Fiorano

Fiorano SOA プラットフォーム

コンポーネント フロー実行ログ

の使用、設定方法

対象バージョン ：2007 SP7 および 9.0.0

はじめに

このガイドブックは、コンポーネント フローの実行中に記録されるログについて、その設定方法および表示方法を例題を用いて説明するものです。

例題は、ガイドブック『**Fiorano オーケストレータによるビジネス プロセス構築の概要 (基礎編)**』で構築方法を説明したデータベース間の同期を取るコンポーネント フロー (DB_Sync) を用いています。DB_Sync コンポーネント フローを作成し、本ガイドブックの説明に従って実際に操作されることをお薦めします。

このガイド ブックは、以下のガイドブックで説明されている知識を有していることを前提としています。

- Fiorano SOA プラットフォームの起動方法
- Fiorano SOA プラットフォームのアーキテクチャ概要
- Fiorano オーケストレータによるビジネス プロセス構築の概要 (基礎編)

目次

1 ログの設定

Fiorano SAO プラットフォームでは、ログを次のカテゴリに分けて記録しています。

- システム イベント ログ
- アクセス ログ (セキュリティ イベント)
- コンポーネント フロー実行ログ

本ガイド ブックは、コンポーネント フローの実行ログについて、その設定方法と表示方法について説明します。
システム イベント ログおよびアクセス ログについては、製品マニュアルを参照してください。

1.1 ログのレベル

Fiorano SOA プラットフォームでは、コンポーネント プロセス内の個々のコンポーネント単位でログのレベルが設定できます。
ログ レベルとはログを記録するレベル (詳細度) を指し、下の表に示すログ レベルが用意されています。

ログ レベルが上がるほど詳細な情報がログされることになります。上位のレベルは、下位のレベルを包含しています。つまり、レベル 3 の Info では、レベル 1 ～ 2 のエラー、ワーニングのログも記録されることになります。

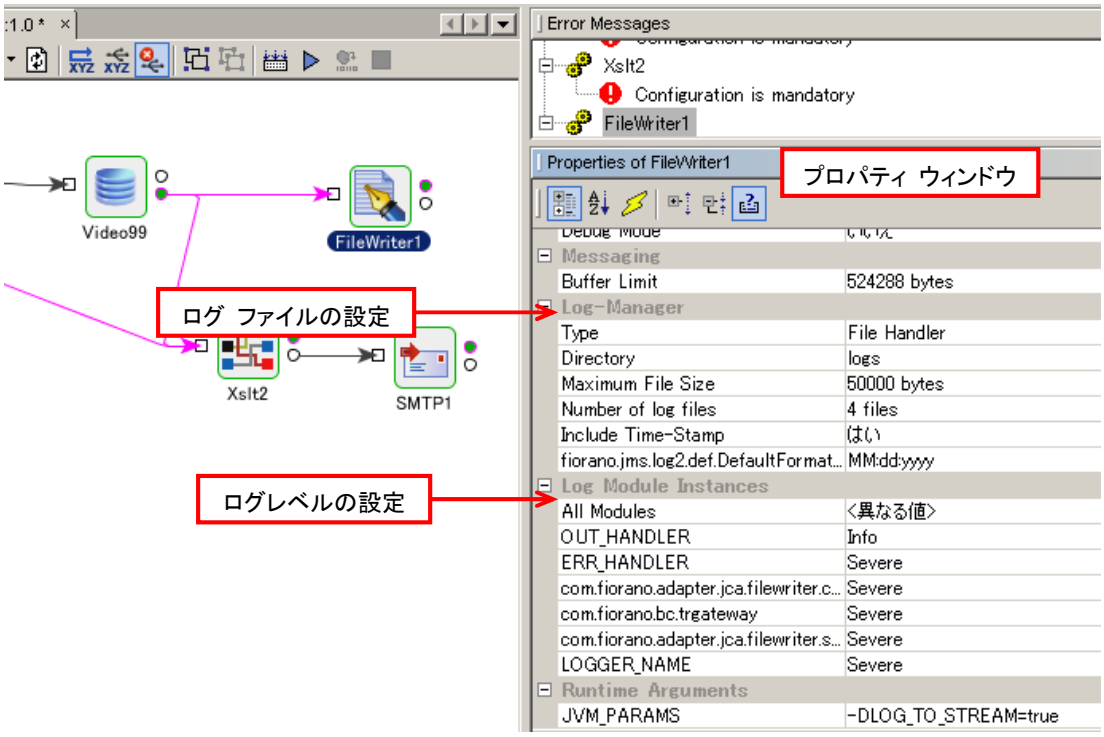
	レベル	説明
0	**Off**	まったくログしない
1	**Severe**	フェイタル エラーなど重大なエラーの発生時のみログする
2	**Warning**	エラー、ワーニングをログする
3	**Info**	エラー、ワーニングに加え、重要なシステム メッセージをログする
4	**Config**	上記のものに加え、システム構成に関するエラー (例えば、JDBC ドライバーにアクセスできないなど) が発生した場合にログする このレベルでログされるエラーは、コンポーネントのプロパティ設定と実環境の相違に起因するものがほとんどである
5	**Fine**	コンポーネントの動作過程をログする (例えば、「ライセンスを確認した」、「スケジューラを起動した」など)
6	**Finer**	さらに詳細な動作過程をログする
7	**Finest**	コンポーネントの動作過程を細大もらさずログする
8	**All**	すべてのエラー、ワーニング、システム メッセージ、イベントの発生をログする

1.2 コンポーネントのログ設定

ログ の設定は、コンポーネント毎に、コンポーネントのプロパティ ウィンドウで行います。

ログの設定は、次の 2 つのカテゴリに分かれています。

- ログ ファイルに関する設定 (Log Manager のパラメータ)
- ログ レベルの設定



1.3 ログ ファイル

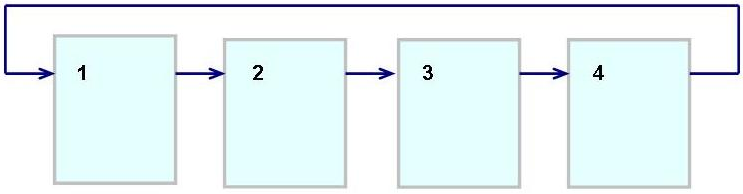
1.3.1 設定項目

ファイル サイズ、ファイル数、日付形式以外の項目は、デフォルト設定を変更しないことをお薦めします。

プロパティ項目名	デフォルト値	説明
Type	File Handler （ログをファイルに書き出します）	次のハンドラーも選択できます **Console Handler** (ESB サーバーの実行マシンにウィンドウ表示されます) **Custom Handler** (ユーザー独自のハンドラーを登録できます)
Directory	logs	ログ ファイルのディレクトリ名 (詳細は下記のログ ファイルの場所を参照してください)
Maximum File Size	50000 Bytes	ログ ファイルの最大サイズ

Number of log files	4	ログ ファイルの数 ログ データが上記の最大サイズを超えると、ログは新たなファイルに書き出されます。ログのファイル数がこの項目で指定した値を超えると、最初の古いログ ファイルに上書きします。
Include Time-Stamp	はい	ログにログ時刻jを付加する否か
Fiorano.jmslog2.def.DefaultFormatter.dateformat （日付形式の指定）	MM.dd.yyyy (デフォルト設定では、05/17/2008 の形式で記録されます)	日付形式を指定します。 例) yyyy.MM.dd ---- 2008/05/17

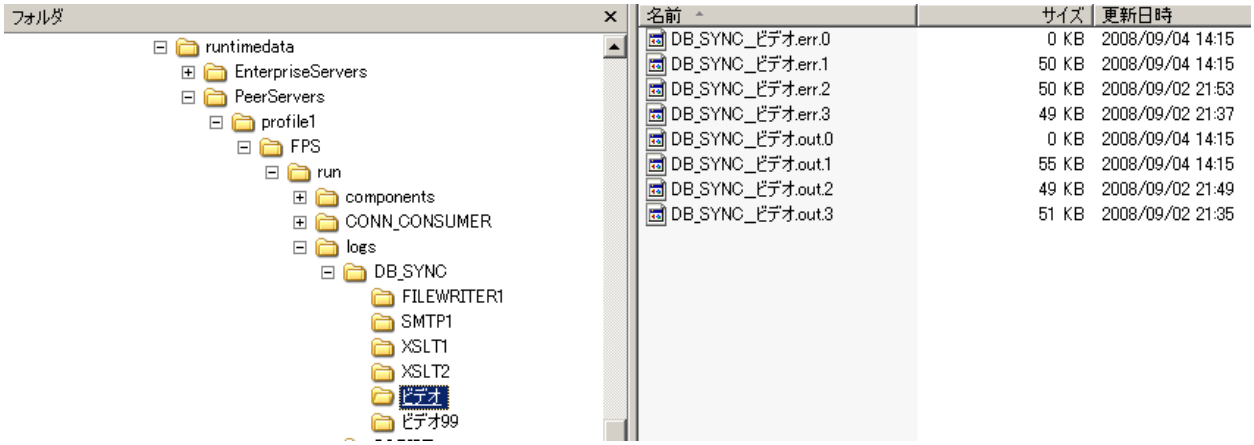
ログ記録はファイルにアペンド (追加) されながら書き出されます。ファイルがここで指定したサイズを超えると、新しいファイルを生成し、そこにログ記録が追加されていきます。ファイルの数がここで指定した値を越える場合には、最初のファイルに戻って書き出されます。その際、最初のファイルに記録されていたログは消去されます。



1.3.2 ログファイルのロケーション

コンポーネントのログ ファイルは、ピア サーバー実行マシンの次の場所に作成されます。

<インストール ディレクトリ> **/runtimedata/PeerServers/<プロファイル名>/FPS¥run/logs**



インストール ディレクトリ (Windows 版のデフォルトは、`C:¥Program Files¥Fiorano¥FioranoSOAXXXXXXX`)

