Hewlett Packard
Enterprise

JDK8 リリースノート (翻訳版)

Java™ SE Development Kit (JDK) 8.0
for the OpenVMS Integrity servers Operating System
for the Java™ Platform

目次

はじめに

The Java™ SE Development Kit (JDK) 8.0 for the OpenVMS Integrity servers Operating System for the Java™ Platform （以下では「JDK」といいます）をご利用いただき、ありがとうございます。 このリリース・ノートには、OpenVMS Integrityサーバー版 Oracle® Java™ Platform における、このリリース特有のインストール手順、新機能、既知の問題、その他の情報が記載されています。

このキットを使用すると、OpenVMS バージョン8.4以降の Integrityサーバー・システム上で、 Java™ アプレットおよびプログラムを開発および実行できます。 つまり、 HPE OpenVMS V8.4 for HPE Integrityサーバーシステムと VMS Software Inc. (VSI) OpenVMS のV8.4-1H1以上のシステムです。

JDKキットには、サーバー環境で動作するアプリケーションのプログラム実行速度を 最大限に高めるように設計された HotSpot仮想マシンが含まれています。

Java環境をセットアップするには、次のコマンドを使用します。

```
$ @SYS$MANAGER:JAVA$80_SETUP.COM
```

JDK 8.0 の新機能

Java SE 8は、多数の新機能と拡張を含む主要機能リリースです。Java SE 8のOracle実装であるJDK 8での新機能の概要は、次のOracle社のサイトを参照してください。

» JDK 8の新機能
» What's New in JDK 8

次のような注目すべき新しい言語機能とAPIを含んでいます。

- ラムダ式
- メソッド参照
- デフォルト・メソッド (ディフェンダーメソッド)
- 新しい Stream API
- java.util.Optionalの導入
- 新しい Date/Time API
- Nashorn JavaScriptエンジン
- PermGenの廃止

修正された問題

これは新製品リリースです。ただし、以前のバージョンで実装されたプラットフォーム固有の修正は、 この新しいリリースに適用されます。

JDK 8 の各アップデートリリースごとにOracleが行ったバグ修正の詳細については、 Oracle社の"Java Development Kit 8 Update Release Notes" を確認してください。

» Java 8 リリースのハイライト
» JDK 8 Update Release Notes

互換性

JDK 8.0 は以前のJDKバージョンと高い互換性があり、ほとんどの既存のJava プログラムは 8.0プラットフォームでそのまま動作します。ただし、一部に非互換性が存在し、 これについてはOracle社の"Compatibility Guide for JDK 8"で説明されています。

» JDK 8の互換性ガイド
» Compatibility Guide for JDK 8

Java for OpenVMS の以前のバージョンとの互換性

以下に挙げる項目は Java 6 for OpenVMS と Java 8 for OpenVMS の違いの中で、プログラムの動作に影響を与える可能性があるものです。

- 64ビットポインターのみ使用
  Java6では、HotSpot Java Virtual Machine (JVM) が64ビットポインターを利用して2GBを超えるメモリーの使用を可能にしました。しかし、ランチャーやJavaクラスライブラリから呼び出される共有イメージなど、他のバイナリコンポーネントは、32ビットポインターのみを使用していました。Java8 for OpenVMSでは、64ビットポインターのみを使用します。このため、Javaネイティブインタフェース（JNI）を使用するCまたはC++アプリケーションコードは、64ビットポインターを使用するために (`/POINTER_SIZE=64` で) 再コンパイルする必要があります。アプリケーションコードの性質によっては、コードの変更が必要になることがあります。

- Symbolベクトルの互換性
  前項目に関連しますが、Java8 for OpenVMSに同梱されている共有イメージのシンボル・ベクトルは、Java6のイメージのものと必ずしも一致しません。これらの共有イメージとリンクするJavaネイティブインタフェース（JNI）を使用するCまたはC++アプリケーションコードはすべて再リンクする必要があります。

