# TrID

**第 1.0 版**

**平成 22 年 3 月 31 日**

## 履歴

| 版数 | 改版日 | 内容 | 備考 |
|---|---|---|---|
| 1.0 | 2010/03/31 | （初版） | |

# 目　次

# 別添資料

- 別添資料 1　060_03_識別性能比較

# 1. TrID について

**本資料は、TrID 及びその識別機能を DLL の形で提供する TrID Lib を調査対象とする。バージョンについては、特に記述がない限り、TrID v2.02 および TrID Lib v1.01 を対象とする。**

## 1.1. 種別

TrID[1]はファイルフォーマット識別ツールである。また、TrID Lib[2]は TrID のフォーマット識別機能を提供する DLL である。

## 1.2. ライセンス

- TrID[3]
  - オープンソースではない。 -
  - 個人もしくは非営利目的の利用において無償である。 -
- TrID Lib[4]
  - オープンソースではない。
  - 個人もしくは非営利目的の利用において無償である。
  - フリーウェアアプリケーションに TrID Lib を使用する場合は、そのアプリケーションのドキュメントとウェブページに TrID Lib の開発者名を明記し、TrID Lib のサイトへのリンクを張ることが必要である。
  - 業務などで利用する場合は開発者への連絡が必要である。
  - DLL を配布する際にはライセンスファイルを含まなければならない。
  - 配布されるすべてのファイルがオリジナルの名前を保有する必要がある。
- TrIDNet[1]
  - オープンソースではない。 -
  - 個人もしくは非営利目的の利用において無償である。 -
  - 業務などで利用する場合は開発者への連絡が必要である。 -
- TrIDScan[2]
  - 不明である。 -

## 1.3. 開発・運用主体

（1） 開発者 -

Marco Pontello 氏

（2） 現在の運用者 -

Marco Pontello 氏 -

## 1.4. 概要

（1） 概要

TrID はファイルに含まれるバイナリシグネチャ（ヘッダ情報やマジックナンバー）から、そのファイルフォーマットを識別するためのユーティリティソフトである。各フォーマットの特徴を XML 形式で記述した定義ファイルを使用し、入力したファイルのフォーマットを識別し、結果を出力する。

また、TrID のフォーマット識別機能を他のアプリケーションに組み込むためのライブラリが TrID Lib という DLL として提供されている。

[1] TrID の GUI 版（実行画面は図 3.参照）
[2] 識別可能なフォーマットの種類を増やすためのツール（詳細は 2.1（3）を参照）

その他のツールとして、TrID の GUI 版である TrIDNet[5]（実行画面は図 3 参照）や、TrIDScan[6]（詳細は 2.1（３）を参照）が公開されている。

（２） 開発の経緯・目的

TrID は、随時最新のフォーマットに対応し、識別可能なフォーマットの種類を増やすことができるように開発されている。

## 1.5. 開発状況

（１） 開発進捗状況

- TrID[1]

2003 年 6 月 7 日 -
v1.00 リリース -
2003 年 8 月 13 日 -
v1.23 リリース -
2003 年 11 月 15 日 -
v1.50 リリース -
2003 年 11 月 20 日 -
v1.55 リリース -
2004 年 12 月 22 日 -
v1.56 リリース -
2006 年 6 月 4 日 -
v2.00 リリース -
2007 年 1 月 11 日 -
v2.02 リリース -

- TrID Lib[2]

2008 年 3 月 31 日 -
v1.00 リリース -
2008 年 5 月 22 日 -
v1.01 リリース

（２） 最新の安定バージョン

- TrID

v2.02

- TrID Lib

v1.01

不具合についての情報は明記されていないが、サポート掲示板において情報・質問を受け付けている。定義ファイルの更新情報も掲示板で公開されている。

（３） 最新の開発バージョン -

TrID、TrID Lib ともに不明である。

（４） ロードマップ -

TrID、TrID Lib ともに不明である。

## 1.6. システム要件

TrID、TrID Lib ともに動作環境は明記されていないが、Windows XP 上で動作確認済みである。

# 2. 機能・構成

## 2.1. 機能概要

（1） TrID

TrID は入力されたファイルの情報（バイナリシグネチャ）とあらかじめ用意された定義ファイルの情報とを比較し、識別成功率が高い順に指定した数だけフォーマットを出力する。フォーマットのバージョンも判別しているが、出力結果は拡張子単位である。