`XXXXXX` は、バージョン番号を示しています。

runtimedata : コンポーネント フローの実行に必要なデータおよび実行中に保存されるデータやログなどの保存場所

プロファイル名 : ピア サーバーを起動する際に指定しているプロファイル名。プロファイルとはサーバーのコンフィグレーション パラメータの設定を指し、設定値はプロファイル名と同名のファイルに保存されています。デフォルトのプロファイル名は

"**Profile1**" です。ピア サーバーを Profile1 とは別のプロファイルを指定して起動している場合には、そのプロファイル名と同じ名前のディレクトリ下にログ ファイルが作成されます。

logs : ログ ファイルが保存される場所。コンポーネント フロー毎にディレクトリが作成され、その下にコンポーネント毎のディレクトリが作成される。

ログは、その種類に応じて次の 2 つに分けて保存されます。

out : 通常のシステム ログが保存されます。(ファイル名の例 : DB_SYNC__ビデオ 99.out.0)

err : エラーに関するログのみが保存されます。(ファイル名の例 : DB_SYNC__ビデオ 99.err.0)

ファイル名の最後にある数字は、ファイルの順番を指しています。0 のファイルがいっぱいになると、次に 1 のファイルが作成されます。0 → 1 → 2 → 3 とファイルが作成されていき、3 のファイルがいっぱいになると、0 のファイルにかきだされます。

1.4 ログ レベルとログ モジュール

ログ レベルの設定は、コンポーネントが使用しているログ モジュールの単位で指定します。コンポーネントの処理内容によって使用しているログ モジュールの種類は異なっています。例えば、DB コンポーネントでは、下のキャプチャ画面で示すログ モジュールが使用されており、それぞれデフォルトのログ レベルが設定されています。

モジュール **デフォルトのログ レベル**

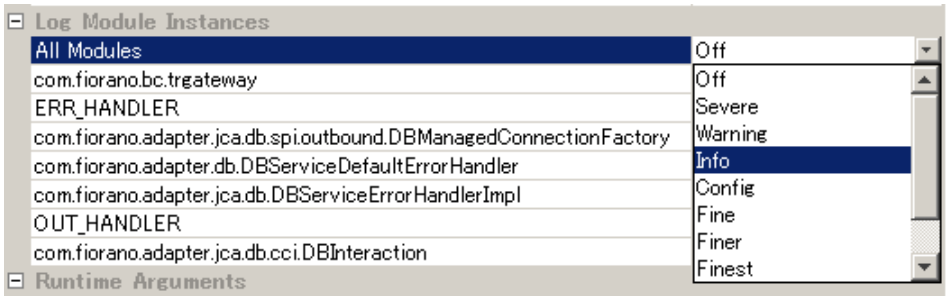
Log Module Instances	
All Modules	<異なる値>
com.fiorano.bc.trgateway	Severe
ERR_HANDLER	Severe
com.fiorano.adapter.jca.db.spi.outbound.DBManagedConnectionFactory	Severe
com.fiorano.adapter.db.DBServiceDefaultErrorHandler	Severe
com.fiorano.adapter.jca.db.DBServiceErrorHandlerImpl	Severe
OUT_HANDLER	Info
com.fiorano.adapter.jca.db.cci.DBInteraction	Severe

コンポーネントの種類を問わず、デフォルトのログ レベルの設定は、OUT_HANDLER ログ モジュールが "Info" に、その他のログ モジュールが "Severe" に設定されています。OUT_HANDLER ログ モジュールは、出力するメッセージ (データ) の生成に関わるログを書き出す場合に使用しているもので、製品にバンドルされているほとんどすべてのプリビルト コンポーネントが使用しています。

デフォルト設定の意味は、次のコンポーネントに渡すデータを出力する部分については詳細なログを記録し、コンポーネント内部処理については重大なエラーについてのみログを記録するというものです。この設定は、本番稼動に適した設定です。コンポーネントフローの開発中には、より詳細なログが書き出されるよう、すべてのログ モジュールについてログ レベルの設定を変更します。

ログ レベルの変更

ログ レベルの変更は、プロパティ ウィンドウで行います。ログ モジュール毎に変更してもいいのですが、**[All Module]** の項目を変更すればすべてのモジュールを一括して変更できます。



次のログ レベルを推奨します。

コンポーネント フローの開発中 : **Info**

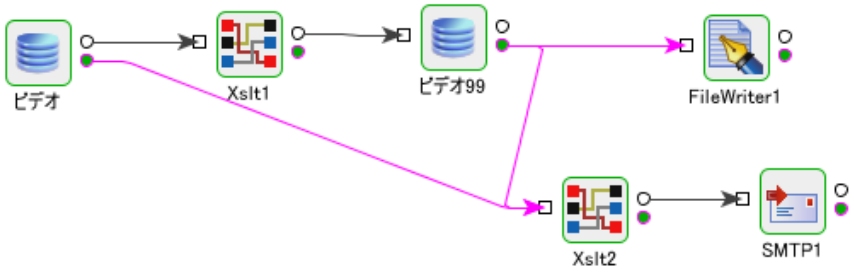
ユーザー独自のコンポーネントを開発している場合 :**Finest**

上記のログ レベルでデバッグに必要な情報を得られますが、より詳細なログ記録が必要な場合には、"All" を指定します。

2. Studio におけるログの表示

前述のログ ファイルを直接テキスト エディタなどで表示させてもよいのですが、ログの表示ツールも用意されています。
この章では、Studio のオーケストレータ イーゼルからログを表示する方法について説明します（後述するように Event Manager、Web Console によるログの閲覧方法もあります)。

このガイド ブックでは、下図のコンポーネント フローを例題として使用します。

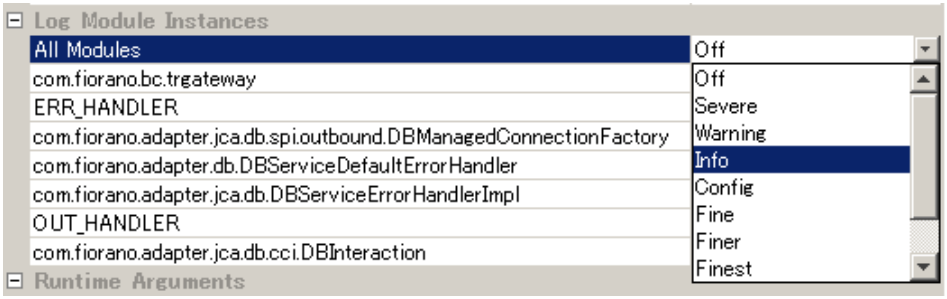


このコンポーネント フローは、ガイドブック『Fiorano オーケストレータによるビジネス プロセス構築の概要（基礎編)』でその構築方法を説明しています。

2.1 事前準備

事前の準備として、下記の設定を行い、エラーが起こる状況としておきます。

1. [ビデオ] コンポーネントのデフォルト設定を変更し、ログ レベルを “**Info**” にします。すべてのログ モジュールで、“**Info**” レベルのログが記録されるようにします。

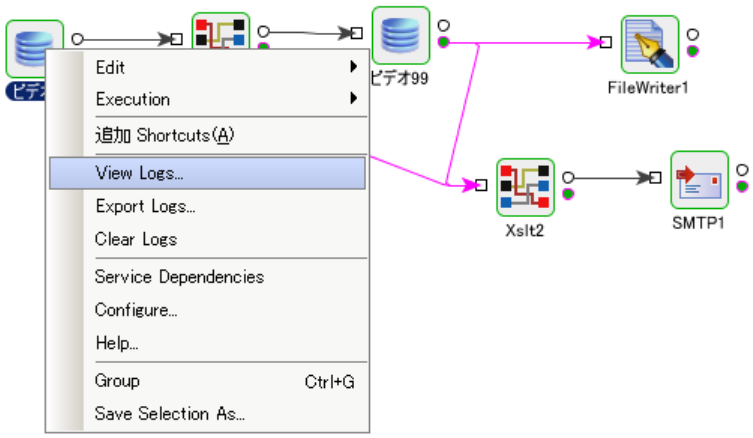


2. エラー状況を創出し、ログが記録されるようにします。
   フローの先頭にある [ビデオ] コンポーネントは、一定間隔（例 20 秒）で Mckoi データベースにアクセスし、SELCET クエリを実行するようスケジュールされています。
   エラー状況を作り出すためには、Mckoi データベースを起動させないで(停止したまま) コンポーネント フローを実行します。Mckoi データベースにアクセスできないため、Video コンポーネントがデータベースに接続しようとした時点でエラーが発生します。