- 論理名 `JAVA$ENABLE_ENVIRONMENT_EXPANSION` の廃止
  Javaプログラムの実行コマンドは非常に長くなる可能性があり、DCLコマンド行の最大長が問題になることがあります。論理名 `JAVA$ENABLE_ENVIRONMENT_EXPANSION` は、Java for OpenVMSの以前のバージョンで、この問題を回避するために使用されました。定義された場合、Javaコマンドラインで使用したい引数のリストを論理名に定義することができ、"`$`" で始まる引数は（最初の"`$`"を除いて）その論理名として扱われ、Java内部で置き換えられます。
  これによりコマンドラインを短くでき、問題を回避できます。 この機能は、最も一般的には（–cp または –classpath コマンドラインオプションによる）非常に長くなることがある
  Javaクラスパスの指定に使用されています。 しかし他の目的にはほとんど使用されていませんでした。
  Java 8 for OpenVMSでは、以前の論理名 `JAVA$ENABLE_ENVIRONMENT_EXPANSION` が定義されていた場合のように、"`$`"で始まるかどうかにかかわらず、常に `-cp` または `-classpath` に指定される値を確認し、もし論理名であれば内部で置き換えられます。
  また、Java 8では、クラスパス指定でワイルドカード（"`*`"）の使用がサポートされています。この機能もクラスパスの長さを短縮するために使用できます。

- 論理名 `JAVA$FILENAME_CONTROLS` のデフォルト値が "8"
  論理名 `JAVA$FILENAME_CONTROLS` は、Javaが（UNIXおよびOpenVMS形式の間で）ファイル名を解釈およびマップする方法を制御するために使用します。この論理名のデフォルト値は、一般に最大の柔軟性と予測通りの結果をもたらす "8" になりました。
  環境に合わせて `JAVA$FILENAME_CONTROLS` を適切に定義してください。特に `.jar` ファイルや .class ファイルにODS-2ファイルシステムが使用されている場合は注意してください。特定のファイル名のマッピング要件を満たすために変数JAVA$M_MULTI_DOT_KEEP_LAST を設定するには、 JAVA$FILENAME_CONTROLS.COM （デフォルトインストールなら SYS$COMMON:[java$80.com] にあります）の例を参照してください。

- 論理名 JAVA$FORK_PIPE_STYLE の変更
  Java 6 では、この論理名に 0,1,2 の値を指定して、親プロセスと子プロセスの間のパイプの確立方法を制御することができました。値が 2 の場合、メールボックスまたは標準UNIXスタ
  イルのパイプの代わりに、ソケットが使用されます。もし JAVA$FORK_PIPE_STYLE が定義されていない場合、デフォルト値の 1 （プロセス間通信にメールボックスを使います）が使用されます。
  この機能はJava 8 on OpenVMSでも同様ですが、ただし、値 2 はサポートされなくなりました。値 2 または無効な値が指定された場合には、デフォルト値の 1 が自動的に使用されます。

- イメージのデバッグバージョンはありません
  デバッグバージョンのビルドができないほど HotSpot Java 仮想マシンのサイズが大きくなったので、Java 8 for OpenVMSでは、実行プログラムと共有イメージのデバッグバージョンを提供していません。

- ファイル名の大文字と小文字の区別
  Java 8 for OpenVMSのこのリリースでは、ファイル名の大文字小文字の区別が厳しく、
  .java と .class のファイル名は、そのクラスの名前と完全一致する必要があります。たとえば、myClassというJavaクラスがある場合、対応するソースファイルの名前はmyClass.java
  にする必要があります。 これは、JVM（javaコマンド）と javac コンパイラなどのユーティリティの両方に影響します。しかし、クラスのコンパイル時に、 javac にJavaソースコード
  ファイル名の大文字小文字を任意に指定することは可能で、するとコンパイラは実際のディスク上にある（パブリッククラス名と一致していると想定される）ファイル名を選択して使用しようとします。