動作の流れは以下の通りである。

① 入力されたファイルからバイナリシグネチャを抽出する。
② 抽出した情報と各フォーマットの定義ファイルの情報とを比較して、識別成功率を算出する。
③ 用意された定義ファイルすべてに対して②を行う。
④ ③の結果、識別成功率の高いフォーマットを指定された数だけ表示する。

例として、Word ファイル「lasik_info.doc」を識別した結果を図 1（赤枠内）に示す。出力結果として、「Microsoft Word document」（70.7%）と「(.) Generic OLE2 / Multistream Compound File」（29.3%）が表示され、前者が正しい識別結果である。

```
C:\TrID>trid c:\test\doc\lasik info.doc

TrID/32   File Identifier v2.02   (C) 2003 06 By M.Pontello

Collecting data from file: c:\test\doc\lasik info.doc
Definitions found: 1959
Analyzing...

 70.7% (.DOC) Microsoft Word document (58000/1/5)
 29.3% (.) Generic OLE2 / Multistream Compound File (24000/1)
```

**図 1 出力結果の例[1]**

（2） TrID Lib

TrID Lib は TrID のファイルフォーマット識別機能を利用するための API である TrID Engine[7]を DLL の形で提供するものである。

（3） TrIDScan

TrIDScan はファイルフォーマットの定義ファイルを作成するツールである。入力された特定のフォーマットのテストファイルのバイナリシグネチャから、そのフォーマット特有のパターンを検出し、定義ファイルを作成する。読み込むファイルの数や形式が多いほど定義ファイルの精度は高くなる。

同一フォーマットに複数のバージョンが存在する場合、バージョン毎にテストファイルを用意し、それを TrIDScan に読み込ませて定義ファイルを作成することで、フォーマットだけでなくバージョンの識別も可能になる。

TrIDScan の実行例（Java の class ファイル）を図 2 に示す。実行した結

果、”newtype.trid.xml”という名前の定義ファイルが作成される。

```
D:\Trid>tridscan f:\test\*.class

TrID/32   Scan Module v1.56   (C) 2003 04 By M.Pontello

Checking files...
Found 97 matching files
Header Block Size: 222
Scanning for patterns...
Pattern(s) found: 3
Last Pattern end at offset: 11
Scanning for raw strings...
Raw string(s) found: 64
Pre filtering strings...
Phase 1... 100%
Phase 2... 100%
Erasing substrings... 100%
String(s) found: 3
Writing XML file ...
Finished!
```

**図 2 TrIDScan の実行例（class ファイル）[6]**

XML 形式で作成された定義ファイルを TrID で使用するには、別途公開されている TrIDDefsPack[8]というツールで TRD 形式[3]のファイルに定義ファイルをまとめる必要がある。

[3] TrID Definitions Package。定義ファイルをつなげ圧縮しただけのデータファイル。仕様については公開されていない。拡張子は「.TRD」である。

## 2.2. 設計

### 2.2.1. コンポーネント構成

コンポーネント構成は公開されておらず、ソースの解析も行えないため不明である。

### 2.2.2. 外部インタフェース

（1）TrIDLib

他システムからファイルフォーマット識別機能を利用するには、TrID Lib を組み込む方法がある。TrID Lib では表 1 に示すメソッドが利用できる。

**表 1 TrID Lib で利用できるメソッド**

| | | 機能 |
|---|---|---|
| Analyze () | int | 分析する |
| GetInfo (int infoType, int infoIdx, char *out) | int | 実行結果を指定した形式で返す |
| LoadDefsPack (char *fileName) | int | 定義をロードする |
| SubmitFileA (char *fileName) | int | 識別するファイルを読み込む |

C#や C++などの言語で利用するためのサンプルコードが提供されているが、Java 版は提供されていない。

（2）TrIDNet

図 3 は「OpenOffice.org Writer document」フォーマットのファイルを入力し、識別を実行した画面である。後方画面では「OpenOffice.org Writer document」（75.0%）、「OpenOffice.org Draw document」（23.6%）、「ZIP compressed archive」（1.3%）の 3 種類の識別結果が出力され、前方画面で適合率が最も高く正しい識別結果である「OpenOffice.org Writer document」の詳細が表示されている。