上記の状況で、コンポーネント フローを実行します。

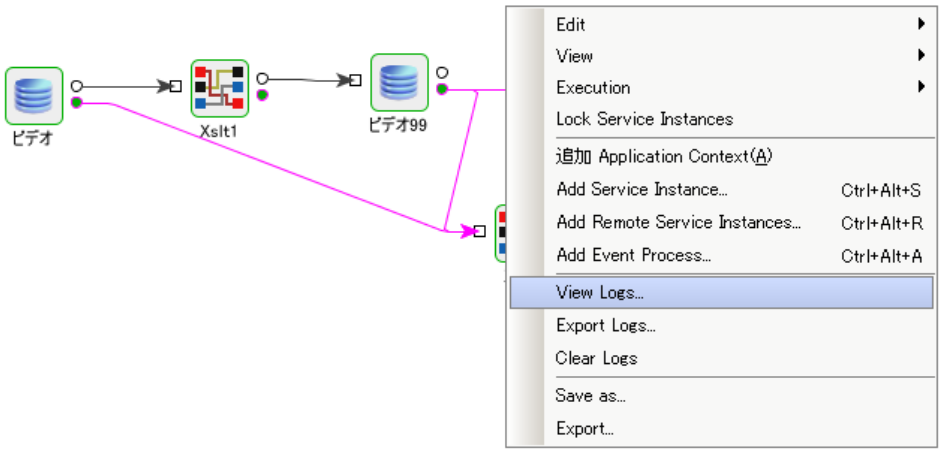
2.2 ログ ウィンドウの表示

ログを表示させるためには、任意のコンポーネントを右クリックし、メニューから **[View Logs…]** を選択します。



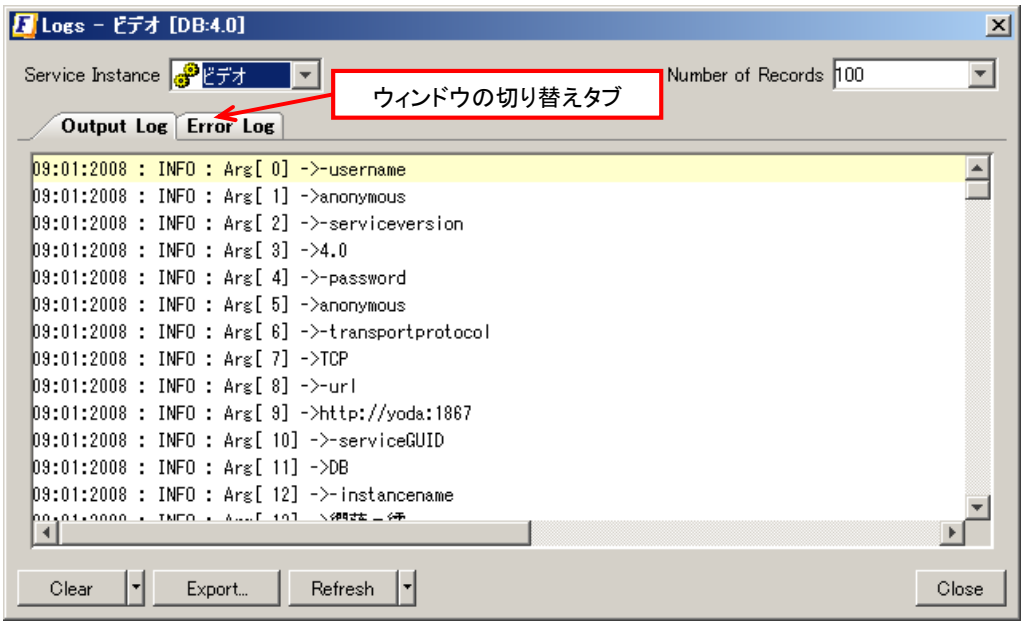
イーゼル上の任意の箇所を右クリックし、メニューから **[View Logs…]** を選択しても同じように、ログ ウィンドウが表示されます。

どちらの方法でもかまいません。



ログ記録は、コンポーネント フローが実行されている状態でも、停止している状態でも見ることができます。当然のことながら、停止している状態では、ログ ファイルに残されている過去のログだけとなります。コンポーネント フローを実行中であれば、次に説明するログ ウィンドウの **[Refresh]** ボタンで最新のログ記録を表示できます。

メニューから **[View Log…]** を選択すると、次のキャプチャ画面のように、ログ ウィンドウが現れます。

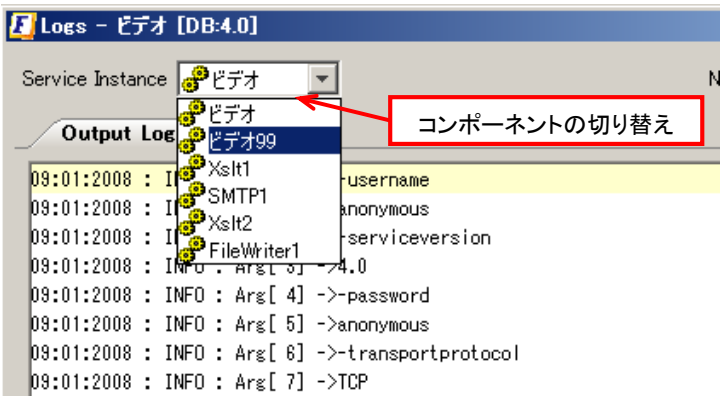


ログ記録は、通常のシステム メッセージを記録する out ログ ファイルと、エラー メッセージを記録する err ログ ファイルに分類されています。

ログ ウィンドウもこれに合わせて、**[Output Log]** と **[Error Log]** の 2 つのウィンドウを持っています。タブを選択することによって、ウィンドウを切り替えることができます。

ログ ウィンドウに表示されるログ記録は、コンポーネント毎に分かれています。

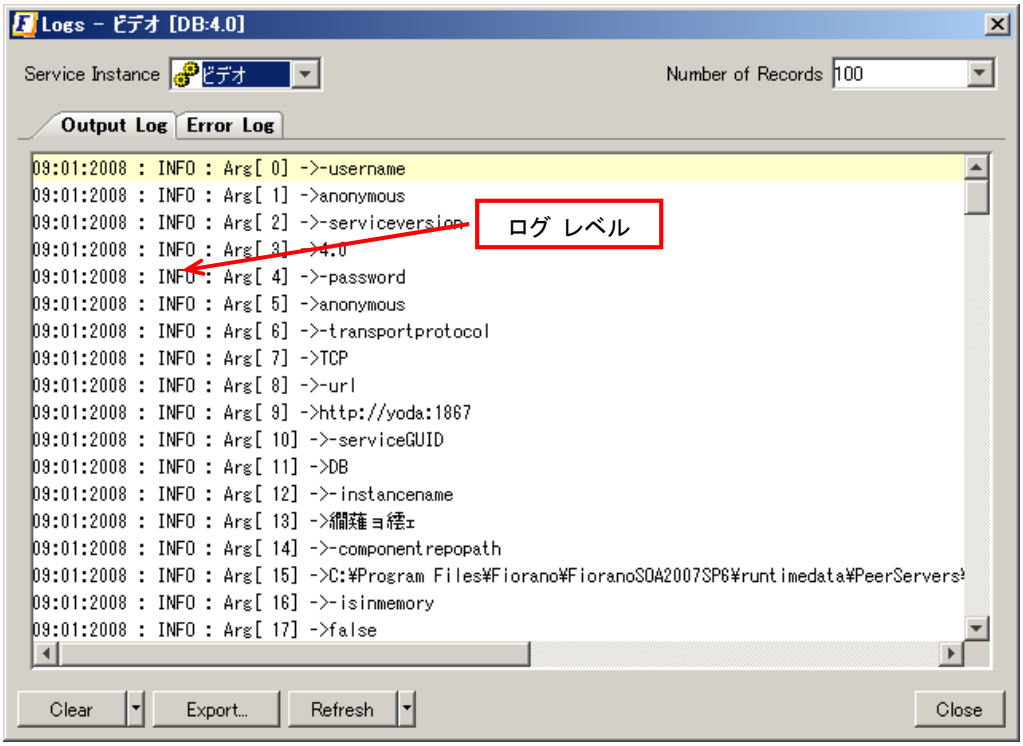
プルダウン メニューからコンポーネントを選択することで、表示するコンポーネントを切り替えることができます。



2.3 ログの解析

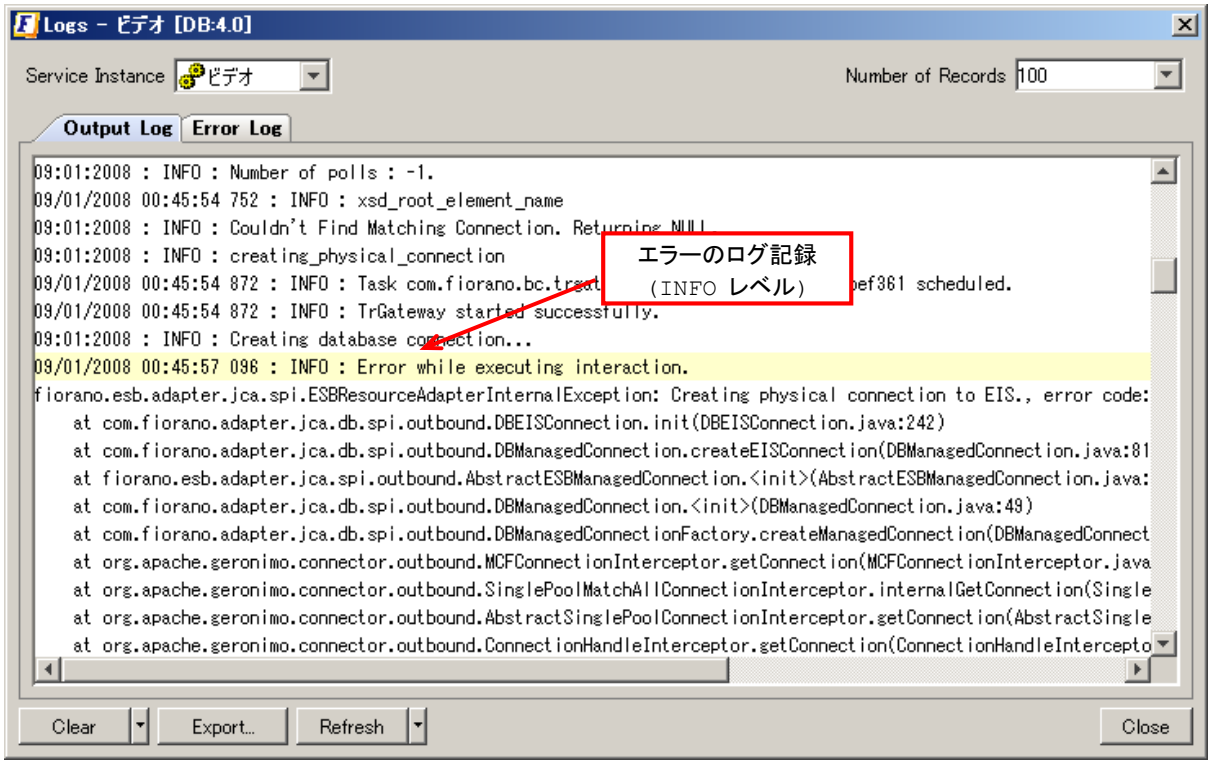
次のキャプチャ画面は、[ビデオ] コンポーネントの Output Log のものです。

