーの"Content-type: charset"の文字コードの指定及び、SMTP サーバホストの複数指定が可能となります。

ContentTypeCharsetSMTPAppender は循環バッファからメッセージを送信するため、ERROR レベルのログイベントが発生した際には、ERROR 発生以前のログも送信されます。
循環バッファのサイズはパラメータ「BufferSize」で指定することが可能です。

2.3.4.2 Appenderクラス

クラス名	jp.co.intra_mart.common.platform.log.appender.ContentTypeCharsetSMTPAppender

2.3.4.3 前提条件

- **ユーザ ID、パスワードによる認証（SMTP-AUTH、POP before SMTP など）を必要とするメール送信はできません。**
  ユーザ ID とパスワードによる認証を必要としないメールサーバを利用してください。
- ContentTypeCharsetSMTPAppender を利用する場合には、以下のライブラリが必要です。

最新モジュールは下記 URL よりダウンロードすることが出来ます。（2008 年 6 月 30 日現在）

ライブラリ	ダウンロード URL
JavaMail 1.4.1	http://java.sun.com/products/javamail/index.jsp
JAF 1.1.1（JavaBeans Activation Framework）	http://java.sun.com/products/javabeans/jaf/downloads/index.html
Janino 2.5.14	http://janino.net/index.html

Janino のライブラリは、ServicePlatfrom 起動時のクラスパスに追加する必要があります。
以下に、Janino のライブラリ「janino.jar」を ServicePlatfrom 起動時のクラスパスに追加する方法を示します。

Janino のライブラリを「<%im_path%>/janino/lib/ janino.jar」に解凍した場合、
<%im_path%>/conf/imart.xml は、以下のように設定してください。

```
<compiler>
    <class>
        <archive>
            <file/>
            <directory>janino/lib</directory>
        </archive>
    </class>
</compiler>
...
```

※ JavaMail と JAF は、intra-mart WebPlatform/AppFramework インストール時に同梱されており、別途インストールする必要はありません。（同梱されるライブラリのバージョンに関してはリリースノートを参照して下さい）

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

2.3.4.4 パラメータ

パラメータ名称	形式	説明	初期値	必須
SMTPHost	String	SMTP サーバのホスト名 タグを複数指定することで、SMTP サーバのクラスタリングが可能となります。	---	○
To	String	宛先のメールアドレス タグを複数指定することで、複数の宛先に送信することが可能となります。	---	---
From	String	From ヘッダーの値。「差し出し人」	ユーザ ID + ホスト名	---
Subject	String	Subject ヘッダーの値。「タイトル」	logger 名 ＋ メッセージ	---
Charset	String	本文の文字コード ヘッダーの“Content-type: charset”の文字コードを指定することが可能となります。	---	---
BufferSize	String	循環バッファサイズ	512	---
Evaluator	String	Evaluator クラス名 Evaluator クラスを作成することで、メールの送信対象とするログレベル判定などを行うこと可能となります。	---	---

2.3.4.5 ContentTypeCharsetSMTPAppenderの設定例

im_logger.xml

```
<appender name="EMAIL"
  class="jp.co.intra_mart.common.platform.log.appender. ContentTypeCharsetSMTPAppender">
    <SMTPHosts>
        <SMTPHost>192.XXX.XXX.001</SMTPHost>
        <SMTPHost>192.XXX.XXX.002</SMTPHost>
        <SMTPHost>192.XXX.XXX.003</SMTPHost>
    </SMTPHosts>
    <to>sample02@localhost</to>
    <from>sample01@localhost</from>
    <charset>ISO-2022-JP</charset>
    <layout class="ch.qos.logback.classic.PatternLayout">
        <Pattern>%date %-5level %logger{35} - %message%n</Pattern>
    </layout>
</appender>

<root>
  <level value="ERROR" />
  <appender-ref ref="EMAIL" />
</root>
・・・
```

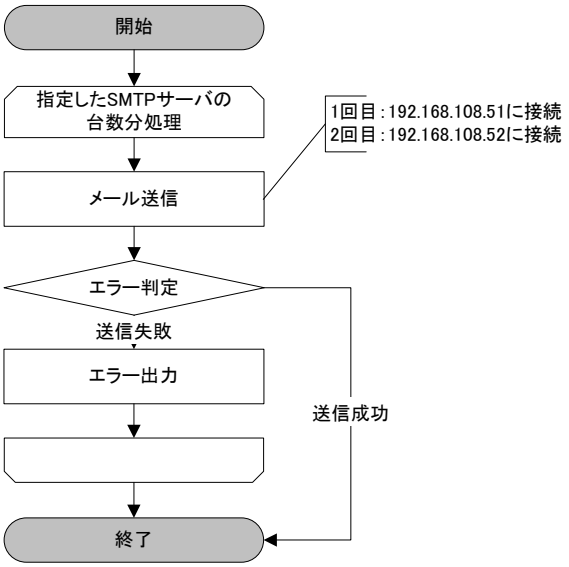
2.3.4.5.1 SMTPサーバのホスト名を複数指定した場合の動作

<SMTPHosts/>に複数の SMTP サーバのホスト名を指定した場合には、以下の動作仕様に準拠します。

ログ出力設定ファイル

```
    <SMTPHosts>
        <SMTPHost>192.168.108.51</SMTPHost>
        <SMTPHost>192.168.108.52</SMTPHost>
    </SMTPHosts>
```

＜メール送信処理フロー＞



2.3.4.5.2 Evaluatorの利用

ContentTypeCharsetSMTPAppender はデフォルトで ERROR レベルのログのみを送信しますが、
Evaluator クラスを作成することで、送信対象となるログレベルを独自に判定するなどの処理を実装することが可能となります。
以下に利用例を記載します。

手順1

Evaluator クラスを作成します。
ここでは、送信対象となるログレベル WARN レベルとする Evaluator クラスを作成します。

継承するクラス	ch.qos.logback.core.spi.ContextAwareBase
インタフェース	ch.qos.logback.core.boolex.EventEvaluator

（例） sample.log. evaluator. LogLevelJudgmentEvaluator.java

```
package sample.log. evaluator;

import ch.qos.logback.classic.Level;
import ch.qos.logback.classic.spi.LoggingEvent;
import ch.qos.logback.core.boolex.EvaluationException;
import ch.qos.logback.core.boolex.EventEvaluator;
import ch.qos.logback.core.spi.ContextAwareBase;

public class LogLevelJudgmentEvaluator extends ContextAwareBase implements EventEvaluator {

  public boolean evaluate(Object event) throws NullPointerException,
            EvaluationException {

        LoggingEvent e = (LoggingEvent)event;

        // WARN レベルのログのみ送信
        if(e.getLevel().equals(Level.WARN)){
              return true;
        }else{
              return false;
        }
  }
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

```
                                                    ・
                                                    ・
```

手順2

作成した Evaluator クラスを設定ファイルに定義します。

（例） im_logger.xml

```
<appender name="EMAIL" class="jp.co.intra_mart.common.platform.log.appender.ContentTypeCharsetSMTPAppender">
    <Evaluator class="sample.log. evaluator. LogLevelJudgmentEvaluator" />
    <SMTPHosts>
        <SMTPHost> host1 </SMTPHost>
        <SMTPHost> host2 </SMTPHost>
        <SMTPHost> host3 </SMTPHost>
    </SMTPHosts>
                                                    ・
                                                    ・
                                                    ・
```

※（例）では、im_logger.xml に Evaluator クラスの定義を行っていますが、
実環境においては、ContentTypeCharsetSMTPAppender ならびに Evaluator クラスを利用したいログの制御ファイル（ログ設定ファイル）に定義してください。

2.3.5 Tips

Tips *「DBAppender」*

当ドキュメントに記載した Appender 以外に、
Logback では、接続したデータベース上の専用テーブルにログを格納する「DBAppender」が提供されています。
ただし、パフォーマンス上の問題などから、intra-mart WebPlatform/AppFramework では「DBAppender」の利用は非推奨となります。

「DBAppender」の詳細については、Logback のホームページ（http://logback.qos.ch/）よりご確認ください。

2.4 Layout（レイアウト）

Layput は以下の役割を担います。

- Appender に紐付き、その Appender のログ出力フォーマットの指定を可能とします。これにより、Appender 単位でのフォーマットの指定が可能となります。

（例）

＜出力項目＞

- ログレベル
- スレッド名
- ロガー名
- メッセージ

＜フォーマット＞

<pattern> %level %thread %logger{255} %msg%n </pattern>

2.4.1 PatternLayout

様々な形式でログを出力することを可能としたレイアウトの基本パターンです。

パターン文字列で指定したフォーマットにしたがって、ログが出力されます。

2.4.1.1 Layoutクラス

クラス名	ch.qos.logback.classic.PatternLayout

2.4.1.2 パターン文字列

<table>
<tr><th>パターン文字列</th><th colspan="3">説明</th></tr>
<tr><td rowspan="5">c{length}
lo{length}
logger{length}</td><td colspan="3">ログイベントのロガー名
{length}を指定することで、出力するロガー名の長さを制限することができます。
（略語アルゴリズムに準拠）</td></tr>
<tr><td>変換パターン</td><td>ロガー名</td><td>出力結果</td></tr>
<tr><td>%logger</td><td>mainPackage.sub.sample.Bar</td><td>mainPackage.sub.sample.Bar</td></tr>
<tr><td>%logger{10}</td><td>mainPackage.sub.sample.Bar</td><td>m.s.s.Bar</td></tr>
<tr><td>%logger{15}</td><td>mainPackage.sub.sample.Bar</td><td>m.s.sample.Bar</td></tr>
<tr><td rowspan="5">d{pattern}
date{pattern}</td><td colspan="3">出力日付</td></tr>
<tr><td colspan="2">変換パターン</td><td>出力結果</td></tr>
<tr><td colspan="2">%date</td><td>2006-10-20 14:46:49,812</td></tr>
<tr><td colspan="2">%date{ISO8601}</td><td>2006-10-20 14:46:49,812</td></tr>
<tr><td colspan="2">%date{HH:mm:ss.SSS}</td><td>14:46:49.812</td></tr>
<tr><td colspan="2">%date{dd MMM yyyy ;HH:mm:ss.SSS}</td><td>20 oct. 2006;14:46:49.812</td></tr>
<tr><td>m / msg / message</td><td colspan="3">ログメッセージ</td></tr>
<tr><td>p / le / level</td><td colspan="3">ログレベル</td></tr>
<tr><td>t / thread</td><td colspan="3">ログを生成したスレッド名</td></tr>
<tr><td>X{key}
mdc{key}</td><td colspan="3">ログが生成されたスレッドの MDC</td></tr>
<tr><td>nopex
nopexception</td><td colspan="3">例外出力の抑止
例外を出力しないようになります。</td></tr>
</table>

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

2.4.1.2.1 MDCの利用

PatternLayout で提供されているパラメータ「X{key}、mdc{key}」を利用することにより、独自に定義した key で保存した情報をログに出力することが可能となります。
MDC に保存された情報はスレッドローカル変数として扱われます。

(例)
ユーザアプリケーションで定義した作業区分「work_division」をログに出力する場合

・ プログラム

