jvmservice について (ドラフト)

seraphy

2005 年 5 月 30 日

目次

1 はじめに

1.1 本ドキュメントについて

本ドキュメントは 2005 年 5 月 30 日現在の開発中バージョンの jvmservice の現状について説明している。**本ドキュメントはドラフト版であり、正確でも完全でもない。**

1.2 jvmservice とは

1.2.1 目的・用途

jvmservice は、Windows2000/XP/2003 上でのサービスプロセスとして JAVA 言語で記述されたクラスファイルを起動させることができるようにするランチャーアプリケーションである。

これは主に、自作の JAVA で記述されたサービスアプリケーションを、Windows のサービスとして統合できるようにするためのツールとして使われることを想定している。既存の製品群、たとえば Apache Tomcat などには、すでにサービス化するための機構を自分でもっており、そのようなものにとってかわる利点は無い。

そのかわり jvmservice はコンソールアプリケーションとして設計された JAVA アプリケーションを、比較的簡単な修正により、サービスに対応したサーバアプリケーションにアップグレードするための機構を提供する。

また、自作アプリケーションが何個あってもサービス化できるように、jvmservice ではいくつでもサービスを作成することができる。

1.2.2 コンセプト、実現方法

jvmservice はサービス名を登録するとき、JavaVM を起動するパラメータと実行するクラスなどの情報を格納した**起動パラメータ定義ファイル**ををサービスに関連づける。そのため、jvmservice は 1 つの実行ファイルで、いくつもの JAVA アプリケーションをサービスとして実行できる。

jvmservice が実際にサービスとして起動されたとき、jvmservice では JNI の機構を用いてサービスプロセス内に JavaVM を作成しクラスをロードし実行する。このとき、サービスに関連づけられた**起動パラメータ定義ファイル**から必要な JavaVM のパラメータや、実行するクラスなどの情報を取得する。

jvmservice プロセスは指定された任意の (複数・並列可) クラスの静的メソッドを呼び出し、これらの全てのスレッドが停止するとサービスプロセスを終了し停止状態となる。

jvmservice に停止指示が送られるか、オペレーティングシステムがシャットダウンを開始すると、jvmservice サービスプロセスは JavaVM に対して新しい停止用スレッドを作成し、JavaVM 内から `java.lang.System#exit(0)` を実行する。

これにより、JavaVM は `java.lang.Runtime#addShutdownHook(java.lang.Thread)` で登録されたシャットダウンフックを実行してから JavaVM が停止し、JavaVM が

サービスプロセスに対して `exit` するように要求し、サービスプロセスが終了する。

1.3 開発・動作環境

1.3.1 オペレーティングシステム

jvmservice が対象とするオペレーティングシステムは以下のものである。

- Windows2000 Professional/Server
- WindowsXP Professional

おそらく、Windows2000 以降の NT 系の OS であれば動くものと思われるが、動作確認はしていない。

1.3.2 JAVA 環境 (JDK/JRE)

jvmservice の開発にあたり対象とした JavaVM は、Sun Microsystems が提供する JDK/JRE のうち、以下のものである。

- JDK/JRE 1.3.1_15
- JDK/JRE 1.4.2_08
- JDK/JRE 1.5.0_02

jvmservice は、JNI1.2 仕様に基づいて JavaVM を作成し、操作している。

この仕様に適合するバージョン 1.3.1 以降の Java 環境であればどれでも動作すると思われるが、動作確認はしていない。Sun Microsystems 以外の JavaVM での動作確認はしていない。

1.3.3 開発言語・環境

jvmservice は、Microsoft Visual Studio.NET2003 の Visual C++ によって記述されている。

MFC/ATL などのライブラリは使っていない。

1.4 ライセンス

2 導入デモンストレーション

jvmservice の利用方法として以下にデモンストレーションを示す。

2.1 入手方法

sourceforge の jvmservice プロジェクトから、jvmservice の実行ファイルとサンプル用 JAVA コードを取得する。

ファイル名は 2005 年 5 月 30 日現在では、以下のようになっている。(実際にはファイル名の後ろにバージョンかリビジョンを示す数字がつけられている。)