- 混合構文ファイル名
  以前のバージョンのJava for OpenVMSでは、混合構文ファイル名（UNIX形式とOpenVMS形式のファイルパス構文の組み合わせを含むファイル名）を使用することができていました。このような使用はJava 8 for OpenVMSではサポートされていません。通常は、理想的にはファイル名はUNIX形式の構文になっているべきです。たとえば、次のコードでは例外が発生します。
  ```
  File file = new File (" [.log]/filetest.log");
  ```

- java.awt.headless システムプロパティ
  Java 8 for OpenVMSのこのリリースでは、システムプロパティ java.awt.headless のデフォルト値は"true"に設定されています。 AWT GUIコンポーネントを使用するJavaアプリケーションでは、javaコマンドライン("-Djava.awt.headless=false")またはプログラム中で java.awt.headless を明示的に"false"に設定する必要があります。
  具体的な例：Archive Backup System（ABS）のGUIを使用する場合、以下に示すように起動スクリプト SYS$COMMON:[MDMS.SYSTEM]MDMS$START_GUI.COM のjavaコマンドラインが
  -Djava.awt.headless=false を含むように変更する必要があります。
  ```
  $ java "-Xmx64M" "-Djava.awt.headless=false" "absview.ABSView"
  ```

- 論理名 DECC$READDIR_DROPDOTNOTYPE はJava 8のセットアップで定義されます
  この論理名は、OpenVMS C RTLが拡張子（ファイル・タイプ）なしのファイル名をどのように扱うかを制御します。この論理名が定義されていない場合、拡張子のないファイルを含むデ ィレクトリをjarファイルに追加するような操作を実行すると、拡張子のないファイル名の最後に"."が追加されてjarファイルに保存されます。 このため、Javaコードがjarファイル内のこれらのファイルにアクセスすると問題が発生します。 ディレクトリをスキャンしてファイル名のリストを返すときに、拡張子のないファイル名の最後に"."が追加されるのは、C RTL の通常の
  動作です。 これは、DECC$READDIR_DROPDOTNOTYPEの定義により変更できます。

- 終了状態
  java、javac、およびその他の実行可能なユーティリティーは、通常の正常終了時には、"%X00000001"という状態コードで終了します。この動作は、以前のOpenVMSのJavaのリリースとは異なります。

- エラーログの場所
  クラッシュ（回復不能なエラー状態）が発生した場合、JVMはクラッシュに関する有用な情報を含むログファイルを作成しようとします。 以前のOpenVMSのJavaのバージョンでは、これらのファイルをUNIX/Linuxのテンポラリディレクトリに対応するディレクトリに作成していました。これは、特に定義されていない限り、OpenVMS C RTL によってSYS$SCRATCHにマ ッピングされます。あいまいさを避けるため、このリリースではテンポラリディレクトリではなく、明示的にSYS$SCRATCHを使用しています。

- HPE Secure Web Browser の互換性
  Java 8 for OpenVMS は、現在 HPE Secure Web Browser for OpenVMS と互換性がありません。互換ブラウザプラグインが今後提供される予定です。

- Availability Manager Analyzer の互換性
  Availability Manager Analyzerキットには、互換性のあるJRE（Java Runtime Environment）が含まれています。現在、Java 8 を使用すると Availability Manager Analyzer は正しく動作しません。キットに含まれるJREを上書きしたりバイパスしたりしないでください。 今後、Java 8 で使用できるように更新された Availability Manager Analyzerが利用可能になる予定です。

- SCTPサポート
  Java 8 for OpenVMS は、SCTP（Stream Control Transmission Protocol）をサポートしています。SCTPを使用する場合は、SCTPを有効にしなければいけないことに注意してください。HPE TCP/IP Services の場合、SCTPはデフォルトでは有効になりません。次のように明示的に有効にする必要があります。
  ```
  $ tcpip
  TCPIP> sysconfig "-c" "sctp"
  TCPIP> exit
  ```