```
// 「user_application_key」情報の初期化
MDC.remove("work_division");

// MDC に情報を保存
MDC.put("work_division", "作業区分 0001");

// ログの出力
Logger.info("処理を終了しました。");
```

・ 設定ファイル「user_logger_xxx.xml」

```
<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
  <layout class="ch.qos.logback.classic.PatternLayout">
    <pattern>[%level] %logger{10} - %X{work_division} %msg%n</pattern>
  </layout>
</appender>
```

2.4.1.2.2 MDC利用時の注意事項

MDC を利用する際は、以下の点に注意してください。

- MDC に保存した内容は、明示的に初期化が行われない限り、情報（値）が初期化されることはありません。そのため、必要に応じて初期化処理（MDC.remove(key)）を実装する必要があります。
- intra-mart WebPlatform/AppFramework では、製品の標準ログ出力機能にて MDC を利用しています。そのため、同一の key を使用した場合、正常にログが出力されなくなります。
  以下の key は使用しないでください。

KEY 名称		
client.session.id	request.remote.host	sapconnector.id.time
db.connection.identifier	request.remote.address	security.id.session
db.connection.res	request.url	security.id.account
db.sql	request.url.referer	security.id.group
db.time	request.page.time	security.id.logintype
log.id	request.accept.time	transition.log.type.id
log.report.sequence	sapconnector.id.identifier	transition.access.user.id
log.thread.group	sapconnector.im.id.account	transition.path.page.next
memory.free	sapconnector.im.id.group	transition.time.response
memory.total	sapconnector.sap.id.host	transition.exception.name
memory.initial	sapconnector.sap.id.client	transition.exception.message
memory.max	sapconnector.sap.id.user	transition.path.page.previous
request.id	sapconnector.id.bapiname	

2.4.1.3 Layoutの設定例

im_logger.xml

```
<configuration>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <layout class="ch.qos.logback.classic.PatternLayout">
      <pattern>[%level] %logger{10} - %msg%n</pattern>
    </layout>
  </appender>

  <root>
    <level value="debug" />
    <appender-ref ref="STDOUT" />
  </root>
</configuration>
```

＜ログの出力例＞

```
10489 DEBUG [main] com.marsupial.Pouch - Hello world
                    ・
                    ・
                    ・
```

2.4.2 OutputStackTracePatternLayout

PatternLayout を拡張したレイアウトです。

利用可能なパターン文字列及び利用方法は PatternLayout 同様です。

PatternLayout との違いは、このレイアウトを使用することで、通知された例外情報を Exception ログとして、別ファイルに出力します。

製品が標準で提供しているログ（システムログ、特定用途ログ）では、デフォルトで OutputStackTracePatternLayout が設定されています。

出力されたログ ID により、システムログ、特定用途ログと紐付けることが可能となります。

2.4.2.1 Layoutクラス

クラス名	jp.co.intra_mart.common.platform.log.layout.OutputStackTracePatternLayout

2.4.2.2 パラメータ

パラメータ名称	形式	説明	デフォルト値
enableOutputStackTrace	boolean	Exception ログを出力するかどうかの設定	true
stackTraceDir	String	Exception ログの出力ディレクトリ	${im.home}/log/platform/exception/
stackTraceFilename	String	Exception ログのファイルパターン	'exception_'yyyy-MM-dd_HH-mm-ss'_%logId.log'

2.4.2.3 OutputStackTracePatternLayoutの設定例

im_logger.xml

```
<layout class="jp.co.intra_mart.common.platform.log.layout.OutputStackTracePatternLayout">
  <pattern>[%d{yyyy-MM-dd HH:mm:ss.SSS}]     %-5level  %logger{255}   %X{log.id} -    %msg%nopex%n</pattern>

  <enableOutputStackTrace>true</enableOutputStackTrace>
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

```
  <stackTraceDir>${im.home}/log/platform/exception/</stackTraceDir>
  <stackTraceFilename>'exception_'yyyy-MM-dd_HH-mm-ss'_%logId.log'</stackTraceFilename>
</layout>
・・・
```

2.5 独自に作成したクラスの利用

ログ機能では、Appender、Layout クラスを独自に作成（拡張）することが可能となっています。

Appender や Layout を独自に作成した場合には、以下の手順にしたがってデプロイする必要があります。

2.5.1 クラスのデプロイ

Appender や Layout を独自に作成した場合には、アーカイブ（jar、zip）、アーカイブディレクトリ、またはクラスファイルを配置したディレクトリを Java のクラスパスに設定する必要があります。

※　設定変更後はサーバの再起動が必要です。

設定ファイル	imart.xml

2.5.1.1 アーカイブをクラスパスに設定する場合

キー名称	intra-mart/platform/java/compiler/class/archive/file

<imart.xml>

```
<compiler>
    <class>
        <archive>
            <file>log_classes/log_sample.jar</file>
        </archive>
    </class>
</compiler>
```

2.5.1.2 アーカイブを配置したディレクトリをクラスパスに設定する場合

キー名称	intra-mart/platform/java/compiler/class/archive/directory

<imart.xml>

```
<compiler>
    <class>
        <archive>
            <directory>log_classes</directory>
            <file/>
        </archive>
    </class>
</compiler>
```

2.5.1.3 クラスファイルを配置したディレクトリをクラスパスに設定する場合

キー名称	intra-mart/platform/java/compiler/class/path

<imart.xml>

```
<compiler>
    <class>
        <path>log_classes</path>
        <archive>
            <file/>
        </archive>
    </class>
</compiler>
```

2.6 commons-logging、log4jの利用

2.6.1 SLF4J経由でのログの出力

intra-mart ver6.1(以下 IMv6.1)にて、ユーザアプリケーションプログラムで commons-logging 及び、log4j を利用したログの出力を行っていた場合も、SLF4J のブリッジライブラリ(jcl104-over-slf4j-xxx.jar、log4j-over-slf4j-xxx.jar)を利用することで、ユーザアプリケーション側のプログラムを変更することなく、intra-mart ver7.0(以下 IMv7.0)でログの出力を行うことが可能となります。

➢ 設定ファイルの作成

① conf/logディレクトリ配下にIMv7.0 用のログ設定ファイル(xml)を新たに作成します。

② IMv6.1 でcommons-logging.propertiesやlog4j.propertiesに定義されていた出力設定を、作成した IMv7.0 用のログ設定ファイル(xml)に定義します。

※ 設定ファイルの作成方法(xml)、および、フォーマット等については、「5.5 設定ファイルの新規作成」を参照してください。

➢ ユーザアプリケーションプログラムの配置

① IMv6.1 で利用していたユーザアプリケーションプログラムを配置します。

上記対応を行なった場合、SLF4J のブリッジライブラリを経由して、Logback でログの出力が行われるため、IMv6.1 で使用していた以下のライブラリ及び設定ファイルを配置する必要はありません。

- commons-logging のライブラリ(commons-logging.jar、commons-logging-api.jar)
- commons-logging.properties
- log4j のライブラリ(log4j-xxx.jar)
- log4j.properties

2.6.2 Tips

Tips *「SLF4Jを経由しないログの出力」*

以下手順では、SLF4J のブリッジライブラリを使用せずに、IMv6.1 と同様の手法でログの出力を行うことが可能となっています。SLF4J のブリッジライブラリを使用しない場合には、commons-logging または、log4j のライブラリの管理、ならびに、設定ファイルの管理は独自に行う必要があります。

➢ 以下の SLF4J ブリッジ・ライブラリの利用を中止(削除)して、既に利用している commons-logging または、log4j のライブラリ(jar)をクラスパスに追加

＜SLF4J ブリッジ・ライブラリ＞
commons-logging ・・・ jcl104-over-slf4j-xxx.jar
log4j ・・・ log4j-over-slf4j-xxx.jar

※ 上記 SLF4J ブリッジ・ライブラリは、以下のディレクトリに存在しています。
- Application Runtime/doc/imart/WEB-INF/lib 配下
- Server Manager/bin ＋ 各 Service Platform/bin 配下

※ SLF4J ブリッジ・ライブラリの利用中止(削除)以外は、設定ファイルの配置等、IMv6.1 での利用方法と同様となります。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

3 システムログ

3.1 概要説明

ログ API を利用して出力される汎用的なログです。
具体的には、以下の内容がシステムログとして出力されます。
※ 全て同一のログファイル（system.log）に出力されます。

- サーバの運用状態を通知するためのログ
  サーバの状態をトレースするために必要な情報が出力されます。
- エラー発生時のログ
  サーバ運用中に発生した様々なエラーが出力されます。
- アプリケーション共通マスタのインポート／エクスポートのバッチの実行状況に関するログ
  インポートまたはエクスポート対象のファイル、コミット件数に関する情報が出力されます。
- バッチプログラムの実行状況に関するログ
  バッチプログラムが正しく実行出来たかどうかについて出力されます。

3.2 パラメータ

設定ファイル	im_logger.xml	
出力	ファイル	system.log
	コンソール	
ログレベルの初期値	info	

フォーマット文字列	説明	初期値
%date	日時	○
%X{log.id}	ログ ID	○
%thread	メッセージ通知したスレッドID	---
%X{log.thread.group}	メッセージ通知したスレッドグループ	---
%msg	メッセージ	○

4 特定用途ログ

4.1 概要説明

特定の用途に使用されるログです。
特定用途ログについては、ログごとに専用の識別名が用意されており、ロガー名の先頭に必ず識別名が設定されます。
（例）
DEBUG DB_LOG.jp.co.intra_mart.framework.base.data.LoginGroupDBConnector

具体的には、以下のログが特定用途ログに該当します。

- ネットワークログ
- メモリログ
- データベースログ
- リクエストログ
- セキュリティログ
- 画面遷移ログ

デフォルト値（有効 / 無効）は、ログの種類によって異なります。

ログ種別	識別名	デフォルト値
ネットワークログ	NETWORK_LOG	off
メモリログ	MEMORY _LOG	off
データベースログ	DB _LOG	off
リクエストログ	REQUEST _LOG	info
セキュリティログ	SECURITY _LOG	error
画面遷移ログ	TRANSITION _LOG	info

4.2 詳細ログの出力

ログの出力レベル［DEBUG］に関しては、ログの種類により出力されるメッセージの性質が異なります。

ログ種別	DEBUG の出力内容
ネットワークログ	ネットワークに関する詳細な利用状況
メモリログ	出力なし
データベースログ	データベースに関する詳細な利用状況 ※ SQL文はDEBUGレベルで出力されます。
リクエストログ	出力なし
セキュリティログ	ログイン認証の詳細な制御情報
画面遷移ログ	出力なし

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

4.3 ネットワークログ

サーバ間のネットワーク通信に関するログです。ネットワークに関する状態が通知されるので、ネットワーク関連の操作に関してトレースするための情報として利用可能です。

4.3.1 パラメータ

識別名	NETWORK_LOG	
設定ファイル	im_logger_network.xml	
出力	ファイル	network.log
	コンソール	
ログレベルの初期値	off	