- jvmservice.zip
- jvmservice_binary.zip
- sample.zip

jvmservice.zip は、ソースコードを含む完全なセットであり、jvmservice_binary.zip の内容と、sample.zip の内容を含んでいる。とりあえず、jvmservice.zip をダウンロードすれば全て揃うので手っ取り早いといえる。

2.2 内容説明

導入デモンストレーションでは、以下のファイルを利用して行う。

- jvmservice.exe
- jvmservice.preload
- sample/jvmserv1.conf
- sample/SampleJVMService.java

これらのファイルの中身について説明する。

2.2.1 jvmservice.exe

jvmservice.exe は本アプリケーションの中核である。このファイルはサービスとして起動される実行ファイル本体であると同時に、サービスの登録と登録解除を行うツールでもある。

2.2.2 jvmservice.preload

jvmservice.preload は **jvmservice.exe** がサービスとして起動されたときに読み込むダイナミックリンクライブラリ (DLL) の一覧を記述したテキストファイルであり、同梱されているのは、その雛形である。

使用するマシンの環境にあわせて、このファイルで起動する JavaVM の実際の DLL「**jvm.dll**」のフルパスを指定する必要がある。

2.2.3 sample/jvmserv1.conf

sample/jvmserv1.conf はサービスとして起動されるときに JavaVM に渡される起動パラメータと、実行するクラスや、その引数を記述した「起動パラメータ定義ファイル」であり、同梱されているのは、その雛形である。

導入デモンストレーションでは、このファイルをもとに説明を行う。

2.2.4 sample/SampleJVMService.java

sample/SampleJVMService.java はデモンストレーション用の JAVA ソースコードである。

導入デモンストレーションでは、このファイルを javac によりコンパイルし java.exe で実行するケースと、それをサービスに登録するための手順について説明を行う。

2.3 サンプル用 JAVA コードの実行

2.3.1 SampleJVMService の説明

jvmservice はサービスとして実行するため画面等で動作を確認できない。

そこで、まず、サンプル用プログラムが、どのような動きをするものかを確かめる。

この sample.zip には、SampleJVMService.java という、単純に 1 秒間に一度の間隔でファイルに数字を出力する無限ループのプログラムがある。

このプログラムは起動オプションとして出力先ファイルへのパスを 1 つ指定する必要がある。

2.3.2 ソースコード (抜粋)

以下は、SampleJVMService.java のメインメソッドを抜粋したものである。

```
public static void main( final String[] args ) throws Exception {
    final PrintStream out = new PrintStream(
        new FileOutputStream( args[ 0 ]  ) );

    final Thread mainThread = Thread.currentThread();

    final Thread shutdownThread = new Thread() {
        public void run() {
            out.println( "シャットダウン要求開始" );
            for( int idx=0; idx<10 && mainThread.isAlive(); idx++) {
                mainThread.interrupt();
                try{
                    mainThread.join( 100 );
                }
                catch( InterruptedException e ) {
                }
            }
            out.println( "シャットダウン要求完了" );
            out.close();
        }
    };
    Runtime.getRuntime().addShutdownHook( shutdownThread );

    int idx = 0;
    for( ;; ) {
        try{
            out.println( idx++ );
            Thread.sleep( 1000 );
        }
        catch( InterruptedException e ) {
            break;
        }
    }
    out.println( "終了します。" );
}
```

2.3.3 SampleJVMService の実行

以下は C:/temp/sample というフォルダ上にコンパイルされた **SampleJVMService.class** が配置されていると仮定する。

```
java.exe -cp C:/temp/sample SampleJVMService C:/temp/test.txt ↵
```

C:/temp/test.txt ファイルが作成され、数字が出力されていることを確認する。

2.3.4 終了処理

終了するにはコンソール画面で CTRL-C の押下によって java.exe を強制終了させる。

画面上には「終了します。」と、つづけて「シャットダウン要求完了」と出るはずである。

SampleJVMService.java では、`java.lang.Runtime#addShutdownHook()` を用いて、シャットダウンフック用スレッドを登録している。