- 論理名 `JAVA$DAEMONIZE_MAIN_THREAD` の廃止
  以前のリリースではこの論理名を定義して、メインJVMスレッドを「デーモン化」し、メインスレッド上で実行されるさまざまなタイプの割り込み（特にAST）の影響を減らしていました。これは Java 8 ではデフォルト動作です。 つまり、論理名
  `JAVA$DAEMONIZE_MAIN_THREAD` は設定しても意味はなく、JVMの動作にも影響ありません。

インストール

必須要件

このキットを実行するための必須要件は以下の通りです：

- JDK 8 は ODS-5 形式のディスクにインストールしてください

- OpenVMS Integrity サーバー V8.4

- OpenVMS 国際化データキット (VMSI18N キット) のインストール
  Java debugger (`jdb`) やユーティリティのために必要

- TCP/IP Services for OpenVMS Integrity サーバー V5.7 以上で、最新パッチを含むもの (テストと検証は TCP/IP Services for OpenVMS を使用して実行されましたが、C RTL のソケット関数を使っているだけなので、代りに MultiNet TCP/IP for OpenVMS を使用しても、このJDKは動作するはずです)

- DECWindows Motif V1.7 (AWTを使用する場合)

- スレッドマネージャアップコールのカーネルサポートを有効にしてください（イメージフラグまたはシステムパラメータMULTITHREADにより、スレッドマネージャアップコールを無効にしないでください）

上記のように、JDKはODS-5ディスクにインストールします。

1. 次のコマンドでインストールするディスクがODS-5ボリュームであることを確認します。（ここではJDKを2DKB400にインストールしようとしています）

```
$ SHOW DEV $2$DKB400 /FULL

Disk $2$DKB400: (NODE11), device type COMPAQ BD018635C4, is online,
    mounted, file-oriented device, shareable, served to cluster via MSCP
    Server, error logging is enabled.

    .
    .
    .
  Volume Status: ODS-5, subject to mount verification, file high-water marking
      write-back caching enabled.
```

2. Software Download のWebページから `HPE-I64VMS-JAVA80-V0800--1.PCSI_SFX_I64EXE(~255,000 blocks)` をダウンロードして、RUNコマンドで展開します。

```
$ run HPE-I64VMS-JAVA80-V0800--1.PCSI_SFX_I64EXE
UnZipSFX 6.00 of 20 April 2009, by Info-ZIP (http://www.info-zip.org).
  inflating: hpe-i64vms-java80-v0800--1.pcsi$compressed     (~304,000 blocks)
  inflating: hpe-i64vms-java80-v0800--1.pcsi$compressed_esw       (18 blocks)
  inflating: hpe-i64vms-java80-v0800--1.pcsi$compressed_hpc       (12 blocks)
$
```

   重要：上記のファイルを移動する場合、3つを同じディレクトリに置いてください。

3. 次のコマンドを入力して、キットをインストールします。2DKB400は、JDKをインストールするディスクです。

```
$ PRODUCT INSTALL JAVA80 /DEST=$2$DKB400:
```

   インストール時にPRODUCT INSTALLコマンドで指定できる機能については、"POLYCENTER Software Installation Utility User's Guide"を参照してください。

   インストール手順が進むにつれて、次の情報が表示されます。

```
Performing product kit validation of signed kits ...
%PCSI-I-HPCVALPASSED, validation of DKA100:[VMS$COMMON.KITS.JAVA8]hpe-i64vms-jav
a80-v0800--1.pcsi$compressed;1 succeeded

The following product has been selected:
    HPE I64VMS JAVA80 V8.0              Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and for
any products that may be installed to satisfy software dependency requirements.

Configuring HPE I64VMS JAVA80 V8.0: JAVA for OpenVMS Itanium

    © Copyright 2017 Hewlett-Packard Development Company, L.P.

    Hewlett Packard Enterprise

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
```

```
    HPE I64VMS JAVA80 V8.0              DISK$V842L1SYS:[VMS$COMMON.]

  Portion done: 0%...10%...60%...70%...80%...90%...100%

  The following product has been installed:
    HPE I64VMS JAVA80 V8.0              Layered Product

  HPE I64VMS JAVA80 V8.0: JAVA for OpenVMS Itanium

    Post-installation tasks are required.

    To use Java, users must execute the following command:

      $ @SYS$STARTUP:JAVA$80_SETUP.COM

  $
```