INFO レベルのログが記録されていることが確認できます。最初の数行は、コンポーネントの起動に関するログです。これは INFO レベルで記録されるもので、デフォルト設定の Sever や Warning のレベルでは記録されないログです。



文字化けを起こしているのは、コンポーネント名です。コンポーネントの実行インスタンスの名前が日本語になっていると、このログ表示ツールでは文字化けを起こしてしまいます。実際のログ ファイルには、日本語のコンポーネント名が正しく記録されています。(次期バージョンで、修正する予定でいます。)

ログを下方向に順にみていくと、次の画面のようにエラーの発生が記録されていることがわかります。



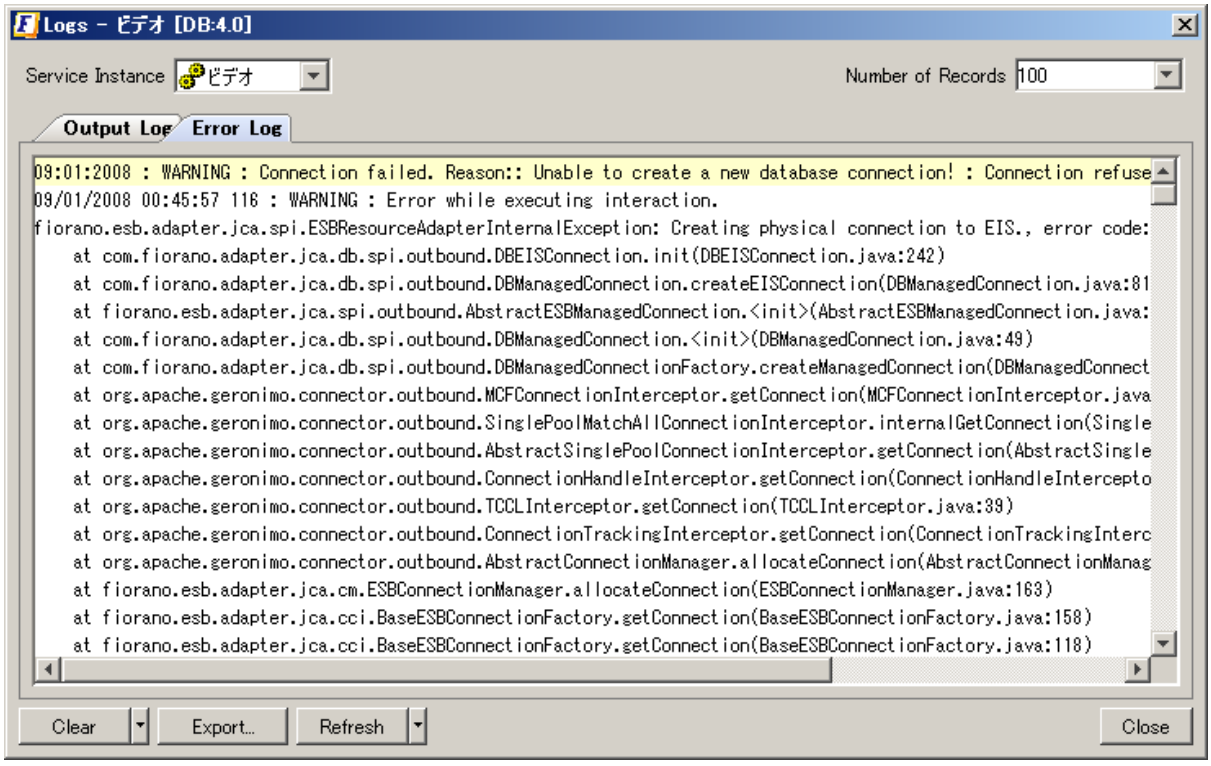
この前後のログは、次の意味を持っています。(重要と思われ部分は、黄色のマーカーで示しています。)

(DB 接続を開始)
09:01:2008 : INFO : Creating database connection... (DB への接続開始)

(09/01/2008 00:45:57 に DB への接続中にエラーが発生)
(EIS (Enterprise Information System (外部システム) への物理的な接続を生成中にエラーが発生。エラー コード : RESOURCE_CONNECT_ERROR)
09/01/2008 00:45:57 096 : INFO : Error while executing interaction.
fiorano.esb.adapter.jca.spi.ESBResourceAdapterInternalException: Creating physical connection to EIS., error code: RESOURCE_CONNECT_ERROR
:
:
:

(スケジューラに設定されている秒数 (20 秒) が経過すると、また接続を試みるが、同じようにエラーが発生)
09:01:2008 : INFO : Creating database connection...
09/01/2008 00:46:17 836 : INFO : Error while executing interaction.
:

[Error Log] ウィンドウは、エラー ログの内容を表示します。エラー ログでは、同一のエラー情報が WARNIG レベルで記録されます。

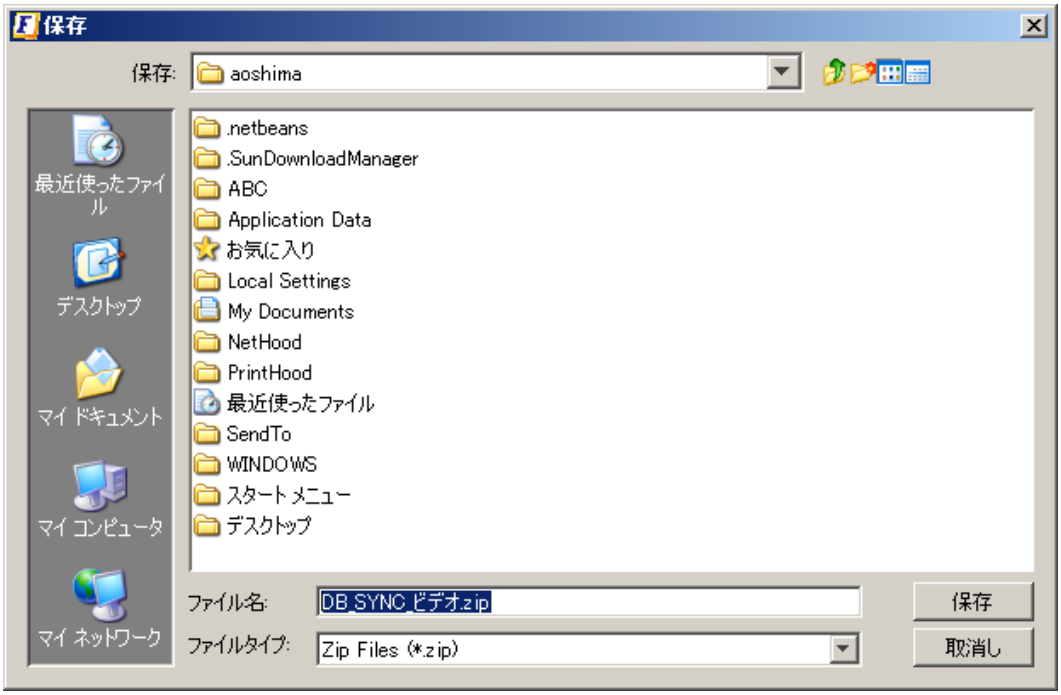


2.4 ログ記録の削除

ログ記録を削除するには、**[Clear]** ボタンをクリックします。Output Log および Error Log の両ログ記録を削除できます。一方の記録だけを削除したい場合には、ボタンの右にあるプルダウン メニューから選択してください。

2.5 ログ記録のエクスポート

ログ記録をファイルに書き出すには、**[Export]** ボタンをクリックしてください。該当するコンポーネントのログ ファイルを ZIP（圧縮）ファイルとして書き出すことができます。デフォルトのディレクトリは、**C:¥Documents and Settings¥[ユーザー]** となっていますが、任意の場所を選択できます。



3. エラー、障害の検出について

コンポーネントでエラーが発生した場合の処理は、CPS (カスタム プロパティ シート) で設定することになっています。[ビデオ] コンポーネントのエラー発生時の処理は、デフォルト設定となっています。