そのため、CTRL-C を受理した JavaVM が exit を実行しようとした結果、シャットダウンフックが動作し、それがメインスレッドに対して `java.lang.Thread#interupt()` による割り込みを発生させ、メインスレッドのループが中断していることがわかる。

サービスプロセスとして実行させるクラスは何らかのシグナルが発生しないかぎり無限ループ構造になっているものが多いと考えられる。

このようなプログラムを jvmservice でサービス化するためには、jvmservice の停止要求である `java.lang.System#exit(0)` によってプログラムを安全にシャットダウンできるようする必要がある。

実際にサービス化するまえに、まずコンソール画面上での CTRL-C の押下により、シャットダウンフックでの後始末が開始され、プロセスが安全に終了することを確認する必要がある。

2.4 サービス登録／登録解除ツール

jvmservice.exe はサービス登録／解除ツールであると同時に、実際にサービスプロセスとしても起動される実行ファイルである。

サービスを登録するには jvmservice を`--regist` オプションで起動する。

コマンドラインには以下のような形式で指定する。

```
jvmservice.exe --regist サービス名 --config 起動パラメータファイル
    [--log ログファイル ] [--workingdir 作業ディレクトリ]
    [--dependencies [依存サービス名 ...] ]
```

サービスとして登録するまえに、まず、jvmservice.exe を適切な場所に配置しておく必要がある。

jvmservice.exe の起動オプションで`--regist` と指定することで、起動された jvmservice.exe の 現時点での実行ファイルの存在するパス がサービス起動パスとして登録される。

したがって、一度サービスとして登録したら、サービスを登録解除しないかぎり jvmservice.exe を削除したり他の場所に移動させてはならない。

また、見てのとおり、JAVA 起動のためのパラメータは直接指定されていない。これは「起動パラメータファイル」に記述しておき、サービスに登録する際にはそのファイルへのパスを登録する。

「起動パラメータファイル」については後述する。

2.4.1 登録オプションの簡単な説明

以下に登録のためのオプションの説明を簡単に行う。

- `--regist` 指定したサービス名で登録する
- `--config` サービスが実行する JavaVM の起動パラメータ、および実行対象クラスなどの定義が格納されたファイルを指定する。
- `--log` JavaVM および jvmservice が出力するログメッセージを記録するファイルを指定する。(デフォルトは出力しない)
- `--workingdir` サービスが起動されたとき指定されたディレクトリをカレントディレクトリに変更する。(デフォルトは変更なし)
- `--dependencies` jvmservice が起動する際に依存する他のサービス名を列挙する。(デフォルトは無し)

少なくともサービス名と設定ファイルは指定する必要がある。

サービス名はアルファベットと数字からなる任意の名前がつけられるが、スラッシュや¥、およびダブルクォートなどの記号は使えない。

また、jvmservice の動きを理解するまでは、設定ファイルやログファイルなどのパスは 絶対パスで記述する ことを推奨する。

2.5 起動パラメータ定義ファイル

jvmservice.exe のサービス登録時には、JAVA のクラスや起動パラメータといったものは直接は指定しない。

かわりに、そういった JavaVM 起動時のパラメータや、実行するクラスおよびメソッド名、また、メソッドに引き渡すパラメータといった設定を、記述している「起動パラメータ定義ファイル」を指定する。

設定ファイルはシンプルなテキストファイル形式であり、指定する項目ごとに 1 行づつ指定する。

それぞれの項目 (1 行) は、以下のいずれかとなる。

1. コメント行
2. JavaVM の起動パラメータ
3. 実行するクラスメソッドの指定

空行であるか、すべてが空白であるか、最初の空白以外の文字が# (シャープ) で始まる場合はコメント行とする。

最初の空白以外の文字が- (ハイフン) で始まる場合は、JavaVM への起動パラメータとする。

最初の空白以外の文字が- (ハイフン) 以外であれば、実行するクラスとメソッド、およびパラメータとする。

なお、コメント行以外は環境変数の展開が行われる。

%〜% で囲まれた文字列を変数名として環境変数が存在すれば展開する。変数名として存在しない場合は変換されずに、そのまま残される。[*1]