パターン文字列	説明	初期値
%date	日時	○
%X{log.id}	ログ ID	○
%X{log.report.sequence}	ログ出力順序番号(シーケンス番号)	○
%thread	メッセージ通知したスレッドID	○
%X{log.thread.group}	メッセージ通知したスレッドグループ	---
%msg	メッセージ	○

4.4 メモリログ

サーバのメモリ利用状況に関するログです。出力情報は、サーバが使用しているメモリ量に関するものです。
このログにより得られた情報は、システムのパフォーマンスチューニングに役立てることができます。

4.4.1 パラメータ

識別名	MEMORY_LOG	
設定ファイル	im_logger_memory.xml	
出力	ファイル	memory.log
ログレベルの初期値	off	

パターン文字列	説明	初期値
%date	日時	○
%X{log.report.sequence}	ログ出力順序番号(シーケンス番号)	○
%X{memory.free}	現在確保しているメモリ内の空き領域	○
%X{memory.total}	現在確保しているメモリ総量	○
%X{ memory.initial }	初期ヒープサイズ	○
%X{ memory.max }	最大ヒープサイズ	○

4.4.2 メモリ監視間隔(時間)の設定

※ intra-mart 独自のパラメータ設定は im_logger.xml で行います。

設定ファイル	im_logger.xml
パラメータ名	configuration / intra-mart / memory / log time
単位	ミリ秒
デフォルト値	1000
書式	<configuration> <intra-mart> <memory> *<log time="1000"/>* </memory> <intra-mart> </configuration>

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

4.5 データベースログ

データベースアクセスに関するログです。データベースからのデータ取得時や情報更新時などに出力されます。つまり、データベース連携モジュールの各 API を利用すると、その状況がログに記録されることになります。
このログは、アプリケーションの動作状況やシステムのパフォーマンスに関する調査時に、非常に有効な情報となりますが、多量に出力されてしまうという性質上システムのパフォーマンスに影響を与えてしまうため、運用環境での利用はあまりおすすめできません。

4.5.1 パラメータ

識別名	DB_LOG	
設定ファイル	im_logger_database.xml	
出力	ファイル	database.log
	コンソール	
ログレベルの初期値	off	

パターン文字列	説明	初期値
%date	日時	○
%X{request.id}	リクエスト ID	○
%X{log.id}	ログ ID	○
%X{log.report.sequence}	ログ出力順序番号(シーケンス番号)	○
%thread	メッセージ通知したスレッドID	---
%X{log.thread.group}	メッセージ通知したスレッドグループ	---
%X{db.connection.identifier}	接続ID	○
%X{db.connection.res}	コネクションID	○
%X{db.sql}	SQL	○
%X{db.time}	処理時間	○
%msg	メッセージ	○

4.5.2 パラメータ置換の設定

この設定を有効にすると、preparedstatement の値を置換した SQL 文をログに出力します。
※ intra-mart 独自のパラメータ設定は im_logger.xml で行います。

設定ファイル	im_logger.xml
パラメータ名	configuration / intra-mart / database/log sqlparam
デフォルト値	false
書式	<configuration> <intra-mart> <database> *<log sqlparam="false"/>* </database> <intra-mart> </configuration>

4.5.2.1 注意事項

- 実行した SQL 文を出力する場合には、ログレベルを「DEBUG」レベルに設定する必要があります。
- 置換された値は、データベースのカラムの型に関わらず、全てシングルクォート「'」で囲まれた文字列として置換れます。
  そのため、出力された SQL 文をそのまま実行するとカラムの型の不一致からエラーになる場合があります。
  必要に応じて編集して下さい。

（例）
カラム:item_cd ・・・ VARCHAR 型
カラム:price ・・・ NUMBER 型

＜出力される SQL 文＞
SELECT * FROM item_table WHERE item_cd = '0001' AND ***price = '3000'***;

4.5.3 データベースログの出力判定方式の設定

データベースログのロガー名は、ログ出力要求を行ったクラス名の先頭に識別名「DB_LOG」が設定されます。
（「"DB_LOG." + ログ出力要求を行ったクラス名」形式）

データベースログの出力判定方式は以下の 2 種類が用意されています。

- CALLER_CLASS_NAME
- ROOT_DB_LOGGER

パラメータ設定は im_logger.xml で行います。

設定ファイル	im_logger.xml
パラメータ名	configuration / intra-mart / database/ log isEnabledMode
デフォルト値	CALLER_CLASS_NAME
書式	<configuration> <intra-mart> <database> *<log isEnabledMode ="ROOT_DB_LOGGER"/>* </database> <intra-mart> </configuration>

4.5.3.1 出力判定方式「CALLER_CLASS_NAME」

この方式では、ロガー名を厳密に判定してデータベースログの出力可否を決定します。
これにより、ログ出力要求を行ったクラス単位で、データベースログの出力制御を行うことが可能です。
ただし、ログ出力要求を行ったクラス名を特定する際に、スタックトレースを生成する（=Thread.currentThread().getStackTrace()）ため、出力判定方式「ROOT_DB_LOGGER」よりも処理が遅くなります。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

4.5.3.2 出力判定方式「ROOT_DB_LOGGER」

この方式では、識別名「DB_LOG」だけを判定してデータベースログの出力可否を決定します。
出力判定方式「CALLER_CLASS_NAME」のような細かい出力可否制御は行えませんが、出力判定処理が速くなります。特に、データベースログを出力しない場合に効果を発揮します。

4.5.3.3 出力判定方式の違いによる出力制御の例

```
.
.
.
<logger name="DB_LOG" additivity="false">
    <level value="off" />
    <appender-ref ref="STDOUT" />
    <appender-ref ref="DB_FILE" />
</logger>

<logger name="DB_LOG.aaa.Foo">
    <level value="info" />
</logger>

<logger name="DB_LOG.bbb.Bar">
    <level value="debug" />
</logger>
.
.
.
```

上記のようにログレベルが設定されている場合、出力判定方式ごとの出力可否は以下のようになります。

- 出力判定方式「CALLER_CLASS_NAME」の場合
  - ログ出力要求を行ったクラス単位で、データベースログの出力可否を決定
    - aaa.Foo クラスは、info レベル以上の DB ログが出力可能
    - bbb.Bar クラスは、debug レベル以上の DB ログが出力可能
    - 上記以外のクラスが DB 操作を行っても、DB ログは出力されない

- 出力判定方式「ROOT_DB_LOGGER」の場合
  - 識別名「DB_LOG」だけを判定してデータベースログの出力可否を決定
    - aaa.Foo クラスで DB 操作を行っても、DB ログは出力されない
    - bbb.Bar クラスで DB 操作を行っても、DB ログは出力されない
    - 上記以外のクラスで DB 操作を行っても、DB ログは出力されない

4.6 リクエストログ

リクエストに関するログです。「INFO」レベルのログが出力されるタイミングは、リクエストの処理が終了した際（＝レスポンスを返却する直前）です。なお、リクエストを受け付けた時にログを出力することも可能です。その際は、「DEBUG」レベルに設定して下さい。これにより、処理中の状態、つまり、リクエストを受け付けたが、レスポンスはまだ返却されていない状態であるリクエストを特定することが可能です。「リクエストを受け付けた時のログ」と「レスポンスを返却した時のログ」は、リクエスト ID で紐付けることが可能です。

4.6.1 パラメータ

識別名	REQUEST_LOG	
設定ファイル	im_logger_request.xml	
出力	ファイル	rquest.log
ログレベルの初期値	info	

パターン文字列	説明	初期値
%date	日時	○
%X{request.id}	リクエスト ID	○
%X{log.id}	ログ ID	○
%X{log.report.sequence}	ログ出力順序番号（シーケンス番号）	○
%thread	メッセージ通知したスレッドID	○
%X{log.thread.group}	メッセージ通知したスレッドグループ	---
%X{client.session.id}	セッションID	○
%X{request.remote.host}	接続してきたリモートホスト	○
%X{request.remote.address}	接続してきたリモートアドレス	---
%X{request.url}	リクエストしてきた URL	○
%X{request.url.referer}	リクエストしてきたページの URL	○
%X{request.page.time}	ページの処理時間（ミリ秒）	○
%X{request.accept.time}	リクエストの受付時刻	○
%msg	リクエストを受け付けた時のログは「IN」、 レスポンスを返却した時のログ」は「OUT」が出力されます。	---

4.6.1.1 注意事項

- リクエストを受け付けた時のログを出力する場合には、ログレベルを「DEBUG」レベルに設定する必要があります。
- 初期設定では、パターン文字列「%msg」は設定されていません。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

4.7 セキュリティログ

アクセスセキュリティの運用状況に関するログです。ユーザのログイン認証行為に関する情報を得ることができます。このログとアクセスログを合わせて解析することにより、不正アクセスを見つけ出すための情報を得ることも可能です。

4.7.1 パラメータ

識別名	SECURITY_LOG	
設定ファイル	im_logger_security.xml	
出力	ファイル	security.log
ログレベルの初期値	error	

パターン文字列	説明	初期値
%date	日時	○
%X{request.id}	リクエスト ID	○
%X{log.id}	ログ ID	○
%X{log.report.sequence}	ログ出力順序番号（シーケンス番号）	○
%thread	メッセージ通知したスレッドID	○
%X{log.thread.group}	メッセージ通知したスレッドグループ	---
%X{security.id.session}	セッションID	○
%X{security.id.account}	アカウントID	○
%X{security.id.group}	ログイングループ	○
%X{security.id.logintype}	ログインタイプ （super / group / user）	○
%msg	メッセージ	○

4.7.1.1 注意事項

- ログイン認証情報（ログイン時、ユーザ ID、パスワード不正などによるログイン失敗時）は、ログレベルを「INFO」レベルに設定する必要があります。

4.8 画面遷移ログ

ユーザの画面遷移状況に関するログです。

4.8.1 パラメータ

識別名	TRANSITION_LOG	
設定ファイル	im_logger_transition.xml	
出力	ファイル	transition.log
ログレベルの初期値	info	

パターン文字列	説明	初期値
%date	日時	○
%X{request.id}	リクエスト ID (リクエスト単位で同一の ID が出力)	○
%X{log.id}	ログ ID	○
%X{log.report.sequence}	ログ出力順序番号(シーケンス番号)	○
%thread	メッセージ通知したスレッドID	---
%X{transition.log.type.id}	遷移タイプ REQUEST ・・・通常の遷移 FORWARD ・・・RequestDispatcher#forward()時の遷移 INCLUDE ・・・RequestDispatcher#include()時の遷移	○
%X{request.remote.address}	クライアントの IP アドレス	○
%X{request.remote.host}	クライアントのホスト名	○
%X{transition.access.user.id}	ログインユーザID	○
%X{client.session.id}	セッションID	○
%X{transition.path.page.next}	遷移先画面のパス	○
%X{transition.time.response}	応答時間	○
%X{transition.exception.name}	例外名	○
%X{transition.exception.message}	例外メッセージ	○
%X{transition.path.page.previous}	遷移元画面のパス	○