インストール後の作業

インストールが完了したら、次のコマンドの出力が以下に示すのと同様になることを確かめ、Javaが正しくインストールされていることを確認できます。 （OSバージョン、インストールディスク、使用可能なメモリー、ロケール設定などに依存して表示に差異が出ます）

```
$ @sys$startup:java$80_setup.com
$ java "-XshowSettings"
VM settings:
    Max. Heap Size (Estimated): 185.00M
    Ergonomics Machine Class: server
    Using VM: Java HotSpot(TM) 64-Bit Server VM

Property settings:
    awt.toolkit = sun.awt.X11.XToolkit
    file.encoding = ISO8859-1
    file.encoding.pkg = sun.io
    file.separator = /
    java.awt.graphicsenv = sun.awt.X11GraphicsEnvironment
    java.awt.headless = true
    java.awt.printerjob = sun.print.PSPrinterJob
    java.class.path = .
    java.class.version = 52.0
    java.endorsed.dirs = /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/lib/endorsed
    java.ext.dirs = /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/lib/ext
    java.home = /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre
    java.io.tmpdir = /SYS$SCRATCH
    java.library.path = /usr/lib
    java.runtime.name = Java(TM) SE Runtime Environment
    java.runtime.version = 1.8.0.03-vms-rc1
    java.specification.name = Java Platform API Specification
    java.specification.vendor = Oracle Corporation
    java.specification.version = 1.8
    java.vendor = Hewlett-Packard Enterprise
    java.vendor.url = https://www.hpe.com/global/java/alpha/
    java.vendor.url.bug = https://www.hpe.com/global/java/alpha/
    java.version = 1.8.0.03-OpenVMS
    java.vm.info = mixed mode
    java.vm.name = Java HotSpot(TM) 64-Bit Server VM
    java.vm.specification.name = Java Virtual Machine Specification
    java.vm.specification.vendor = Oracle Corporation
    java.vm.specification.version = 1.8
    java.vm.vendor = Hewlett-Packard Enterprise
    java.vm.version = 25.51-b02
    line.separator = ¥n
    os.arch = ia64
    os.name = OpenVMS
    os.version = V8.4-2L1
    path.separator = :
    sun.arch.data.model = 64
    sun.boot.class.path = /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/lib/resourc
es.jar
        /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/lib/rt.jar
```

```
        /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/lib/sunrsasign.jar
        /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/lib/jsse.jar
        /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/lib/jce.jar
        /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/lib/charsets.jar
        /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/lib/jfr.jar
        /DISK$V842L1SYS/SYS0/SYSCOMMON/java$80/jre/classes
    sun.boot.library.path =
    sun.cpu.endian = little
    sun.cpu.isalist =
    sun.io.unicode.encoding = UnicodeLittle
    sun.java.launcher = SUN_STANDARD
    sun.jnu.encoding = ISO8859-1
    sun.management.compiler = HotSpot 64-Bit Server Compiler
    sun.os.patch.level = unknown
    user.dir = /DKA100/VMS$COMMON/KITS/JAVA8
    user.home = /SYS$SYSROOT/SYSMGR
    user.language = en
    user.name = SYSTEM
    user.timezone =

Locale settings:
    default locale = English
    default display locale = English
    default format locale = English
    available locales = , ar, ar_AE, ar_BH, ar_DZ, ar_EG, ar_IQ, ar_JO,
        ar_KW, ar_LB, ar_LY, ar_MA, ar_OM, ar_QA, ar_SA, ar_SD,
        ar_SY, ar_TN, ar_YE, be, be_BY, bg, bg_BG, ca,
        ca_ES, cs, cs_CZ, da, da_DK, de, de_AT, de_CH,