この動作は WindowsNT のコマンドプロンプトと同じものである。

[*1] 実際にはサービスプロセスの環境であるためシステム環境変数だけが認識されると考えてよい。

2.5.1 サンプル用起動パラメータ定義ファイル

まず **SampleJVMService** を実行するために必要な起動パラメータ定義ファイルについて先に示す。

これはサンプルプログラムに含まれる「**SampleJVMService.conf**」という名前のファイルの内容である。

```
-Xrs
-Djava.class.path=C:/temp/sample
-verbose:jni

SampleJVMService C:/temp/test.txt
```

以下、この内容をもとに「起動パラメータ定義ファイル」に指定する項目について説明する。

2.5.2 JavaVM への起動パラメータ

- (ハイフン) ではじまるパラメータは JavaVM へ送られるパラメータである。

JavaVM への起動パラメータは **jvmservice** にとって、非常に重要である。

以下のパラメータは必須といってよい。

- -Xrs
- -Djava.class.path=クラスパス

`-Xrs` は JDK/JRE1.3 より導入されたシャットダウン監視スレッドを無効にするためのオプションである。

このオプションが指定されていない場合、ログオフなどのアクションを検知すると JavaVM はシャットダウンを開始してしまう。

サービスプロセスとしてはシャットダウン要求は Windows のサービスマネージャから要求されるべきもののため、JavaVM が勝手にシャットダウンさせないように、`-Xrs` を忘れずに指定する必要がある。

なお、`-X` オプションはベンダー固有であり、シャットダウン監視スレッドも Sun Microsystems 以外の JavaVM では異なる可能性がある。

`-Djava.class.path=` はクラスパスの指定である。

実際に SampleJVMService.class が配置されているディレクトリを指定する必要がある。

`-D` はシステム・プロパティとして登録することを指示している。

`java.exe` から起動する場合には、これらのシステム・プロパティは設定済みになっているが、**jvmservice** では、JNI から JavaVM を起動するパラメータとして、このように指示する必要がある。

そのほかによく使われそうなパラメータとしては、以下のようなものがある。

- -Xmx メモリサイズ m
- -verbose:[gc/jni]

いずれも JavaVM への起動パラメータである。

-Xmx は java.exe の起動でも馴染み深いであろう、JavaVM が確保する最大メモリサイズを指定するパラメータである。
他にも-X で始まる **java.exe** が受理するような JavaVM を制御するパラメータは、凡そ受理できると考えてよい。

-verbose は JavaVM からの診断メッセージを出力することを指定する。
gc ではガベージコレクタの動作、jni ではネイティブメソッドの呼び出しなどをトレースすることができるようになる。とくに jni でトレースすると JavaVM がシャットダウンする様子も確認できる。
ただし、**java.exe** の場合とは異なり、**jvmservice.exe** の場合では-verbose による JavaVM からの診断メッセージは、サービス登録時に指定したログファイルへと書き込まれる。ログファイルを指定しない場合は-verbose を指定しても実際には何処にも出力されないことになる。

-verbose を指定せず、ログファイルだけを指定した場合は JavaVM からの診断メッセージは出力されないが、**jvmservice.exe** からのメッセージは記録される。

jvmservice の動き理解できるようになるまでは、とりあえず、ログファイルを指定してサービスを登録し、JavaVM の起動パラメータとして -verbose:jni を指定し JavaVM からのログを見ることを推奨する。

2.5.3 実行するクラスの指定

実行するクラスの指定は、以下のような書式で指定する。
実行するクラスは複数あってもよく、また並列実行の有無を指定することができる。

[&] クラスの完全修飾名 [#クラスメソッド名] [引数 1 [引数 2 [...]]]