4.8.1.1 注意事項

遷移元画面、および、遷移先画面のパスは以下のように出力されます。

遷移タイプ	遷移元画面のパス	遷移先画面のパス
REQUEST	HTTP ヘッダ「Referer」のサーブレットパス部分 (HTTP ヘッダ「Referer」が取得できない場合は出力されません)	URL のサーブレットパス部分
FORWARD	RequestDispatcher#forward()を実行する前の遷移先画面パス	ServletRequest#getRequestDispatcher()時に指定したパス
INCLUDE	RequestDispatcher#forward()を実行する前の遷移先画面パス	ServletRequest#getRequestDispatcher()時に指定したパス

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

いづれの遷移タイプでも、遷移元画面、および、遷移先画面がスクリプト開発モデルだった場合には、スクリプト開発モデルの画面のパスが出力されます。

例えば、3 つの JSP「**1_request.jsp**」、「**2_forwarded.jsp**」、「**3_included.jsp**」が存在し、
「**1_request.jsp**」から「**2_forwarded.jsp**」へフォーワードし、
フォーワード先の「**2_forwarded.jsp**」内で「**3_included.jsp**」をインクルードする場合、
遷移元画面、および、遷移先画面のパスは以下のように出力されます。
（なお、この時の画面遷移 ID は、同一の ID が出力されます。）

出力順	遷移タイプ	遷移元画面のパス	遷移先画面のパス
1	REQUEST	-	**1_request.jsp**
2	FORWARD	**1_request.jsp**	**2_forwarded.jsp**
3	INCLUDE	**2_forwarded.jsp**	**3_included.jsp**

応答時間は、遷移タイプによって出力される値が異なります。

遷移タイプ	応答時間
REQUEST	リクエストの処理開始から、レスポンスを返却するまでの時間をあらわします。
FORWARD	リクエストの処理開始から、FORWARD した時点 (=ディスパッチ前の時点)までの時間をあらわします。
INCLUDE	リクエストの処理開始から、INCLUDE した時点(=ディスパッチ前の時点)までの時間をあらわします。

4.8.1.2 制限事項

遷移タイプ「FORWARD」、および、「INCLUDE」のログは、ServletRequest#getRequestDispatcher()で取得した RequestDispatcher を利用してフォーワード/インクルードされた場合に出力されます。
（ServletContext 経由で RequestDispatcher を取得した場合は出力されません）

5 設定ファイルと設定

5.1 設定ファイルの種類

intra-mart 標準のログ機能が利用する設定ファイルを以下に示します。
それぞれの設定ファイルについて十分に理解をした上で、設定を編集して下さい。

ファイル名	用途
conf/log/im_logger.xml	システムログに関する設定及び、各種ログ設定ファイルの読み込みを行うログ機能の基本設定ファイル
conf/log/im_logger_database.xml	データベースログに関する設定ファイル
conf/log/im_logger_memory.xml	メモリログに関する設定ファイル
conf/log/im_logger_network.xml	ネットワークログに関する設定ファイル
conf/log/im_logger_request.xml	リクエストログに関する設定ファイル
conf/log/im_logger_security.xml	セキュリティログに関する設定ファイル
conf/log/im_logger_transition.xml	画面遷移ログに関する設定ファイル

5.2 設定ファイルの編集に関する注意点

設定ファイルを編集した場合、その変更内容をサーバの動作に反映させるには、該当するサーバの再起動が必要です。
すでにサービスの運用を開始している環境に対して設定ファイルを編集する場合は、メンテナンス時間を設けて全てのサーバを停止した後に作業を行うようにしてください。運用中のサーバに対して設定ファイルを編集することや、ネットワーク連携しているサーバのうち1つまたはすべてではない複数のサーバに関してのみ（他のサーバは運用を継続）設定を変更した場合、予期せぬエラーの原因となることがあります。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

5.3 im_logger.xml

5.3.1 概要説明

im_logger.xml はログ出力の基本設定ファイルとなります。
im_logger.xml は以下の役割を担っています。

- ロガー単位での出力制御が行われていないログ（システムログ）の出力設定
- 製品基盤部分（Axis2、Seasar）のログの出力設定
- 製品独自のパラメータの設定（メモリ監視間隔の設定、SQL パラメータ置換の設定）
- 各種ログ設定ファイルの読み込み（特定用途ログ、ユーザアプリケーションで独自に作成したログなどの設定ファイルの読み込み）

```
<configuration>
  <jmxConfigurator />

  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    ～
  </appender>
  <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    ～
  </appender>

  <root>
      <level value="info" />
    ～
  </root>

  <!--
     - Logger for Axis2
    -->
  <logger name="org.apache.axis2.util.PrettyPrinter">
      <level value="error"/>
  </logger>

  <!--
     - Logger for Seasar
    -->
  <logger name="org.seasar">
      <level value="debug" />
  </logger>

  <!--
     - Parameter for intra-mart only
  -->
  <intra-mart>
      <database>
          <log sqlparam="false"/>
      </database>
      ～
  </intra-mart>

  <!-- !!! DO NOT MODIFIED !!! -->
  <!--%INCLUDE FOR ANOTHER LOG CONFIG%-->

</configuration>
```

ログの出力設定
（ロガー単位で、出力制御されておらず、システムログとして出力されるもの）

製品基盤部分のログ出力設定

製品独自のパラメータ設定

各種ログ設定ファイルの読み込み
（置換文字列）

5.3.2 注意事項

- im_logger.xml の移動や削除は行わないでください。

➢ im_logger.xml に、直接ログの出力設定を追加することも可能ですが、設定を追加する際には、<logger>要素より上に利用する<appender>要素を記述してください。

＜正しい設定例＞

正常にログが出力されます。

```
・
・
・
<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <layout class="ch.qos.logback.classic.PatternLayout">
        <Pattern>%date %level %msg</Pattern>
    </layout>
</appender>

<logger name="foo.Bar" additivity="false">
    <level value="DEBUG" />
    <appender-ref ref="STDOUT" />
</logger>
・
・
・
```

使用する Appender が
<logger>要素より上に定義されている

＜誤った設定例＞

ログが出力されません。

```
・
・
・
<logger name="foo.Bar" additivity="false">
    <level value="DEBUG" />
    <appender-ref ref="STDOUT" />
</logger>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <layout class="ch.qos.logback.classic.PatternLayout">
        <Pattern>%date %level %msg</Pattern>
    </layout>
</appender>
・
・
・
```

使用する Appender が
<logger>要素より下に定義されている

➢ 以下、conf/log 配下に配置されたログ設定ファイルを読み込むための置換文字列となります。
編集、削除等は絶対に行わないでください。
置換文字列の編集、削除等行われた場合、ログが正常に出力されなくなります。

```
・
・
・
<!-- !!! DO NOT MODIFIED !!! -->
<!--%INCLUDE_FOR_ANOTHER_LOG_CONFIG%-->
```

5.4 特定用途ログの設定ファイル

5.4.1 概要説明

特定用途ログとされるログ（データベースログ、メモリログ、ネットワークログ、リクエストログ、セキュリティログ、画面遷移ログ）は、以下専用の設定ファイルにて出力設定が行われています。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

ログの種類	ファイル名
データベースログ	im_logger_database.xml
メモリログ	im_logger_memory.xml
ネットワークログ	im_logger_network.xml
リクエストログ	im_logger_request.xml
セキュリティログ	im_logger_security.xml
画面遷移ログ	im_logger_transition.xml

特定用途ログの各設定ファイルは、ログ出力の基本設定ファイルである「im_logger.xml」にその内容が include されます。

以下 im_logger_database.xml を例に、各設定について解説します。

設定1 （im_logger_database.xml 抜粋）

```
<included>            ・・・ ①
```

解説1

① include ファイルであることを定義します。
< included >< /included >で囲われた範囲が、im_logger.xml に include されます。

設定2 （im_logger_database.xml 抜粋）

```
<!--
    - DB_LOG
  -->
<appender name"DB_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">        ・・・ ②
    <file>${im.home}/log/platform/database.log</file>                                   ・・・ ③
```

解説2

② データベースログで使用する Appender 名と Appender クラスを定義しています。
「RollingFileAppender」を定義しているので、データベースログの内容がファイルに出力されます。

③ データベースログファイルの出力先と出力ファイル名を定義しています。
データベースログの内容が「database.log」ファイルに出力されます。

設定3 （im_logger_database.xml 抜粋）

```
    <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
        <FileNamePattern>${im.home}/log/platform/database%i.log</FileNamePattern>     ・・・ ④
        <MinIndex>1</MinIndex>
        <MaxIndex>5</MaxIndex>
    </rollingPolicy>

    <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
        <MaxFileSize>10MB</MaxFileSize>                                                 ・・・ ⑤
    </triggeringPolicy>
```

解説3

④ データベースログファイルのローテイト時のバックアップファイル名称パターンを定義しています。

「database%i.log」と定義しているので、database1.log、database2.log・・・というかたちで、バックアップファイルが生成されます。

⑤ データベースログファイルのローテイト条件を定義しています。
「SizeBasedTriggeringPolicy」を定義しているので、「MaxFileSize」に指定した 10MB でファイルサイズローテイトが行われます。

設定4 （im_logger_database.xml 抜粋）

```
<layout class="jp.co.intra_mart.common.platform.log.layout.OutputStackTracePatternLayout">        ・・・ ⑥
    <pattern>[%d{yyyy-MM-dd HH:mm:ss.SSS}]%X{log.report.sequence}  ・・・ </pattern>              ・・・ ⑦

    <enableOutputStackTrace>true</enableOutputStackTrace>
    <stackTraceDir>${im.home}/log/platform/exception/</stackTraceDir>
    <stackTraceFilename>'exception_'yyyy-MM-dd_HH-mm-ss'_%logId.log'</stackTraceFilename>     ・・・ ⑧
</layout>
```

解説4

⑥ データベースログの Layout クラスを定義しています。
「OutputStackTracePatternLayout」を定義しているので、Exception が通知された場合には、Exception の内容が別ファイルに出力されます。

⑦ データベースログの出力項目と出力フォーマットを定義しています。

⑧ Exception が通知された際に生成される Exception ログのファイル名称パターンを定義しています。
「'exception_'yyyy-MM-dd_HH-mm-ss'_%logId.log'」と定義しているので、Exception が通知された場合には、「exception_2008-06-30_17-41-21_5hpz9b0i3vgsq.log」といった形式のファイルが生成されます。

設定5 （im_logger_database.xml 抜粋）

```
<logger name="DB_LOG" additivity="false">
    <level value="off" />                    ・・・ ⑨
    <appender-ref ref="STDOUT" />            ・・・ ⑩
    <appender-ref ref="DB_FILE" />
</logger>
```

解説5

⑨ データベースログのログレベルを定義しています。
「level value="off"」となっているので、デフォルトではデータベースログは出力されません。
「level value」に指定したログレベルのログが出力されることとなります。

⑩ データベースログが使用する Appender を定義してます。
「STDOUT」 と「DB_FILE」が定義されているので、コンソールとファイルにデータベースログの内容が出力されます。
※ 「STDOUT」については、im_logger.xml に定義されているものを利用しています。