デフォルト設定におけるデータベース接続のエラー発生時の処理

エラー ポートにエラー メッセージを送信する

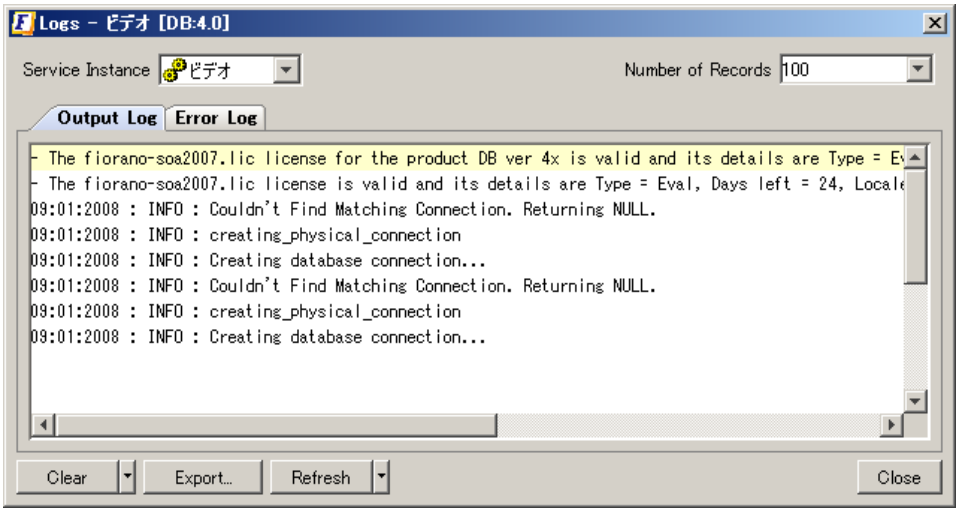
接続の再試行は行わない

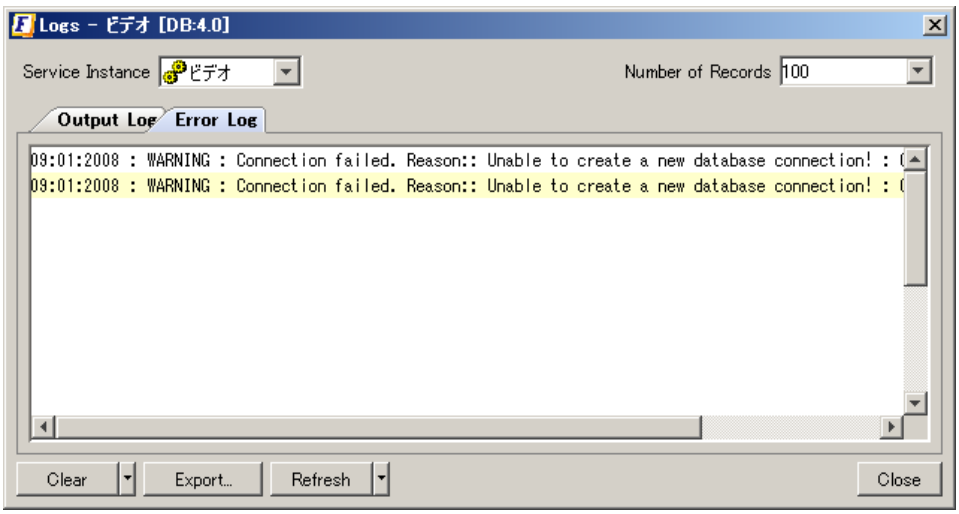
コンポーネントは停止しない

この設定により、[ビデオ] コンポーネントはデータベースへの接続に失敗しても、再接続を試行せずにそのまま動作し続けます。つまり、スケジューラによって 20 秒後にキックされるまで、待機します。そして、ログ記録が示すように、通常の処理としてデータベースに接続を試みます。しかし、データベースを起動していないため、前回と同じようにエラーが発生します。

データベースの負荷状態が高くなっていたり、多数のクライアントからの接続が多数重なっていたりすると、一時的にデータベースに接続できないことがあります。このため、コンポーネントを停止せずに、ある時間を置いた後に接続を試みることは、理にかなった処理方法といえるでしょう。

このことから、デフォルトのログ レベルの設定では、データベース接続のエラー発生はほとんど記録されないようになっています。SP6 からログ内容を改善し、デフォルト設定のログ レベルでも以下のように接続エラーを記録するようになりました。





エラーや障害の検出には、次の 2 つが考えられます。

- ログを常に監視する
- エラー ポートから送信されるエラー メッセージを受け取る

ログ記録を常に監視することで障害の発生を知ることができますが、例題のコンポーネント フローのようにエラー メッセージをシステム管理者が受け取れるように構築しておくことで、より効率的にエラーの発生を知ることができるようになります。

エラーが発生した場合には、ログ レベルの設定とは関係なく、必ずエラー メッセージがエラー ポートから送信されます。このエラー メッセージを受信しシステム管理者に通知する処理をコンポーネント フローに組み込んでおくことが、たいへん重要な意味を持ちます。運用面から考えると、ログ レベルを上げて詳細な記録を取ることは、ディスク スペースの消費やコンポーネント フローの処理パフォーマンスの低下を招きます。安定した動作が見込める本番稼動では、ログ記録を最小限に押さえ、エラー ポートから送信されるエラー メッセージをシステム管理者が確実に受け取れるフローを構築することが効果的な運用方法です。発生したエラーや障害の解析や解消方法の検討時に、ログレベルを上げ、詳細な記録を取るようにします。

イベントのアラートについて

ESB サーバーやピア サーバーの起動、停止などのシステム イベントは、ESB サーバーの runtime ディレクトリに記録されます。

同時に、システム管理者にメールで通知するアラート機能が用意されています。

アラート通知は、システム イベントのカテゴリ単位で登録できます。

システム イベントのカテゴリ

- ESB サーバーのシステム イベント (起動、停止)
- ピア サーバーのシステム イベント (起動、停止)
- セキュリティ関連のシステム イベント (コンポーネント フローやコンポーネントの変更などのアクセス)
- コンポーネント フローのシステム イベント (フローの起動、停止など)
- サービス コンポーネントのシステム イベント (コンポーネントの起動、停止など)

アラートの登録は、Fiorano Web Console で行うことができます。詳細は、製品マニュアルを参照してください。

4. Event Manager によるログの表示

4.1 Event Manager の起動

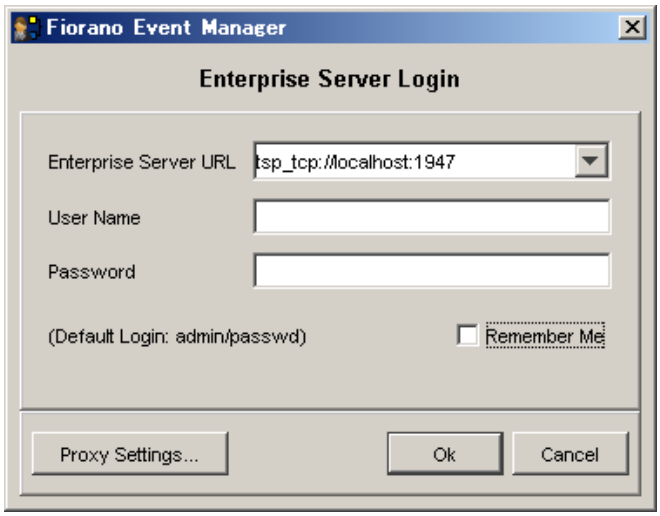
Event Manager は、スクリプト **evst.bat** (Unix / Linux 版の場合は、evst.sh) によって起動できます。

場所

（インストール ディレクトリ）**/esb/tools/evst/bin**

Windows 版では、スタート メニューからも起動できます。

Event Manager が起動すると、次のログイン画面が表示されます。



Enterprise Sever URL

ログインする ESB サーバー (Enterprise サーバー) の URL を指定します。

デフォルト値として次の値があらかじめ設定されています。

tsp_tcp://localhost:1947

デフォルトの ESB サーバー コンフィグレーションでは、各ツールは ポート 1947 に TCP プロトコルによって接続するよう設定されています。ESB サーバーのコンフィグレーション設定を変更している場合は、それに合わせて URL を変更します。

また、別マシン上の ESB サーバーにログインする場合には、localhost ではなく、ESB サーバーの IP アドレス (またはリモート ホスト名) を指定します。

User Name、Password

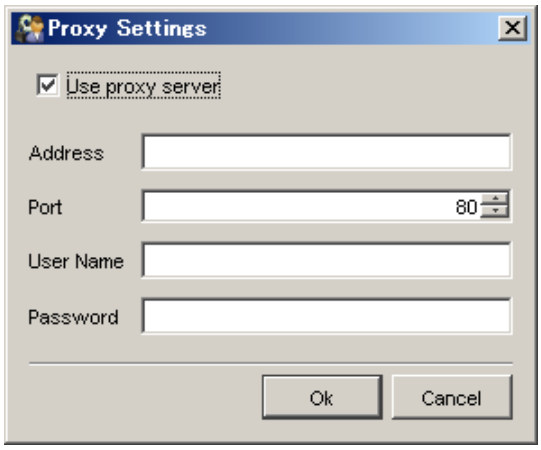
登録されているユーザー名とパスワードを指定します。

初期状態で設定されているユーザー `admin` には、すべての権限が与えられています。パスワードは `passwd` です。

Proxy Settings

プロキシ サーバーを介して ESB サーバーに接続する場合には、[Proxy Settings …] ボタンをクリックします。表示されるダイアログ ボックスに、プロキシ サーバーへの接続情報を入力します。

情報を入力したら、**[Ok]** ボタンをクリックし、最初のログイン画面に戻ります。



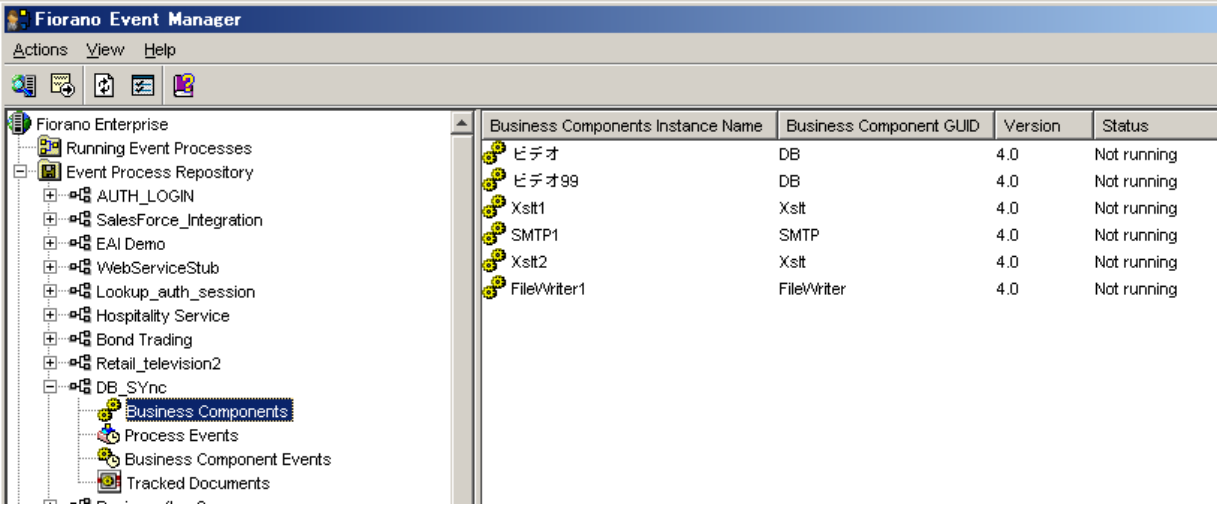
[注意]