        de_DE, de_GR, de_LU, el, el_CY, el_GR, en, en_AU,
        en_CA, en_GB, en_IE, en_IN, en_MT, en_NZ, en_PH, en_SG,
        en_US, en_ZA, es, es_AR, es_BO, es_CL, es_CO, es_CR,
        es_CU, es_DO, es_EC, es_ES, es_GT, es_HN, es_MX, es_NI,
        es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE,
        et, et_EE, fi, fi_FI, fr, fr_BE, fr_CA, fr_CH,
        fr_FR, fr_LU, ga, ga_IE, hi, hi_IN, hr, hr_HR,
        hu, hu_HU, in, in_ID, is, is_IS, it, it_CH,
        it_IT, iw, iw_IL, ja, ja_JP, ja_JP_JP_#u-ca-japanese, ko, ko_KR,
        lt, lt_LT, lv, lv_LV, mk, mk_MK, ms, ms_MY,
        mt, mt_MT, nl, nl_BE, nl_NL, no, no_NO, no_NO_NY,
        pl, pl_PL, pt, pt_BR, pt_PT, ro, ro_RO, ru,
        ru_RU, sk, sk_SK, sl, sl_SI, sq, sq_AL, sr,
        sr_BA, sr_BA_#Latn, sr_CS, sr_ME, sr_ME_#Latn, sr_RS, sr_RS_#Latn, sr__#
Latn,
        sv, sv_SE, th, th_TH, th_TH_TH_#u-nu-thai, tr, tr_TR, uk,
        uk_UA, vi, vi_VN, zh, zh_CN, zh_HK, zh_SG, zh_TW

Usage: java [-options] class [args...]
           (to execute a class)
   or  java [-options] -jar jarfile [args...]
           (to execute a jar file)
where options include:
    -d32          use a 32-bit data model if available
    -d64          use a 64-bit data model if available
    -server       to select the "server" VM
    -client       is a synonym for the "server" VM  [deprecated]
    -hotspot      is a synonym for the "server" VM  [deprecated]
                  The default VM is server,
                  because you are running on a server-class machine.


    -cp
    -classpath
                  A : separated list of directories, JAR archives,
                  and ZIP archives to search for class files.
    -D=
                  set a system property
    -verbose:[class|gc|jni]
                  enable verbose output
    -version      print product version and exit
    -version:
                  require the specified version to run
    -showversion  print product version and continue
```

```
    -jre-restrict-search | -no-jre-restrict-search
              include/exclude user private JREs in the version search
    -? -help      print this help message
    -X            print help on non-standard options
    -ea[:...|:]
    -enableassertions[:...|:]
              enable assertions with specified granularity
    -da[:...|:]
    -disableassertions[:...|:]
              disable assertions with specified granularity
    -esa | -enablesystemassertions
              enable system assertions
    -dsa | -disablesystemassertions
              disable system assertions
    -agentlib:[=]
              load native agent library , e.g. -agentlib:hprof
              see also, -agentlib:jdwp=help and -agentlib:hprof=help
    -agentpath:[=]
              load native agent library by full pathname
    -javaagent:[=]
              load Java programming language agent, see java.lang.instrument
    -splash:
              show splash screen with specified image
  See http://www.oracle.com/technetwork/java/javase/documentation/index.html for m
  ore details.
```

インストールが成功し、Java が正常に機能していれば、JDKを使用して Java ベースのアプリケーションのコンパイルや実行ができます。

JDK の内容

ここでは、システムにインストールされたJDKに含まれるファイルとディレクトリの概要を説明します。

注意： このリリースノートでは、デフォルト位置 SYS$COMMON:[java$80] にJDKをインストールしたと想定しています。もし別の場所にキットをインストールしている場合には、例のデフォルト位置の部分をその場所で差し替えて読んでください。