5.5 設定ファイルの新規作成

5.5.1 概要説明

ユーザアプリケーションプログラムにおいて、専用のロガー名でログの出力、管理を行いたい場合、
ユーザアプリケーション専用のログ設定ファイルを作成して、指定のディレクトリに配置することで
独自に管理することが実現が可能となります。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

※ 作成したログ設定ファイルを指定のディレクトリに配置することで、ログの基本設定ファイルである im_logger.xml に読み込まれます。

設定ファイルの配置ディレクトリ	%install_dir%/conf/log

ここでは、設定ファイルの作成方法ついて解説します。

5.5.2 設定ファイルの作成

5.5.2.1 前提条件

ここでは、以下の条件を前提として設定ファイルを作成します。

- ロガー名 ・・・「USER_APPLICATION_LOG」
- ロガー「USER_APPLICATION_LOG」が使用するアペンダー名 ・・・「USER_APPLICATION_FILE」
- 設定ファイルのディレクトリ ・・・「%install_dir%/conf/log」
- 設定ファイル名 ・・・「user_app_logger.xml」
- ログ出力先 ・・・「ファイル（ローテイト有）」
- ログファイルの出力先 ・・・「%install_dir%/log/user_app」
- ログファイル名 ・・・「user_app.log」
- ローテイト条件 ・・・ ファイルサイズ 5MB でローテイト
- バックアップファイルの最小値 ・・・「1」
- バックアップファイルの最大値 ・・・「5」
- Exception の別ファイル出力 ・・・ 有効
- 出力レベル ・・・「INFO」

5.5.2.2 設定ファイルの作成

手順1

im_logger.xml に読み込まれるファイルなので< included >< /included >を記述します。

```
<included>

</included >
```

手順2

Appender を定義して、ログの出力先、出力ファイル名を決定します。

```
<included>
  <appender name="USER_APPLICATION_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
      <file>${im.home}/log/user_app/user_app.log</file>

      <append>true</append>
      <ImmediateFlush>true</ImmediateFlush>
  </appender>
</included >
```

手順3

ファイルのローテイト条件と、ローテイトの際のバックアップファイルの名称パターンを決定します。

```
<included>
  <appender name="USER_APPLICATION_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
      <file>${im.home}/log/user_app/user_app.log</file>

      <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
          <FileNamePattern>${im.home}/log/user_app/user_app%i.log</FileNamePattern>
          <MinIndex>1</MinIndex>
          <MaxIndex>5</MaxIndex>
      </rollingPolicy>

      <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
          <MaxFileSize>5MB</MaxFileSize>
      </triggeringPolicy>

      <append>true</append>
      <ImmediateFlush>true</ImmediateFlush>
  </appender>
</included>
```

手順4

Layout を定義して、出力フォーマットを決定します。

```
<included>
  <appender name="USER_APPLICATION_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
      <file>${im.home}/log/user_app/user_app.log</file>

      <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
          <FileNamePattern>${im.home}/log/user_app/user_app%i.log</FileNamePattern>
          <MinIndex>1</MinIndex>
          <MaxIndex>5</MaxIndex>
      </rollingPolicy>

      <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
          <MaxFileSize>5MB</MaxFileSize>
      </triggeringPolicy>

      <layout class="jp.co.intra_mart.common.platform.log.layout.OutputStackTracePatternLayout">
          <pattern>[%d{yyyy-MM-dd HH:mm:ss.SSS}]    -    %msg%n</pattern>
      </layout>

      <append>true</append>
      <ImmediateFlush>true</ImmediateFlush>
  </appender>
</included>
```

手順5

Exception の別ファイル出力を有効として、Exception ログの出力先とファイル名称パターンを決定します。

```
<included>
  <appender name="USER_APPLICATION_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
      <file>${im.home}/log/user_app/user_app.log</file>

      <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
          <FileNamePattern>${im.home}/log/user_app/user_app%i.log</FileNamePattern>
          <MinIndex>1</MinIndex>
          <MaxIndex>5</MaxIndex>
      </rollingPolicy>

      <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
          <MaxFileSize>5MB</MaxFileSize>
      </triggeringPolicy>
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

```
        <layout class="jp.co.intra_mart.common.platform.log.layout.OutputStackTracePatternLayout">
            <pattern>[%d{yyyy-MM-dd HH:mm:ss.SSS}]      -     %msg%n</pattern>

            <enableOutputStackTrace>true</enableOutputStackTrace>
            <stackTraceDir>${im.home}/log/platform/exception/</stackTraceDir>
            <stackTraceFilename>'exception_'yyyy-MM-dd_HH-mm-ss'_%logId.log'</stackTraceFilename>
        </layout>

        <append>true</append>
        <ImmediateFlush>true</ImmediateFlush>
    </appender>
</included>
```

手順6

ロガー「USER_APPLICATION_LOG」が出力するログのログレベルを決定します。

```
<included>
    <appender name="USER_APPLICATION_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${im.home}/log/user_app/user_app.log</file>

        <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
            <FileNamePattern>${im.home}/log/user_app/user_app%i.log</FileNamePattern>
            <MinIndex>1</MinIndex>
            <MaxIndex>5</MaxIndex>
        </rollingPolicy>

        <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
            <MaxFileSize>5MB</MaxFileSize>
        </triggeringPolicy>

        <layout class="jp.co.intra_mart.common.platform.log.layout.OutputStackTracePatternLayout">
            <pattern>[%d{yyyy-MM-dd HH:mm:ss.SSS}]      -     %msg%n</pattern>

            <enableOutputStackTrace>true</enableOutputStackTrace>
            <stackTraceDir>${im.home}/log/platform/exception/</stackTraceDir>
            <stackTraceFilename>'exception_'yyyy-MM-dd_HH-mm-ss'_%logId.log'</stackTraceFilename>
        </layout>

        <append>true</append>
        <ImmediateFlush>true</ImmediateFlush>
    </appender>

    <logger name="USER_APPLICATION_LOG" additivity="false">
        <level value="info" />
    </logger>
</included>
```

手順7

ロガー「USER_APPLICATION_LOG」が利用する Appender を決定します。

※ コンソールに出力する Appender は、im_logger.xml に定義されている「STDOUT」を利用することにします。

```
<included>
    <appender name="USER_APPLICATION_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${im.home}/log/user_app/user_app.log</file>

        <layout class="jp.co.intra_mart.common.platform.log.layout.OutputStackTracePatternLayout">
            <pattern>[%d{yyyy-MM-dd HH:mm:ss.SSS}]      -     %msg%n</pattern>

            <enableOutputStackTrace>true</enableOutputStackTrace>
            <stackTraceDir>${im.home}/log/platform/exception/</stackTraceDir>
            <stackTraceFilename>'exception_'yyyy-MM-dd_HH-mm-ss'_%logId.log'</stackTraceFilename>
        </layout>

        <append>true</append>
        <ImmediateFlush>true</ImmediateFlush>
    </appender>
```

```
  <logger name="USER_APPLICATION_LOG" additivity="false">
      <level value="info" />
      <appender-ref ref="STDOUT" />
      <appender-ref ref="USER_APPLICATION_FILE" />
  </logger>
</included>
```

手順8

%install_dir%/conf/log 配下に「user_app_logger.xml」というファイル名称で保存して、ユーザアプリケーション専用のログ出力設定ファイルの作成は完了です。

5.5.3 注意事項

- intra-mart ver7.0 標準のログ出力先ディレクトリ以外のディレクトリにログを出力する場合には、サーバ起動前に、予め出力先ディレクトリを作成しておく必要があります。

  （例） ログの出力先設定： <file>${im.home}/log/user_app/user_app.log</file>の場合
  ${im.home}/log/user_app ディレクトリ配下に user_app.log を出力する場合、サーバ起動前に“log/user_app”ディレクトリを作成してください。

  サーバ起動時にログ出力先のディレクトリが作成されていない場合には、起動時に以下のエラーが出力されます。

```
java.io.FileNotFoundException: ${インストールディレクトリ}/log/user_app/user_app.log(指定されたパスが見つかりません。)
        at java.io.FileOutputStream.openAppend(Native Method)
        at java.io.FileOutputStream.<init>(FileOutputStream.java:177)
        at java.io.FileOutputStream.<init>(FileOutputStream.java:102)
                                                    ・
                                                    ・
                                                    ・
!!!! Please check your LOGBACK configuration file !!!!
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

5.5.4 substitutionPropertyの利用

「substitutionProperty」を使用することで、任意のプロパティファイルに定義した変数をログ出力設定ファイル内で利用すること可能となります。
構文は UNIX のシェル変数と同様の記述となります。

（例 1） 設定ファイルに変数を直接定義する場合

im_logger.xml

```
<configuration>
<substitutionProperty name="user.home.dir" value=" C:// log"/>

<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${user.home.dir}/myApp.log</file>
                                                    .
                                                    .
                                                    .
```

（例 2） 任意のプロパティファイルに変数を定義する場合

im_logger.xml

```
<configuration>
<substitutionProperty file="variables1.properties" />

<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${user.home.dir}/myApp.log</file>
                                                    .
                                                    .
                                                    .
```

プロパティファイル（variables1.properties）は、実行ディレクトリ%install%/bin からの相対パス、もしくは絶対パスで指定する必要があります。

variables1.properties

```
user.home.dir= C:// log
```

（例 3）任意のプロパティファイルに変数を複数定義する場合

im_logger.xml

```
<configuration>

<substitutionProperty file="variables2.properties" />
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <file>${destination}</file>
.
.
.
```

プロパティファイル（variables2.properties）は、実行ディレクトリ%install%/bin からの相対パス、もしくは絶対パスで指定する必要があります。

variables2.properties

```
user.home.dir= C:// log
fileName=myApp.log
destination=${user.home.dir}/${fileName}
```

5.5.5 JMX経由による設定

5.5.5.1 概要説明

JMX(Java Management Extensions)はJ2SE 5.0から導入された、JavaアプリケーションやJVMの状態を監視/管理するための仕組みであり、モニタリングツールを用いることで、JMX の機能を使ってロギングの管理を行うことが可能となります。

ここでは、モニタリングツールとして Jconsole を利用してログレベルの取得、変更を行います。

5.5.5.2 設定

手順1

intra-mart WebPlatform 起動時の JVM 引数に以下を指定します。

- ローカル監視

```
% java -Dcom.sun.management.jmxremote
```

- リモート監視

```
% java -Dcom.sun.management.jmxremote.port=ポート番号
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
```

手順2

JConsole を起動します。

コマンドプロンプトから jconsole コマンドを実行します。

```
% jconsole
```

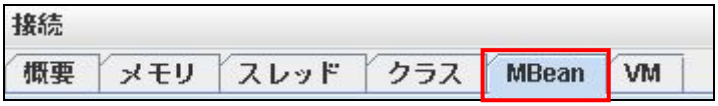
手順3

エージェント接続ウインドで、接続先のプロセスまたはリモートサーバを指定して、接続を行います。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