ESB サーバーが起動していないと、Event manager にはログインできませんので、注意してください。

また、コンポーネント フローのログを表示するためには、ピア サーバーが起動されている必要があります。

4.2 ログの表示

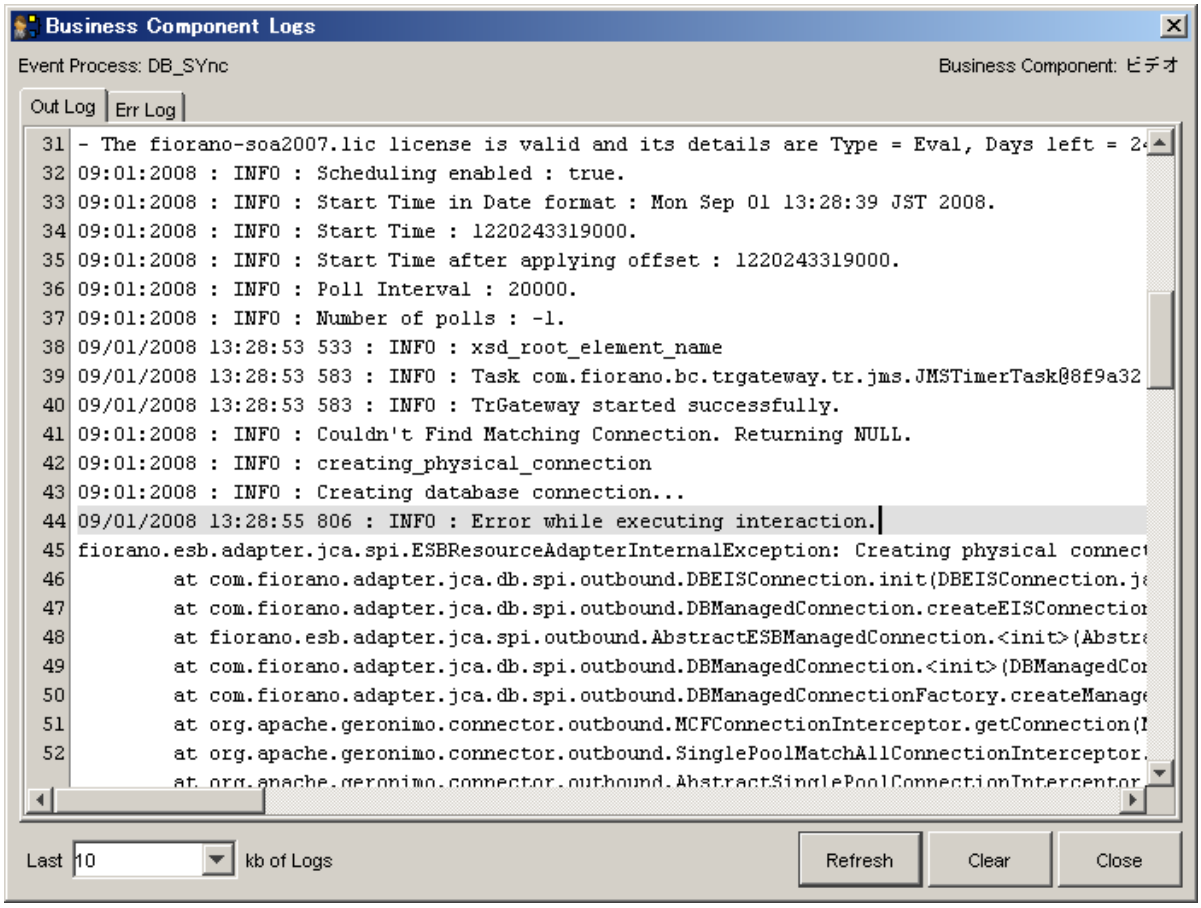
ログインに成功すると、次のようにリポジトリに保存されているコンポーネント フロー (Event process) の一覧が表示されます。



一覧から DB_Sync を選択し、ツリー表示を展開します。Business Components を選択すると、コンポーネント フロー内で使用されているコンポーネントが右側のペインに表示されます。

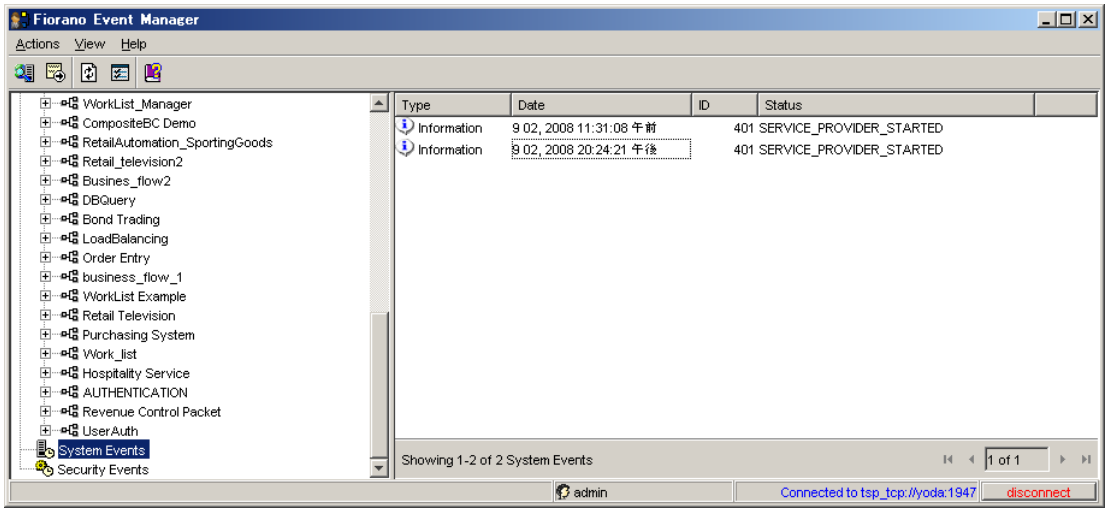
[ビデオ] コンポーネントを右クリックしてプルダウン メニューから [View Logs …] を選択すると、ログ ウィンドウが表示されます。

[ビデオ] コンポーネントをダブルクリックしても、ログ ウィンドウを表示させることができます。

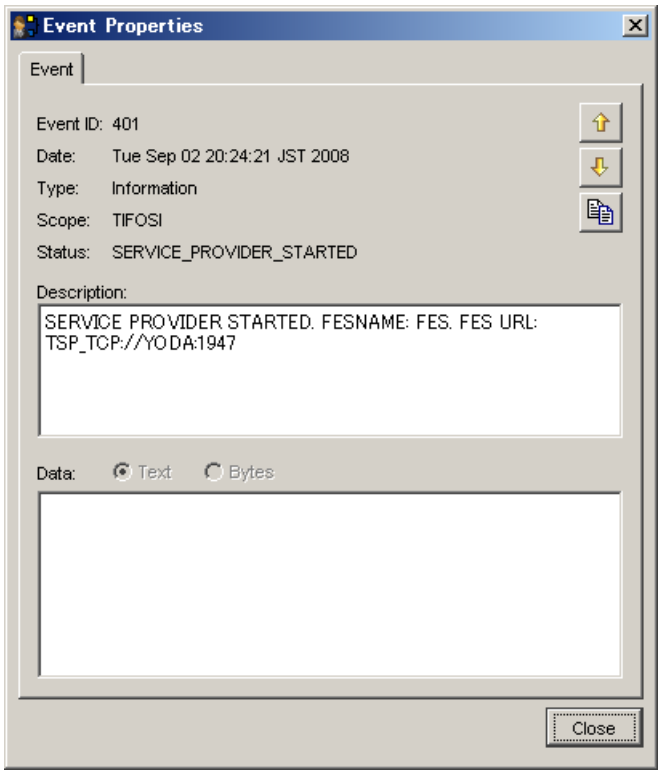


ログ ウィンドウの操作方法は、前章で説明した Studio のログ ウィンドウと同じです。

Event Manager では、コンポーネントのログの他に、システム イベントとセキュリティ イベントも閲覧することができます。システム イベントおよびセキュリティ イベントは、左側のツリー表示の最下段にあります。