それぞれの意味は以下のとおりである。

- & はフォーク (並列実行) の有無
- クラス名はパッケージ名からはじまる完全な名前
- 文字列配列を 1 つ受理する static メソッドの名前
- 文字列引数 (複数可)

クラス名はパッケージ名からはじまる完全な名前でなくてはならない。

また、当然のことならがらクラスパス上に存在しなければならない。

クラス名には# (シャープ) で始まる、メソッド名を指定することができる。

メソッド名を省略した場合は「**main**」が指定されたものとみなされる。

メソッドはクラスで定義されている public 且つ static なメソッドであり、1 つの文字列配列を受け取り、戻り値を返さないメソッドである必要がある。

つまり、以下のシグネチャをもつている必要がある。

```
public static void メソッド名 ( String[] args ) throws Throwable {}
```

メソッドは例外を返しても良い。

例外が返された場合は、そのメソッドがフォークされている場合は単にログに記録される。

フォークされていない場合に例外が返された場合、以降のクラスメソッドの起動は中止される。

いずれの場合も実行中のユーザスレッドがなくなると **jvmservice** はサービスプロセスを終了する。

& (アンパサンド) は指定されたクラスメソッドの実行を待機しないことを示す。[*2]

もし、& (アンパサンド) が指定されている場合、そのクラスメソッドの実行のためのスレッドが作られメソッドが実行されたあと、その完了を待たずに次のクラスの起動が行われる。

この& (アンパサンド) が指定されていない場合、指定されたクラスメソッドの実行が完了するのを待って、次のクラスの起動を行う。ただし、jvmservice は、& (アンパサンド) の有無にかかわりなく、クラスメソッドの実行には、必ず 1 つの独立したスレッドを作成し実行される。つまり、& (アンパサンド) 指定無しによるでは起動したスレッドの完了をただちに join する形となる。

クラスメソッドには 0 個以上の文字列からなる文字列配列が渡される。

引数は、それぞれ空白で区切られる。

引数の中に空白を含む場合は、引数の前後を" (ダブルクォーテーション) で囲む。ダブルクォーテーション自身を含めたい場合は、ダブルクォーテーションで囲まれた中で、"" のようにそれを 2 つ重ねる。

[*2] 指定する場所は行頭であり、行末ではない ことに注意すること。

2.6 サービスの登録

jvmservice.exe のコマンド書式と、**起動パラメータ定義ファイル**の説明が終わったので、ようやくサービスを登録する準備ができた。

ここでは前述の例のとおり、C:/temp/sample/ フォルダ上に SampleJVMService.class があるものとし、テスト用サービス「**jvmserv1**」を登録する。

```
jvmservice.exe --regist jvmserv1
    --config C:/temp/sample/SampleJVMService.conf
    --log C:/temp/sample/SampleJVMService.log
    --dependencies Eventlog ↵
```

--dependencies として Eventlog が指定されている。

jvmservice.exe はスレッドで発生した例外などをイベントログにも記録する。したがって、jvmservice.exe は Eventlog サービスに依存している、といえる。

この指定がデフォルトではないのは「**Eventlog**」という名前が果たして常に同じであるのか不明であるということと、Eventlog サービスに依存する別のサービスに依存する場合がありうるためである。

必要であれば他のサービスを追加する。サービス名はサービス・コントロールパネルを開くと確認できる。名前の大文字小文字は区別されないので、EventLog と eventlog は同じことである。

起動パラメータ定義ファイルやログファイル、依存関係などを変更したい場合は、再度登録を実行する。すでに同じ名前のサービスがあり、その実行ファイルが登録を要求した jvmservice.exe と同じパスであれば、そのサービスに対する更新となる。

うまく登録されると、コントロールパネルの「サービス」で、新しく「**JavaVMService(jvmserv1)**」というサービス名が追加されているはずである。サービス名は「jvmserv1」であるが、表示名としては必ず「JavaVMService(サービス名)」という形式で表示される。

jvmservice.exe は異なる「起動パラメータ定義ファイル」にサービス名をつけて複数個のサービスを登録可能である。

サービスの登録直後、サービスの起動方法は「手動」に設定されている。

ただし、まだ「開始」ボタンをおしてはいけない。

起動すべき JavaVM が指定されていないため、おそらく、起動に失敗するはずである。[*3]

[*3] システム環境変数 PATH に jvm.dll へのパスが通されている場合は起動する。

2.7 ライブラリ定義ファイルの編集

jvmservice.exe と同じディレクトリ上に **jvmservice.preload** というファイルがある。

このファイルは jvmservice.exe がサービスとして起動されるとき読み込まれるダイナミックリンクライブラリ (DLL) を指定している。

2.7.1 ライブラリ定義ファイルの構造

ライブラリ定義ファイルは、必ず、起動された **jvmservice.exe** の拡張子 を ***.preload** に変更したファイル名 となっている。つまり、jvmservice.exe に対して常に 1 つの関係である。