手順4

画面上部のタブから「Mbean」タブを選択します。



手順5

ツリーから以下 Logback の MBean を選択します。

手順6

右画面のタブから「操作」タブを選択します。

右画面の各項目については、以下に記載します。

メソッド	p1	p2	概要
getLoggerLevel	ロガー名	---	p1 に指定されたロガーに関連付けられているログレベルを返却します。
setLoggerLevel	ロガー名	ログレベル	p1 に指定されたロガーに対して、p2 に指定されたログレベルを設定します。
getLoggerEffectiveLevel	ログレベル	---	p1 に指定されたロガーに関連付けられているログレベルを返却します。

手順7

「setLoggerLevel」メソッドにログレベルを指定することで、intra-mart WebPlatform を再起動することなく、ログレベルを変更することが可能となります。

5.6 設定ファイルのバックアップに関する注意点

intra-mart は、<%im_path%>/conf/log/ディレクトリ直下に存在する全ての xml ファイルを、ログ設定ファイルとして利用します。 従って、<%im_path%>/conf/log/ディレクトリ内で、以下のようにファイルをバックアップすると、ログの設定が正しく行われません。

■ 誤ったバックアップ例

```
<%im_path%>
  |
  ├ conf/
  |  |
  |  └ log/
  |     |
  |     ├ im_logger.xml                  バックアップ元のファイル
  |     |
  |     └ コピー ～ im_logger.xml        バックアップしたファイル
  |
```

正しくログの設定を行うには、バックアップのファイルの拡張子を xml 以外に変更するか、<%im_path%>/conf/log/以外のディレクトリにバックアップする必要があります。以下に正しい設定の例を示します。

■ 正しいバックアップ例1

バックアップするファイルの拡張子を xml 以外 に変更します。

```
<%im_path%>
  |
  ├ conf/
  |  |
  |  └ log/
  |     |
  |     ├ im_logger.xml                  バックアップ元のファイル
  |     |
  |     └ コピー ～ im_logger.xml.bak    拡張子を bak にして、バックアップしたファイル
  |
```

■ 正しいバックアップ例 2

<%im_path%>/conf/log/以外のディレクトリにバックアップします。

```
<%im_path%>
  |
  ├ conf/
  |  |
  |  └ log/
  |     |
  |     └ im_logger.xml                  バックアップ元のファイル
  |        |
  |        └ バックアップ/               バックアップ用のフォルダ
  |           |
  |           └ im_logger.xml            バックアップしたファイル
  |
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

6 ログのディレクトリ構造

ここではログの出力設定ファイル、ならびにログの出力先ディレクトリについて説明します。
記載するディレクトリ構造は全て、製品インストール時の初期状態となります。

6.1 Server Managerログディレクトリ構成

```
<%im_path%>
 │
 ├ bin/
 │  │
 │  └ im_logger_manager.xml        Server Manager ログ設定ファイル
 │
 │
 ├ log/
 │  │
 │  └ manager                      Server Manager ログ格納ディレクトリ
 │     │
 │     ├ fumble.log                 標準エラー出力
 │     │
 │     ├ manager.log                Server Manager ログ
 │     │
 │     └ report.log                 標準出力
 │
```

※ report.log と fumble.log 双方の情報が ServerManager の起動コンソールに出力されます。

6.2 Application Runtimeログディレクトリ構成

```
│
├ bin/
│ │
│ └ im_logger_controller.xml        Server Controller ログ設定ファイル
│
│
├ conf/
│ │
│ └ log/                            ログ関連設定ファイル格納ディレクトリ
│   │
│   ├ im_logger.xml                 システム(汎用)ログ設定ファイル
│   │
│   ├ im_logger_database.xml        データベースログ設定ファイル
│   │
│   ├ im_logger_memory.xml          メモリログ設定ファイル
│   │
│   ├ im_logger_network.xml         ネットワークログ設定ファイル
│   │
│   ├ im_logger_request.xml         リクエストログ設定ファイル
│   │
│   ├ im_logger_security.xml        セキュリティログ設定ファイル
│   │
│   └ im_logger_transition.xml      画面遷移ログ設定ファイル
│
├ log/
│ │
│ └ controller/                     Server Controller ログ格納ディレクトリ
│ │ │
│ │ └ controller.log                Server Controller(Agent)ログ
│ │
│ ├ platform                        ApplicationRuntime ログ格納ディレクトリ
│ │ │
│ │ ├ database.log                  データベースログ
│ │ │
│ │ ├ memory.log                    メモリログ
│ │ │
│ │ ├ network.log                   ネットワークログ
│ │ │
│ │ ├ request.log                   リクエストログ
│ │ │
│ │ ├ security.log                  セキュリティログ
│ │ │
│ │ ├ system.log                    システム(汎用)ログ
│ │ │
│ │ └ transition.log                画面遷移ログ
│ │
│ ├ access.log                      アクセスログ(Resin HTTP サーバ用)
│ │
│ ├ stderr.log                      標準エラー出力(Resin HTTP サーバ用)
│ │
│ └ stdout.log                      標準出力(Resin HTTP サーバ用)
│
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

6.3 Shared Memory Serviceログディレクトリ構成

```
│
├ bin/
│  │
│  └ im_logger_controller.xml        Server Controller ログ設定ファイル
│
│
├ conf/
│  │
│  └ log/                            ログ関連設定ファイル格納ディレクトリ
│     │
│     ├ im_logger.xml                システム（汎用）ログ設定ファイル
│     │
│     ├ im_logger_memory.xml         メモリログ設定ファイル
│     │
│     └ im_logger_network.xml        ネットワークログ設定ファイル
│
├ log/
│  │
│  └ controller/                     Server Controller  ログ格納ディレクトリ
│  │  │
│  │  └ controller.log               Server Controller（Agent）ログ
│  │
│  ├ platform                        SharedMemoryService ログ格納ディレクトリ
│  │  │
│  │  ├ memory.log                   メモリログ
│  │  │
│  │  ├ network.log                  ネットワークログ
│  │  │
│  │  └ system.log                   システム（汎用）ログ
│  │
│  ├ stderr.log                      標準エラー出力
│  │
│  └ stdout.log                      標準出力
│
```

6.4 Permanent Data Serviceログディレクトリ構成

```
│
├ bin/
│  │
│  └ im_logger_controller.xml        Server Controller ログ設定ファイル
│
│
├ conf/
│  │
│  └ log/                            ログ関連設定ファイル格納ディレクトリ
│     │
│     ├ im_logger.xml                システム(汎用)ログ設定ファイル
│     │
│     ├ im_logger_memory.xml         メモリログ設定ファイル
│     │
│     └ im_logger_network.xml        ネットワークログ設定ファイル
│
├ log/
│  │
│  └ controller/                     Server Controller ログ格納ディレクトリ
│  │  │
│  │  └ controller.log               Server Controller(Agent)ログ
│  │
│  ├ platform                        PermanentDataService ログ格納ディレクトリ
│  │  │
│  │  ├ memory.log                   メモリログ
│  │  │
│  │  ├ network.log                  ネットワークログ
│  │  │
│  │  └ system.log                   システム(汎用)ログ
│  │
│  ├ stderr.log                      標準エラー出力
│  │
│  └ stdout.log                      標準出力
│
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

6.5 Storage Serviceログディレクトリ構成

```
│
├ bin/
│  │
│  └ im_logger_controller.xml        Server Controller ログ設定ファイル
│
│
├ conf/
│  │
│  └ log/                            ログ関連設定ファイル格納ディレクトリ
│     │
│     ├ im_logger.xml                システム（汎用）ログ設定ファイル
│     │
│     ├ im_logger_memory.xml         メモリログ設定ファイル
│     │
│     └ im_logger_network.xml        ネットワークログ設定ファイル
│
├ log/
│  │
│  └ controller/                     Server Controller ログ格納ディレクトリ
│  │  │
│  │  └ controller.log               Server Controller（Agent）ログ
│  │
│  ├ platform                        StorageService ログ格納ディレクトリ
│  │  │
│  │  ├ memory.log                   メモリログ
│  │  │
│  │  ├ network.log                  ネットワークログ
│  │  │
│  │  └ system.log                   システム（汎用）ログ
│  │
│  ├ stderr.log                      標準エラー出力
│  │
│  └ stdout.log                      標準出力
│
```

6.6 Resource Serviceログディレクトリ構成

```
│
├ bin/
│  │
│  └ im_logger_controller.xml        Server Controller ログ設定ファイル
│
│
├ conf/
│  │
│  └ log/                            ログ関連設定ファイル格納ディレクトリ
│     │
│     ├ im_logger.xml                システム(汎用)ログ設定ファイル
│     │
│     ├ im_logger_memory.xml         メモリログ設定ファイル
│     │
│     └ im_logger_network.xml        ネットワークログ設定ファイル
│
├ log/
│  │
│  └ controller/                     Server Controller ログ格納ディレクトリ
│  │  │
│  │  └ controller.log               Server Controller(Agent)ログ
│  │
│  ├ platform                        ResourceService ログ格納ディレクトリ
│  │  │
│  │  ├ memory.log                   メモリログ
│  │  │
│  │  ├ network.log                  ネットワークログ
│  │  │
│  │  └ system.log                   システム(汎用)ログ
│  │
│  ├ stderr.log                      標準エラー出力
│  │
│  └ stdout.log                      標準出力
│
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

6.7 Serialization Serviceログディレクトリ構成

```
│
├ bin/
│  │
│  └ im_logger_controller.xml        Server Controller ログ設定ファイル
│
│
├ conf/
│  │
│  └ log/                            ログ関連設定ファイル格納ディレクトリ
│     │
│     ├ im_logger.xml                システム（汎用）ログ設定ファイル
│     │
│     ├ im_logger_memory.xml         メモリログ設定ファイル
│     │
│     └ im_logger_network.xml        ネットワークログ設定ファイル
│
├ log/
│  │
│  └ controller/                     Server Controller ログ格納ディレクトリ
│  │  │
│  │  └ controller.log               Server Controller（Agent）ログ
│  │
│  ├ platform                        SerializationService ログ格納ディレクトリ
│  │  │
│  │  ├ memory.log                   メモリログ
│  │  │
│  │  ├ network.log                  ネットワークログ
│  │  │
│  │  └ system.log                   システム（汎用）ログ
│  │
│  ├ stderr.log                      標準エラー出力
│  │
│  └ stdout.log                      標準出力
│
```