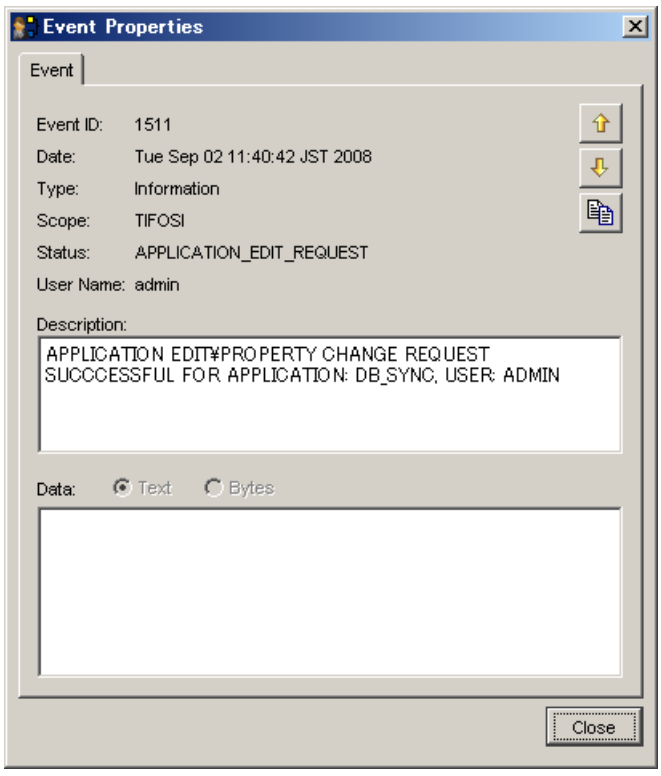


右ペインの項目の 1 つを選択し、詳細を表示させます (右クリックして **[Properties]** を選択するか、ダブルクリックします)。



ESB サーバー (FES) のサービス プロバイダーが起動された日時が、システム イベントとして記録されていることが分かります。サービス プロバイダーとは、ESB サーバーの各サービス (機能) をツールや Peer サーバーに提供する機能のことで、ほとんど ESB サーバーと同義です。

セキュリティ イベントでは、次の詳細表示画面が示すように、コンポーネント フローの編集 (edit) リクエストがユーザー admin から出され、これを許可したことがわかります。



5. Web Console によるログの表示

5.1 Web Console の起動

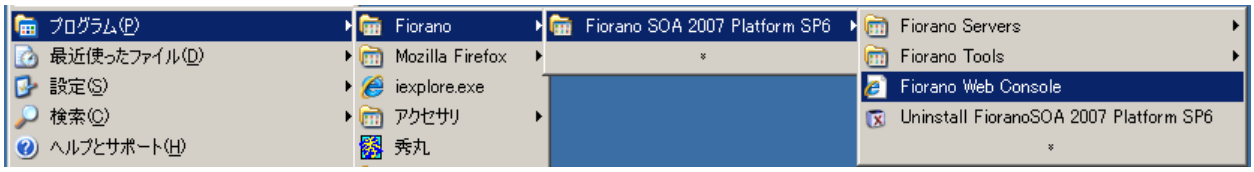
Web Console は、web ブラウザ ベースの監視ツールです。

Web Console を起動するには、任意の web ブラウザで、次の URL を指定します。

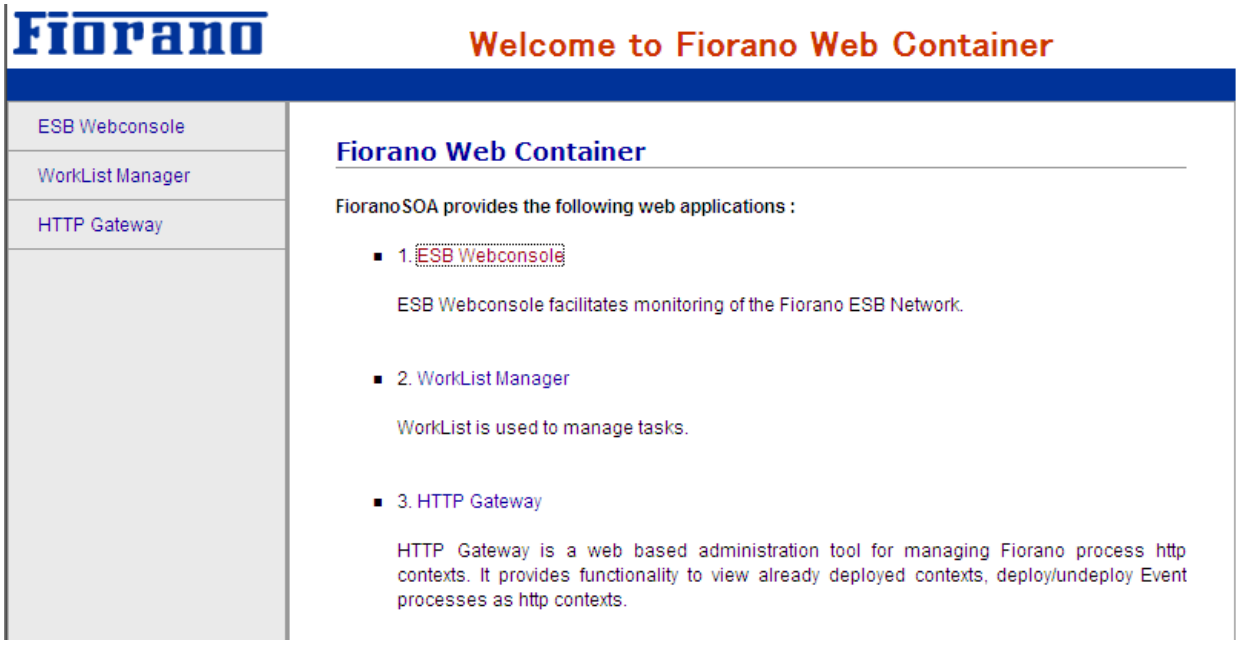
http://localhost:1980/　　（ESB サーバー同じマシン上でブラウザを起動する場合）