開発ツール

SYS$COMMON:[java$80.bin]

Javaプログラミング言語で書かれたプログラムの開発、実行、デバッグ、および文書化に役立つプログラムが含まれています。詳細は、Oracleサイトの JDKツールとユーティリティを参照してください。

重要： 『User Guide』の "Interpreting Commands and OpenVMS Operating System Differences" にある表を確認して、OpenVMS Integrityサーバー版 JDK 8.0 でのニュアンスや違いを完全に理解してください。

ランタイム環境 (JRE)

SYS$COMMON:[java$80.jre]

JDKで使用するJava Runtime Environment（JRE）の実装です。JREには、Java仮想マシン（JVM）、クラスライブラリ、およびJavaプログラミング言語で書かれたプログラムの実行をサポ ートするその他のファイルが含まれています。 （**注記：** JDKに含まれるJREはJREキットとは別のものです）

追加ライブラリ

SYS$COMMON:[java$80.lib]

開発ツールで必要となる追加のクラスライブラリとサポートファイルです。

C ヘッダーファイル

SYS$COMMON:[java$80.include]

Javaネイティブインタフェース(JNI)、JVMツールインタフェース、およびJavaプラットフォームのその他の機能を使用して、ネイティブコードプログラミングをサポートするヘッダファイルです。

ソースコード

SYS$COMMON:[java$80]src.zip

Java core APIを構成するすべてのクラスのソースファイルです。（つまり java.*, javax.*, org.omg.*パッケージのソースファイル、ただしcom.sun.* パッケージは除く） このソースコードは、情報提供のみを目的として提供されており、開発者がJavaプログラミング言語を学んで利用するのに役立ちます。これらのファイルには、プラットフォーム固有の実装コードは含まれておらず、クラスライブラリの再構築には使用できません。 ソースファイルを抽出するには、以下のコマンドを使用します。

```
$ jar xvf src.zip
```

core APIソースファイルを変更しないでください。 core APIの機能を拡張するには、core APIクラスのサブクラスを作成します。

JNI サンプルコード

SYS$COMMON:[java$80.examples.jni]

JNIを使用してJavaからCコードを呼び出し、CからJavaを呼び出す（JVMインスタンスを呼び出す）方法を示す簡単なサンプルコードです。

既知の問題

ここでは、JDK 8.0 for OpenVMS Integrity servers に存在する既知の問題と制限事項について説明します。

- 論理名 JAVA$READDIR_CASE_DISABLE の使用：（論理名についてはUser Guideを参照）論理名 JAVA $READDIR_CASE_DISABLE を定義することにより、Javaプログラムのパフォ ーマンスを改善できます。大文字小文字混在のファイル名抽出（ODS-2での "readdir()" のフ ァイル名大文字小文字の自動修正）が不要な場合に、この論理名でその機能をオフにできます。この場合、Javaコンパイラ（javac）がODS-2ファイル名形式で大文字小文字が混在したクラス名を持つJavaプログラムを参照するときに、"cannot find symbol"（シンボルを見つけることができません）というエラーで失敗します。

- `setReceiveBufferSize(int)` または `setSendBufferSize(int)` メソッドを使用して受信または送信のバッファサイズを設定するには、OpenVMSプロセスに SYSPRV, BYPASS, OPER 特権のいずれかが必要となります。この制限は TCPIP Services によるものです。これらの特権がない場合、これらのJavaメソッドは次のように動作します。
  1.要求された受信または送信のバッファサイズが、システムで設定されているデフォルトのバ ッファサイズより大きい場合、これらのメソッドは失敗します。
  2.要求された受信または送信のバッファサイズが、システムで設定されているデフォルトのバ ッファサイズ以下の場合、システムのデフォルトのバッファサイズとなります。