6.8 Schedule Serviceログディレクトリ構成

```
│
├ bin/
│  │
│  └ im_logger_controller.xml        Server Controller ログ設定ファイル
│
│
├ conf/
│  │
│  └ log/                            ログ関連設定ファイル格納ディレクトリ
│     │
│     ├ im_logger.xml                システム(汎用)ログ設定ファイル
│     │
│     ├ im_logger_memory.xml         メモリログ設定ファイル
│     │
│     └ im_logger_network.xml        ネットワークログ設定ファイル
│
├ log/
│  │
│  └ controller/                     Server Controller ログ格納ディレクトリ
│  │  │
│  │  └ controller.log               Server Controller(Agent)ログ
│  │
│  ├ platform                        ScheduleService ログ格納ディレクトリ
│  │  │
│  │  ├ memory.log                   メモリログ
│  │  │
│  │  ├ network.log                  ネットワークログ
│  │  │
│  │  └ system.log                   システム(汎用)ログ
│  │
│  ├ stderr.log                      標準エラー出力
│  │
│  └ stdout.log                      標準出力
│
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

7 サンプル

7.1 JavaEE開発モデル

7.1.1 作成するもの

- ログ出力プログラム(ログの出力処理を実装するプログラム)
- ログ出力設定ファイル(.xml)

7.1.2 使用するAPI

API	jp.co.intra_mart.common.platform.log.Logger クラス

※ Logger クラスならびにメソッドの詳細については、API リストを参照してください。

7.1.3 前提条件

ここに掲載するサンプルは以下前提条件で作成されています。

クラス	jp.co.sample.application. OutputLog.java	
ロガー名	SAMPLE_LOG	
ログレベル	info	
設定ファイル	*%install_dir%*/conf/log/im_logger_sample.xml	
ログの出力先	コンソール	
	ファイル	*%install_dir%*/log/platform/sample.log

7.1.4 ソースコード

OutputLog.java

```
package jp.co.sample.application;

import jp.co.intra_mart.common.platform.log.Logger;
import java.util.Date;
import java.util.Iterator;

public class OutputLog {

    // ロガーの生成
    static final Logger sampleLogger = Logger.getLogger("SAMPLE_LOG");

    public void doIt() {

        // パラメータの設定
        String parm1= "第 1 パラメータ";
        Integer parm2 = new Integer(10);
        Date parm3 = new Date();
        boolean parm4 = false;
        Object [] args = { parm1, parm2, parm3, parm4 };

        // ログの出力
        sampleLogger.info("パラメータは「{}」と「{}」と「{}」と「{}」です。", args);

    }
}
```

7.1.5 設定ファイル

im_logger_sample.xml

```
<included>

  <appender name="SAMPLE_LOG_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
      <file>${im.home}/log/platform/sample.log</file>

      <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
          <FileNamePattern>${im.home}/log/platform/sample%i.log</FileNamePattern>
          <MinIndex>1</MinIndex>
          <MaxIndex>5</MaxIndex>
      </rollingPolicy>

      <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
          <MaxFileSize>10MB</MaxFileSize>
      </triggeringPolicy>

      <layout class="jp.co.intra_mart.common.platform.log.layout.OutputStackTracePatternLayout">
          <pattern>[%d{yyyy-MM-dd HH:mm:ss.SSS}]%-5level  %logger{255}   %X{log.id} -    %msg%n</pattern>

          <enableOutputStackTrace>true</enableOutputStackTrace>
          <stackTraceDir>${im.home}/log/platform/exception/</stackTraceDir>
          <stackTraceFilename>'exception_'yyyy-MM-dd_HH-mm-ss'_%logId.log'</stackTraceFilename>
      </layout>

      <append>true</append>
      <ImmediateFlush>true</ImmediateFlush>
  </appender>

  <logger name="SAMPLE_LOG" additivity="false">
      <level value="info" />
      <appender-ref ref="SAMPLE_LOG_FILE" />
      <appender-ref ref="STDOUT" />
  </logger>

</included>
```

7.1.6 出力結果

コンソール

```
[INFO] SAMPLE_LOG - パラメータは「第 1 パラメータ」と「10」と「Thu Jun 19 18:25:08 JST 2008」と「false」です。
```

sample.log

```
[2008-06-19 18:25:09.250] INFO SAMPLE_LOG   5hpzh3n1jnm90 - パラメータは「第 1 パラメータ」と「10」と「Thu Jun 19 18:25:09 JST 2008」と「false」です。
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

7.2 スクリプト開発モデル

7.2.1 作成するもの

- ログ出力プログラム(ログの出力処理を実装するプログラム)
- ログ出力設定ファイル(.xml)

7.2.2 使用するAPI

API	Logger オブジェクト

※ Logger オブジェクトならびにメソッドの詳細については、API リストを参照してください。

7.2.3 前提条件

ここに掲載するサンプルは以下前提条件で作成されています。

ファイルパス	*%install_dir%*/pages/sample/log_sample/log_sample.js	
ロガー名	SAMPLE_LOG	
ログレベル	INFO	
設定ファイル	*%install_dir%*/conf/log/im_logger_sample.xml	
ログの出力先	コンソール	
	ファイル	*%install_dir%*/log/platform/sample.log

7.2.4 ソースコード

log_sample.js

```
function outputLog( request ){

        // ロガーの生成
        var sampleLogger = Logger.getLogger("SAMPLE_LOG");

        // パラメータの設定
        var argArray = new Array();
        argArray[0] = "第 1 パラメータ";     // String 型
        argArray[1] = 10;                    // Number 型
        argArray[2] = new Date();            // Date 型
        argArray[3] = false;                 // Boolean 型

        // ログの出力
        sampleLogger.info("パラメータは「{}」と「{}」と「{}」と「{}」です。", argArray);

}
```

7.2.5 設定ファイル

im_logger_sample.xml

```
<included>

  <appender name="SAMPLE_LOG_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
      <file>${im.home}/log/platform/sample.log</file>

      <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
          <FileNamePattern>${im.home}/log/platform/sample%i.log</FileNamePattern>
          <MinIndex>1</MinIndex>
          <MaxIndex>5</MaxIndex>
      </rollingPolicy>

      <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
```

```
            <MaxFileSize>10MB</MaxFileSize>
        </triggeringPolicy>

        <layout class="jp.co.intra_mart.common.platform.log.layout.OutputStackTracePatternLayout">
            <pattern>[%d{yyyy-MM-dd HH:mm:ss.SSS}]%-5level  %logger{255}   %X{log.id} -    %msg%n</pattern>

            <enableOutputStackTrace>true</enableOutputStackTrace>
            <stackTraceDir>${im.home}/log/platform/exception/</stackTraceDir>
            <stackTraceFilename>'exception_'yyyy-MM-dd_HH-mm-ss'_%logId.log'</stackTraceFilename>
        </layout>

        <append>true</append>
        <ImmediateFlush>true</ImmediateFlush>
    </appender>

    <logger name="SAMPLE_LOG" additivity="false">
        <level value="info" />
        <appender-ref ref="SAMPLE_LOG_FILE" />
        <appender-ref ref="STDOUT" />
    </logger>

</included>
```

7.2.6 出力結果

コンソール

```
[INFO] SAMPLE_LOG - パラメータは「第 1 パラメータ」と「10」と「Thu Jun 19 2008 19:20:46 GMT+0900 (JST)」と「false」です。
```

sample.log

```
[2008-06-19 19:20:46.578] INFO SAMPLE_LOG 5hpzh3spypeow - パラメータは「第 1 パラメータ」と「10」と「Thu Jun 19 2008 19:20:46 GMT+0900 (JST)」と「false」です。
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

8 ログの解析

サーバが出力したログは収集・解析をすることにより、そのサーバの運用状態をしるための手段になります。

8.1 httpログ

この機能は、intra-mart WebPlatform(Resin) 専用となります。intra-mart AppFramework の場合は、ご利用の WebApplication Server の仕様をご確認ください。
http ログは、http サーバ部の実行状態を記録するためのもので、Standalone 型で運用した場合と WSC を利用した場合で出力される内容が異なります。
このログは、そのサーバ(AppRSrv)に対するブラウザからの Web アクセスの状態を調査するために利用して下さい。

8.2 サーバ実行ログ

http ログ以外のログは、サーバ実行およびプログラム実行に関わる記録です。
このタイプのログは、メッセージ性質により複数種類のログが存在します。ログファイルはログの種類と同じ数だけ作成されます。
サーバ運用状態を知りたい場合は、必要なログファイルの内容を1つのファイルにまとめることにより様々な情報を得ることができます。(Microsoft Excel(Microsoft Excel は Microsoft 社の製品です)などを利用すると、ログファイルをまとめたり内容をソートしたりといった解析作業を簡単に行うことができます)

状態変化の流れを調査	出力時間とシーケンス番号でソート ログメッセージを出力順に並べ替えることができます。 (シーケンス番号でのソートは特定用途ログのみになります。)
トランザクション単位で調査	メッセージ通知したスレッドIDと出力時間、シーケンス番号でソート ログメッセージを処理トランザクションごとにまとめる事ができます。 (シーケンス番号でのソートは特定用途ログのみになります。)
ユーザの操作状況を調査	リクエスト ID と出力時間でソート セキュリティログ、リクエストログ、画面遷移ログ、データベースログをまとめる事で、操作ログとして利用することが可能となります。

8.3 ログ解析手順の例

複数のログファイルの内容を持つファイル(あるログファイルのコピーに、別のログファイルの内容を追記したもの)を作成します。これを Microsoft Excel で開きます(タブ区切りの CSV 形式ファイルとして読み込みます)。
日時とシーケンス番号の項目でソートすることにより、ログメッセージが処理順に並びますので、実行状況などを調査することができます。
(シーケンス番号でのソートは特定用途ログのみになります。)

(例)
Microsoft Excel にて、
① データ ＞ 並べ替えを選択
② 「並べ替え」ダイアログで、以下を指定
最優先されるキー: 「日時」項目
2 番目に優先されるキー: 「シーケンス番号」項目

8.4 注意事項

- メッセージを通知したスレッド ID:「%thread」については、ログによってデフォルトでは出力が無効となっています。必要に応じてレイアウトに「%thread」を追加してください。
- 特定用途ログのシーケンス番号はログ毎に採番されています。
- ログの出力フォーマット(レイアウト)は、目的に合わせてご利用の環境で適宜設定してください。

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

9 Ver6.1 ログ機能との相違点

intra-mart ver6.1（以下 IMv6.1）では、製品独自のログ機構によりログを出力していましたが、intra-mart ver7.0（以下 IMv7.0）より、「Logback（http://logback.qos.ch/）」と「SLF4J（http://www.slf4j.org/）」を利用してログの出力機構を実現しています。
ここでは、IMv6.1 と IMv7.0 のログ機能の相違点をそれぞれ説明してきます。

9.1 ログ種別の変更

9.1.1 システムログへ統一

IMv6.1 の以下のログを「システムログ」に統一しました。
ログは全て同一のファイルに出力されます。

- システムログ
- エラーログ
- バッチログ
- アプリケーション共通マスタバッチログ

9.1.2 特定の用途を持つログを特定用途ログに変更

システムログと区別するために、以下のログを特定用途ログとしました。
特定用途ログは、各ログを分けるために、ログごとの識別名を用意し、ロガー名の先頭にその識別名が指定されます。

- ネットワーク（識別名：NETWORK_LOG）
- メモリログ（識別名：MEMORY _LOG）
- リクエストログ（識別名：REQUEST _LOG）
- データベースログ（識別名：DB _LOG）
- セキュリティログ（識別名：SECURITY _LOG）
- 画面遷移ログ（識別名：TRANSITION _LOG）

9.2 ログ出力に関する設定ファイルの変更

IMv6.1 では、ログに関する設定は全て imart.xml で行っていましたが、IMv7.0 では、ログ出力に関する設定は、全て専用のファイルで管理するようになりました。

9.3 ログレベルの変更

以下ログレベルの指定が可能になりました。

- ERROR
- WARN
- INFO
- DEBUG
- TRACE

上記 Logback が持つログレベルに対応するために、以下のように既存ログ及び API とのログレベルのマッピングを行いました。

＜imart.xml＞

旧ログ API		新ログ API
通常	通常メッセージ （enable="true"）	INFO レベルで出力
	詳細メッセージ （detail="true"）	DEBUG レベルで出力
エラー	メッセージ （enable="true"）	ERROR レベルで出力
	詳細メッセージ （detail="true"）	ERROR レベルで出力

＜API＞

旧ログ API		新ログ API
システムログ (SystemLogReporter)	通常メッセージ	INFO レベルで出力
	詳細メッセージ	DEBUG レベルで出力
エラーログ (SystemErrorLogReporter)	通常メッセージ	ERROR レベルで出力
	詳細メッセージ	ERROR レベルで出力

※ *上記 API は非推奨となりました。*

9.4 ロガー単位のログ出力

ロガー生成時にロガー名を指定することで、同一ロガー名のロガー単位でのログ出力が可能となりました。

（例）

・log_test.Test.java ユーザアプリケーションプログラム