ライブラリ定義ファイルは単純なテキストファイルであり、空行および最初の文字が #(シャープ) ではじまる行をコメント行とし、それ以外の行は、すべて DLL ファイルへのフルパスが書かれているものとする。

したがって、複数個の DLL を読み込みことが可能である。

定義する順番は依存度のもっとも末端[*4]の DLL から順に指定し、この順番で DLL のロードが行われる。

ファイルへのパスは、まず **%〜%** で囲まれた文字列を環境変数名として環境変数展開[*5]が行われてから LoadLibrary API によって読み込まれる。

読み込みに失敗するとサービスプロセスは起動せずに異常終了する。

2.7.2 jvm.dll の指定

jvmservice.exe は JNI により JavaVM を起動する際に、「**jvm.dll**」というダイナミックリンクライブラリのロードを要求する。

この **jvm.dll** こそが JavaVM の本体である。

もし、ライブラリ定義ファイル中に jvm.dll へのパスが指定されていない場合、jvmservice.exe はデフォルトのライブラリ検索パス[*6]を用いて「**jvm.dll**」をロードしようとする。

その結果、jvm.dll がロードできなければサービスプロセスは異常終了する。

しかし、通常、この **jvm.dll** は環境変数 PATH に通されておらず、また通すべきものでもない。

たとえば、1 つのバージョンの JRE でも、JRE1.4.2_08 を例にすれば以下の 2 つの JavaVM(jvm.dll) が用意されている。

- `%JAVA_HOME%`/jre/bin/client/jvm.dll
- `%JAVA_HOME%`/jre/bin/server/jvm.dll

[*4] つまり Windows の Kernel32.dll などの標準 DLL だけに依存する DLL

[*5] 繰り返しになるが、サービスプロセスにおける環境変数はシステム環境変数と考えてよい。環境変数が存在しない場合は変換されずに残される。

[*6] 簡単にいえば環境変数 PATH の経路で探す。

そこで、jvmservice.exe でも、ロードすべき jvm.dll を指定するために「ライブラリ定義ファイル」に、ロードすべき jvm.dll へのフルパスを記述する必要がある。

以下は、C:/j2sdk1.4.2_08/ に JDK1.4.2_08 がインストールされていると仮定した場合の「ライブラリ定義ファイル」の例である。

```
C:\j2sdk1.4.2_08\jre\bin\server\jvm.dll
#C:\jdk1.3.1_15\jre\bin\hotspot\jvm.dll
#C:\jdk1.5.0_02\jre\bin\client\jvm.dll
```

この例では、いくつかの JRE のバージョンの DLL がかかれているがコメントアウトされており、実際にロードするように指定されているのは、j2sdk1.4.2_08 の server バージョンの jvm.dll である。

2.7.3 異なるバージョンの JavaVM の指定

「**ライブラリ定義ファイル (jvmservice.preload)**」は 1 つしか存在しないため、異なるバージョンの JavaVM を指定することはできない。

もし必要であれば jvmservice.exe のファイル名を変更することで切り分けることは可能である。

たとえば「**jvmservice_jre131.exe**」のようにリネームすれば、それに対応する「**jvmservice_jre131.preload**」というライブラリ定義ファイルが使われるようになる。[*7]

[*7] リネームすることが重要である。異なるフォルダにコピーするだけでは不十分である。

2.8 サービスの実行

サービスが登録され、ライブラリ定義ファイルも設定したら、コントロールパネルのサービスから登録したサービスを選択し、「開始」ボタンを押してサービスが起動することを試すことができる。

2.8.1 サービスの開始

「開始」ボタンを押すと、jvmservice.exe の起動が開始される。エラーダイアログが出ずに「開始」ボタンが選択不可能になり「停止」ボタンが選択可能になれば、サービスプロセスは実行中となっている。

正しく動作しているか判断するために、さきほどの java.exe でのテストで行ったように、C:/temp/test.txt ファイルに数字が書き込まれつづけているか確認する。