http://<ESB サーバーの URL（IP アドレス）>:1980/　　（別マシンのブラウザを使用する場合）

Windows 版では、スタート メニューから起動することもできます。この場合、ブラウザはシステムに登録されている規定のブラウザが使用されます。

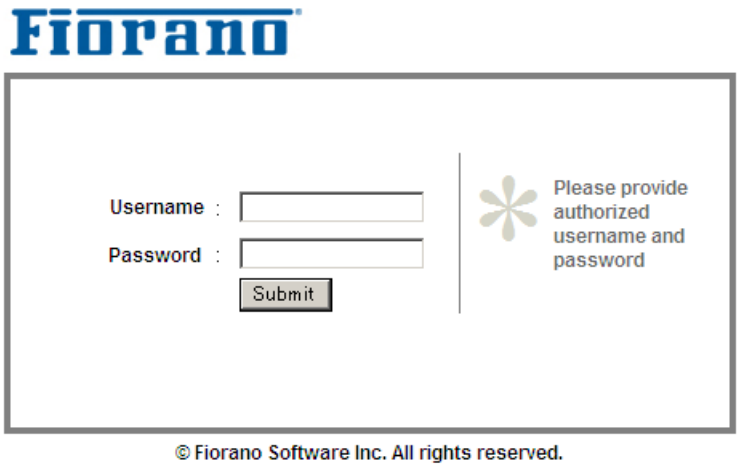


ブラウザで最初に表示されるのは、ESB サーバーに組み込まれている Web コンテナー (jetty) へのアクセス ページです。



[1. ESB Webconsole] をクリックし、Web Console を起動します。

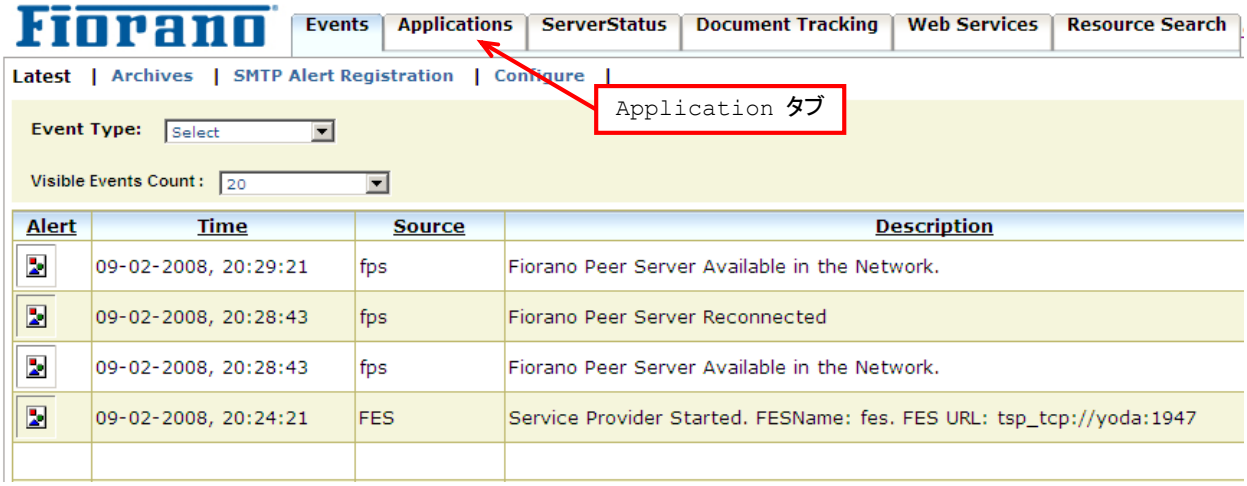
次のキャプチャ画面のように、ログイン 画面が表示されます。



デフォルトで設定されているユーザー名 (admin)、パスワード (passwd) を入力し、ログインします。

5.2 ログの表示

ログインに成功すると、次の画面が表示されます。



最初に表示されるのは、システム イベントの画面です。

コンポーネント フローのログを表示するには、**[Application]** タブをクリックします。

Application タブでは、リポジトリにあるすべてのコンポーネント フローの状況が表示されます。

下段にあるページ ボタンで目的のコンポーネント フローが表示されているページに移動します。

DB_Sync の表示をみると、次の状態にあることがわかります。

Status : Running （実行中）

Category : (TEST) （リポジトリ上で格納されているディレクトリ名）

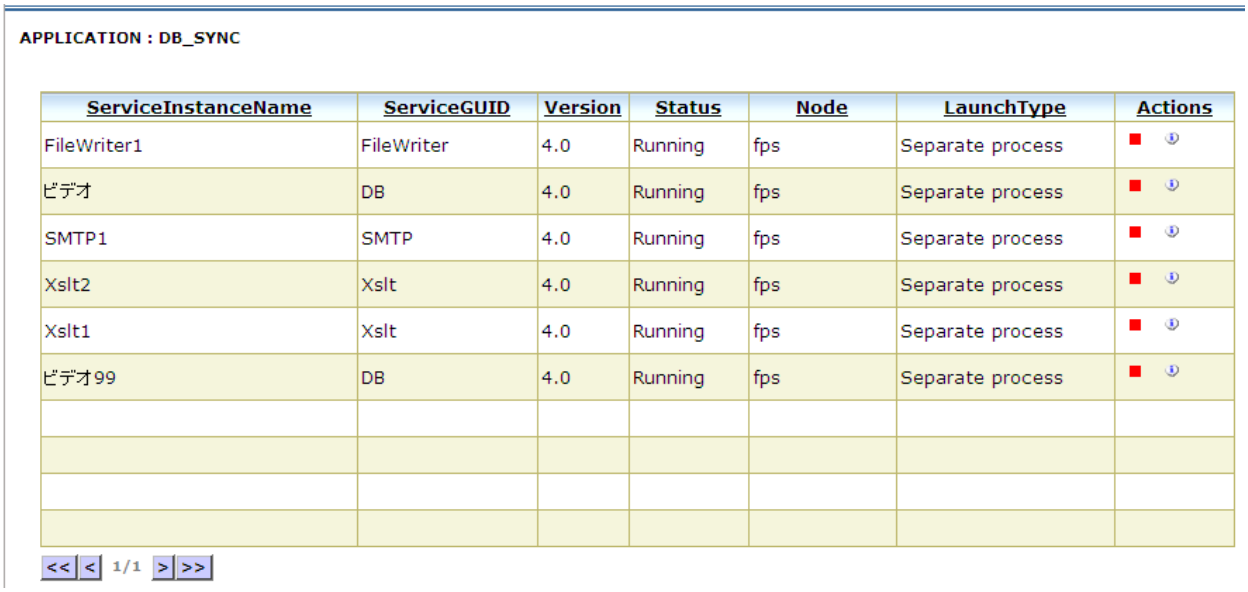
Node : fps （稼動している ピア サーバーの名前）

Actions  (このコンポーネント フローに対して実行できるアクション)

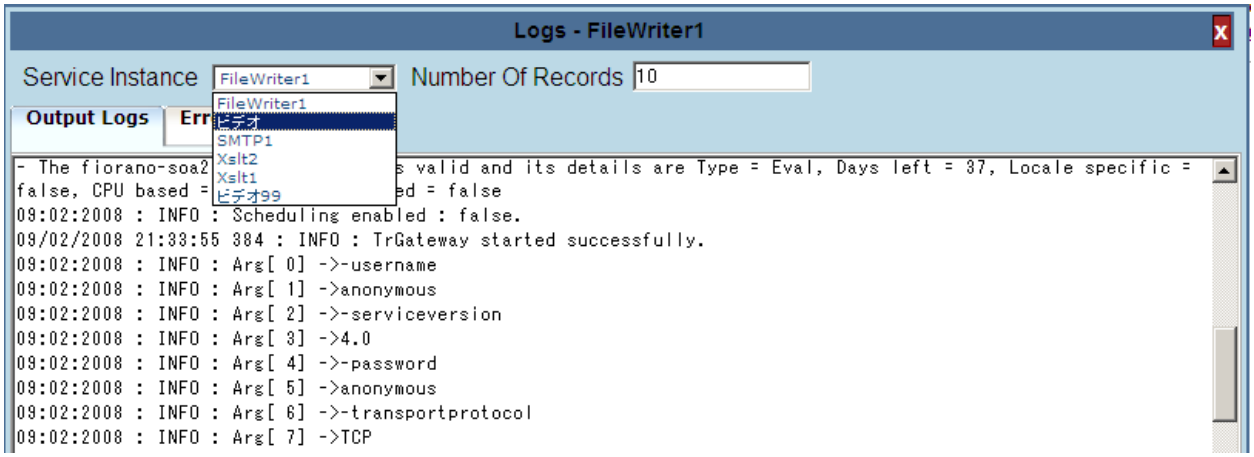
Action の欄に表示されているアイコンは、左から「再起動」、「停止」、「ログの表示」、となっています。

停止状態にあるコンポーネント フローの場合には、「起動」と「ログの表示」 のアクションが実行できます。

DB_SYNC フローをクリックして選択状態にした上でこのページを下方にスクロールすると、DB_SYNC の各コンポーネントの状態が表示されます。



Action 欄からログの表示をクリックすると、次のようにログ表示画面が表示されます。



コンポーネントをプルダウン メニューから選択し、コンポーネントのログを表示させます。ページ下段にあるコンポーネント一覧からログ表示をクリックしても、コンポーネントのログを表示させることができます。

6. ログ ファイルのハンドリング用 API

Fiorano SOA プラとフォームは、ログ ファイルを処理するための API を提供しています。

6.1 API の一覧

青い太字で示した箇所には、ユーザー任意の値を指定します。

ESB サーバーのシステム イベント ログ

`FioranoServiceProvider.getTESLastErrLogs(int **numberOfLines**)`

Err ログ ファイルの最後から指定行数のログ記録を取り出す

`FioranoServiceProvider.getTESLastOutLogs(int **numberOfLines**)`

Out ログ ファイルの最後から指定行数のログ記録を取り出す

ピア サーバーのシステム イベント ログ

`FioranoServiceProvider.getFPSManager().getTPSLastOutLogs(int **noOfLines**,`
`String **fpsName**)`

指定したピア サーバーの Err ログ ファイルの最後から指定行数のログ記録を取り出す

`FioranoServiceProvider.getFPSManager().getTPSLastErrLogs(int **noOfLines**,`
`String **fpsName**)`

指定したピア サーバーの Out ログ ファイルの最後から指定行数のログ記録を取り出す

コンポーネントのログ

`FioranoServiceProvider.getApplicationController().getLastOutTrace(int numberOfLines,`
`String **serviceName**, String **appGUID**, float **appVersion**)`

指定したコンポーネントの Out ログ ファイルの最後から指定行数のログ記録を取りだす

ServiceName : コンポーネント名

appGUID : コンポーネント フローの名前 (GUID)

appVersion : コンポーネント フローのバージョン番号

`FioranoServiceProvider.getApplicationController().getLastErrorTrace(int numberOfLines,`
`String **serviceName**, String **appGUID**, float **appVersion**)`

指定したコンポーネントの Out ログ ファイルの最後から指定行数のログ記録を取りだす

ServiceName : コンポーネント名

appGUID : コンポーネント フローの名前 (GUID)

appVersion : コンポーネント フローのバージョン番号

6.2 サンプル プログラム

次のプログラム ソースは、ログ記録を取り出すプログラムの例です。
青い太字で示している値は、ログを取り出すコンポーネント フローの GUID とコンポーネント名、ピア サーバーの名前、ユーザー名とパスワードで、ユーザー独自の値を設定します。

```
import com.fiorano.esb.rtl.server.FioranoServiceProviderFactory;
import com.fiorano.esb.rtl.server.FioranoServiceProvider;

import javax.naming.Context;
import javax.naming.InitialContext;
import java.util.Hashtable;

public class RTLTest {
    private static String serviceInstanceName = "chat1";
    private static String appName = "SIMPLECHAT";
    private static String peerName = "fps";

    public static void main(String[] args) throws Exception {
        Hashtable env = new Hashtable();

        env.put(Context.SECURITY_PRINCIPAL, "admin");
        env.put(Context.SECURITY_CREDENTIALS, "passwd");
        env.put(Context.PROVIDER_URL, "tsp_tcp://localhost:1947");
        env.put(Context.INITIAL_CONTEXT_FACTORY,
"fiorano.tifosi.jndi.InitialContextFactory");

        InitialContext ic = new InitialContext(env);

        System.out.println("Created InitialContext :: " + ic);

        FioranoServiceProviderFactory fspf = (FioranoServiceProviderFactory)
ic.lookup("TifosiServiceProvider");

        FioranoServiceProvider fsp = fspf.createServiceProvider("admin", "passwd");

        System.out.println("Enterprise Server Output Logs : ");
        System.out.println(fsp.getTESLastOutLogs(100) + "¥n¥n");
        System.out.println("Enterprise Server Error Logs : ");
```

```
        System.out.println(fsp.getTESLastErrLogs(100) + "¥n¥n");

        try {
            System.out.println("Peer Output Logs for " + peerName + " : ");
            System.out.println(fsp.getFPSManager().getTPSLastErrLogs(100, peerName) +
"¥n¥n");
            System.out.println("Peer Error Logs for " + peerName + " : ");
            System.out.println(fsp.getFPSManager().getTPSLastOutLogs(100, peerName) +
"¥n¥n");
        } catch (fiorano.tifosi.common.TifosiException tifosiException) {
            System.out.println("Failed to get logs for peer server " + peerName + ". Check if
peer server " + peerName + " is running");
        }

        System.out.println("Service Output logs for " + serviceInstanceName + " from  " + appName
+ " : ");
        System.out.println(fsp.getApplicationController().getLastOutTrace(100,
serviceInstanceName, appName, 1.0f) + "¥n¥n");
        System.out.println("Service Error logs for " + serviceInstanceName + " from  " + appName
+ " : ");
        System.out.println(fsp.getApplicationController().getLastErrorTrace(100,
serviceInstanceName, appName, 1.0f) + "¥n¥n");
    }
}
```