```
Logger  logger = Logger.getLogger("USER_APPLICATION_LOG");
Logger.info("XXX データの取得に成功しました。");
```

・im_logger.xml

```
<appender name="FILE" class="ch.qos.logback.core.FileAppender">
  <layout class="ch.qos.logback.classic.PatternLayout">
    <Pattern>%-5level %logger{35} - %msg%n</Pattern>
  </layout>
  <File>foo.log</File>
  <Append>true</Append>
</appender>

<logger name=" USER_APPLICATION_LOG " >
<level value="info" />
<appender-ref ref="SECURITY_FILE" />
</logger>
```

・user_app.log

```
[INFO]  log_test.Test.java XXX データの取得に成功しました。
```


Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

9.5 ログの出力先変更

ロガー名単位で、ログの出力先を変更することが可能となりました。
出力先は以下の中から選択することが可能となります。

- コンソール
- ファイル
- メール

9.6 例外（Exception）ログの出力

IMv6.1 までは、エラーログのみ、例外（Exception）が別ファイルに出力されていましたが、
IMv7.0 では、ログ出力 API の引数に Throwable を指定することで、全てのログで例外（Exception）の別ファイル出力を行うことが可能となりました。

※ 例外（Exception）の別ファイル出力を行うには、専用の Layout を利用指定する必要があります。

9.7 操作ログの廃止

IMv6.1 で提供されていた、操作ログが廃止になりました。
IMv7.0 では、特定用途ログとして扱われるセキュリティログ、リクエストログ、画面遷移ログ、データベースログには、リクエスト単位で一意となるリクエスト ID が出力可能であるため、リクエスト ID と日時をキーにセキュリティログ、リクエストログ、画面遷移ログ、データベースログを Excel（Microsoft Excel は Microsoft 社の製品です）等を用いて、マージすることで、操作ログとして利用することが可能となります。

※ 操作ログを DB に出力することはできません。

9.8 ログパラメータ名称の変更

IMv6.1 と IMv7.0 では、ログパラメータ名称が変更されています。
以下に IMv6.1 と IMv7.0 のパラメータ名称の比較表を記載します。

9.8.1 ネットワークログの比較

出力項目	Ver6.1	Ver7.0
日時	{DATE}	%date
ログ ID	---	%X{log.id} *new*
ログ出力順序番号	{SEQUENCE}	%X{log.report.sequence}
メッセージ通知したスレッドID	{THREADID}	%thread
メッセージ通知した スレッドグループ	{THREADGROUP}	%X{log.thread.group}
メッセージ	{MESSAGE}	%msg

9.8.2 メモリログの比較

出力項目	Ver6.1	Ver7.0
日時	{DATE}	%date
ログ出力順序番号	{SEQUENCE}	%X{log.report.sequence}
現在確保しているメモリ内の空き領域	{FREE}	%X{memory.free}
現在確保しているメモリ総量	{TOTAL}	%X{memory.total}
初期ヒープサイズ	{XMS}	%X{ memory.initial }
最大ヒープサイズ	{XMX}	%X{ memory.max }

9.8.3 データベースログの比較

出力項目	Ver6.1	Ver7.0
日時	{DATE}	%date
リクエスト ID	---	%X{request.id} *new*
ログ ID	---	%X{log.id} *new*
ログ出力順序番号	{SEQUENCE}	%X{log.report.sequence}
メッセージ通知したスレッドID	{THREADID}	%thread
メッセージ通知したスレッドグループ	{THREADGROUP}	%X{log.thread.group}
接続ID	{ID}	%X{db.connection.identifier}
コネクションID	{CONNECION}	%X{db.connection.res}
SQL	{SQL}	%X{db.sql}

9.8.4 リクエストログの比較

出力項目	Ver6.1	Ver7.0
日時	{DATE}	%date
リクエスト ID	---	%X{request.id} *new*
ログ ID	---	%X{log.id} *new*
ログ出力順序番号	{SEQUENCE}	%X{log.report.sequence}
メッセージ通知したスレッドID	{THREADID}	%thread
メッセージ通知したスレッドグループ	{THREADGROUP}	%X{log.thread.group}
セッションID	{SESSION}	%X{client.session.id}
接続してきたリモートホスト	{HOST}	%X{request.remote.host}
接続してきたリモートアドレス	{ADDRESS}	%X{request.remote.address}
リクエストしてきた URL	{URL}	%X{request.url}
リクエストしてきたページの URL	{REFERRER}	%X{request.url.referer}
ページの処理時間(ミリ秒)	{TIME}	%X{request.page.time}
リクエストの受付時刻	{ACCEPT_TIME}	%X{request.accept.time}

9.8.5 セキュリティログの比較

出力項目	Ver6.1	Ver7.0
日時	{DATE}	%date
リクエスト ID	---	%X{request.id} *new*
ログ ID	---	%X{log.id} *new*
ログ出力順序番号	{SEQUENCE}	%X{log.report.sequence}
メッセージ通知したスレッドID	{THREADID}	%thread
メッセージ通知したスレッドグループ	{THREADGROUP}	%X{log.thread.group}
セッションID	{SESSION}	%X{security.id.session}
アカウントID	{ACCOUNT}	%X{security.id.account}
ログイングループ	{GROUP}	%X{security.id.group}
ログインタイプ	{LOGINTYPE}	%X{security.id.logintype}
メッセージ	{MESSAGE}	%msg

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

9.8.6 画面遷移ログの比較

出力項目	Ver6.1	Ver7.0
日時	{DATE}	%date
リクエスト ID	---	%X{request.id} *new*
画面遷移 ID	{TRANSITIONID}	---
ログ ID	---	%X{log.id} *new*
ログ出力順序番号	{SEQUENCE}	%X{log.report.sequence}
遷移タイプ	{TYPE}	%X{transition.log.type.id}
クライアントの IP アドレス	{REMOTE_ADDRESS}	%X{request.remote.address}
クライアントのホスト名	{REMOTE_HOST}	%X{request.remote.host}
ログインユーザID	{USER_ID}	%X{transition.access.user.id}
セッションID	{SESSION_ID}	%X{client.session.id}
遷移先画面のパス	{NEXT_PAGE}	%X{transition.path.page.next}
応答時間	{RESPONSE_TIME}	%X{transition.time.response}
例外名	{EXCEPTION_NAME}	%X{transition.exception.name}
例外メッセージ	{EXCEPTION_MSG}	%X{transition.exception.message}
遷移元画面のパス	{PREVIOUS_PAGE}	%X{transition.path.page.previous}

10 サポート

弊社では、Web にて弊社製品に対するサポートおよび技術情報の公開を行っております。当製品に関して不明な点などがございましたら、下記 URL にてホームページにアクセスしていただき、情報検索または弊社サポート窓口までご相談下さい。

intra-mart Developer Support Site アドレス

http://www.intra-mart.jp/support/intramart.cgi

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

11 索引

A

Append 5
Appender 5

B

BufferedIO 6
BufferSize 6, 13

C

Charset 13
commons-logging ..... i, 20
ConsoleAppender 5
ContentTypeCharsetSMTPAppender 12

D

database.log 25
DB_LOG 25
DBAppender 15, 20
DEBUG 3

E

enableOutputStackTrace 18
Encoding 5, 6
ERROR 3
Evaluator ..... i, 13, 14
ExtendedTimeBasedRollingPolicy ..... i, 7, 8

F

File 6
FileAppender 5
FileNamePattern 7, 10
FixedWindowRollingPolicy 10
From 13

G

getLoggerEffectiveLevel 43
getLoggerLevel 43

I

im_logger.xml 33
im_logger_database.xml 25, 32
im_logger_memory.xml 24, 32
im_logger_network.xml 23, 32
im_logger_request.xml 28, 32
im_logger_security.xml 29, 32
im_logger_transition.xml 30, 32
ImmediateFlush 5, 6
INFO 3

J

JAF 12
Janino 12
JavaMail 12
Jconsole 42
JMX 42

L

Layout 16
log sqlparam 25, 26
log time 24
log4j ..... i, 20
Logback 2
Logger 3

M

MaxFileSize 10
MaxIndex 10
MDC 17
memory.log 24
MEMORY_LOG 24
MinIndex 10

N

network.log 23
NETWORK_LOG 23

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

O

OutputStackTracePatternLayout 18

P

PatternLayout 16
POP before SMTP 12
preparedstatement 25

R

REQUEST_LOG 28
RollingFileAppender 6
RollingPolicy 6
rquest.log 28

S

security.log 29
SECURITY_LOG 29
setLoggerLevel 43
SizeBasedTriggeringPolicy 10
SLF4J 2, 59
SMTP-AUTH 12
SMTPHost 13
stackTraceDir 18
stackTraceFilename 18
Subject 13
substitutionProperty 41
system.log 21
SystemErrorLogReporter 60
SystemLogReporter 60

T

Target 5
TimeBasedRollingPolicy 7
To 13
TRACE 3
transition.log 30
TRANSITION_LOG 30

W

WARN 3

あ

圧縮形式 8, 11

か

画面遷移ログ 30

し

詳細ログ 22

せ

セキュリティログ 29

そ

操作ログ 61

て

データベースログ 25

ね

ネットワークログ 23

め

メモリログ 24

り

リクエストログ 28

ろ

ロガー名 3
ログレベル 3, 60

Copyright 2000-2011 株式会社 NTT データ イントラマート All rights Reserved.

intra-mart WebPlatform/AppFramework Ver.7.0
ログ 設定ガイド

2011/06/30 第8版

Copyright 2000-2011 株式会社NTTデータ イントラマート
All rights Reserved.
TEL: 03-5549-2821
FAX: 03-5549-2816
E-MAIL: info@intra-mart.jp
URL: http://www.intra-mart.jp/