また、C:/temp/sample/SampleJVMService.log ファイルにログを出力するように指定しているため、ここに JavaVM からの JNI 関連の診断メッセージが出力されていることを確認できる。

2.8.2 サービスの終了

「停止」ボタンを押すと、しばらくして「停止」ボタンが選択不可能になり「開始」ボタンが選択可能な状態になる。この状態であればサービスプロセスは既に停止している。

ただしく停止されたことを確認するために C:/temp/test.txt ファイルを確認する。

ファイルの末尾に「シャットダウン要求完了」とあれば、正しくシャットダウンされていることがわかる。

同様に C:/temp/sample/SampleJVMService.log ファイルを確認し、最終行に「エンドマーク」がつけられていれば、jvmservice.exe 自身も正常なルートで exit したことがわかる。

2.8.3 ログオフとシャットダウンのテスト

サービスプロセスはログオフしても終了しないが、JavaVM では`-Xrs` オプションを指定しない場合はログオフを検知して自動的にシャットダウンしてしまう。

そうならないことを確認するためにサービスを開始したあと、ログオフしてみると良い。

再度ログオンしてサービスが継続中で C:/temp/test.txt ファイルに黙々と数値を書き込んでいるのであれば問題ない。

つぎに、サービスが動作したままシャットダウンしてみる。

再起動後にログファイルを調べて正常に終了していることを確認する。[*8]

[*8] イベントログには終了のイベントが書き込まれない。これはイベントログが先に閉じられてしまうためである。正しく終了したかどうかはログファイルから確認する必要がある。

2.9 アンインストール

登録したサービスを解除する方法、および、jvmservice.exe を削除する方法について説明する。

2.9.1 サービスの登録解除

サービスの登録解除は、jvmservice.exe を`--unregist` オプションを指定して起動することで行う。当然、サービスはあらかじめ停止しておかなければならない。

コマンドラインには以下のような形式で指定する。

```
jvmservice.exe --unregist サービス名
```

何もエラーダイアログがでなければ、指定したサービスが登録解除されている。

ただし、登録解除は削除予約であり、実際にサービスの登録がレジストリから削除されるのは次回の再起動後である。

削除予約したあとで同じサービス名でサービスを登録しようとすると通常はエラーとなる。起動パラメータ定義ファイルやログファイルなどの設定内容を変えたい場合は、削除せずに登録を再実行することで行う。

再登録および登録解除するためには、サービスを登録したときの jvmservice.exe のフルパスと現在の jvmservice.exe のフルパスが一致しなければならない。

異なるパスからの再登録、登録解除は不正な操作としてエラーとなる。[*9]

2.9.2 jvmservice.exe の削除

jvmservice.exe は 1 つ以上のサービスが登録されている場合、レジストリに対してサービスの起動モジュール、およびイベントログのメッセージリソースとして登録されている。

すべてのサービスを登録解除すれば、jvmservice.exe を削除しても安全である。[*10]

その場合、イベントログに記録されている現在の jvmservice.exe が出力したログ内容についてはメッセージが読み取れない状態になる。[*11]

[*9] サービス名は自由につけられるため、既存のシステム用のサービス名を誤って指定する場合もありうる。このような誤操作を防ぐため、サービスとして登録されている実行ファイルと完全に同じパスからでなければ再登録・削除を許可しない。

[*10] サービスとして登録解除せずに jvmservice.exe をファイルシステムから消す誤操作について保護する方法はない。

[*11] これは、Windows のイベントログの形式では、ログファイル上にはメッセージではなくデータが格納されており、このデータを jvmservice.exe のリソース中のメッセージテンプレートを通すことで人間が読めるメッセージに変換するためである。

3 マニュアル

まだ記述がありません。

3.1 jvmservice 起動オプション

まだ記述がありません。

3.2 起動パラメータ定義ファイル

まだ記述がありません。

3.3 ライブラリ定義ファイル

まだ記述がありません。

3.4 ログファイルの内容

まだ記述がありません。

3.5 イベントビューワの内容

まだ記述がありません。

4　アーキテクチャ

まだ記述がありません。