  回避策として、システムのデフォルトバッファサイズの値を変更することができます。
  ```
  $ sysconfig -r inet tcp_revspace=<default_tcp_receive_buffer_size>
  $ sysconfig -r inet tcp_sendspace=<default_tcp_send_buffer_size>
  $ sysconfig -r inet udp_recvspace=<udp_receive_buffer_size>
  $ sysconfig -r inet udp_sendspace=<default_udp_send_buffer_size>
  ```

- OpenVMSプロセスに SYSPRV, BYPASS, OPER 特権のいずれかがない場合、`setBroadcast(boolean)` メソッドの呼び出しは失敗します。

- 起動時にJavaデバッガ(jdb)が "UTF ERROR" で動きません。（他のユーティリティでも同様の問題が発生することがあります）
  Java jdb ユーティリティーは、UTF-8文字変換を実行するのに C RTL iconv 関数を使用しています。この変換のために RTL が必要とするデータベースファイルは、Java 8 JDKをサポートするすべてのOSバージョンにデフォルトでインストールされているわけではありません。この問題を解決するには、システムにVMSI18Nキットがインストールされていることを確認する必要があります。このキットは、OpenVMSメディア・セットでOpenVMSキットとして提供されています。

- DCLコマンド $ set process/case=sensitive の実行後、Javaは正常に動作しません。

- 論理名 DECC$FILENAME_UNIX_ONLY または DECC$DISABLE_TO_VMS_LOGNAME_TRANSLATION のいずれかが定義されていると、 Javaは正常に動作しません。これらの論理名を設定した場合には、Javaはサポートされません。他の DECC$* 論理名（またはこのような論理名の組み合わせ）を使う場合にも、Javaは正常に動作しないことがあります。

- JDK 5.0 で導入された jinfo, jmap, jsadebugd, jstack 等のトラブルシューティング・ツールは、OpenVMS Integrity サーバーではサポートされていません。

- ごくまれに一部のJavaアプリケーションが起動時に次のメッセージを出して動かないことがあります。
  `"%SYSTEM-I-BREAK_APPL, application break instruction fault, break code=0000000000048005, ..."`
  これは2つ以上の相互依存クラスの並列ジャストインタイム遅延コンパイルに関するスレッドの同期問題によるものです。この問題が完全に解決されるまでは、Javaコマンドラインオプション `-XX:CompileThreshold=0` を指定することで回避できます。このオプションでは、ジ ャストインタイムコンパイルが行われるための、Java仮想マシン（JVM）によるメソッド呼び出しの数を指定します。サーバーアプリケーションに対するデフォルト値は10000です。値0を指定すると、遅延は発生せずに即座にコンパイルが開始されます。値0を指定すると、プログラムの起動時間に影響を与えることに注意してください。ただし、その後の実行時パフォーマンスへの影響はありません。

- 致命的なエラーが発生すると、JVMはクラッシュに関する有用な情報を含むログファイルを作成します。 `-XX:ErrorFile` コマンドラインオプションで指定しない限り、このようなログファイルは論理名 `SYS$SCRATCH` （通常はログインディレクトリ）の指定場所に作成されます。ただし、JVMは（UNIXおよびLinuxシステムの標準スクラッチ領域である） `/tmp` にフ ァイルが作成されたと報告します。論理名 tmp が定義されていない場合、OpenVMS C RTLは `/tmp` を `SYS$SCRATCH` ディレクトリにマップします。論理名 tmp が定義されていて、対応するディレクトリが存在する場合、ログファイルはそのディレクトリに作成されます。例えば、次の定義を行った場合、
  `$ define tmp SYS$SYSDEVICE:[LOGS]`
  ログファイルは `SYS$SYSDEVICE:[LOGS]` に作成されます。（ただしディレクトリへのwrite権限が必要です。）

- スプラッシュスクリーンは小さな画像ファイルのみ対応します。大きな画像ファイルの場合、画像は部分的にしか表示されないことがあります。