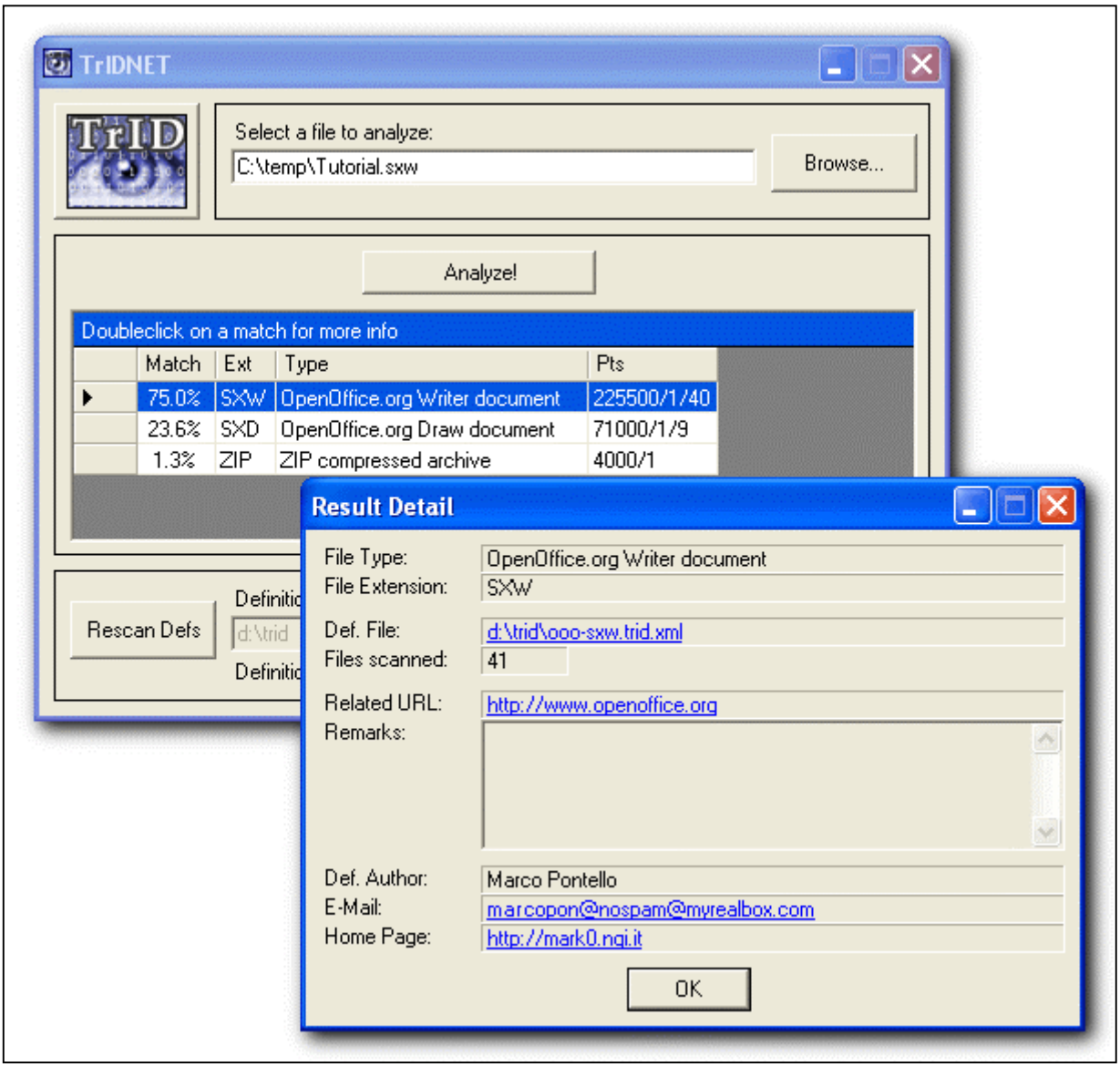


**図 3 TrIDNet の実行画面**[5]

### 2.2.3. データモデル

TrIDScan によって作成される定義ファイルの例（Java の.class ファイル）を以下に示す[6]。

```
<TrID ver="2.00">
    <Info>
        <FileType>Java Bytecode</FileType>
        <Ext>class</Ext>
        <ExtraInfo>
            <Rem></Rem>
            <RefURL></RefURL>
        </ExtraInfo>
        <User>Marco Pontello</User>
        <E-Mail>marcopon-nospam@myrealbox.com</E-Mail>
    </Info>
    <General>
        <FileNum>80</FileNum>
        <CheckStrings>True</CheckStrings>
        <Date>
            <Year>2003</Year>
            <Month>11</Month>
            <Day>14</Day>
        </Date>
        <Time>
            <Hour>03</Hour>
            <Min>10</Min>
            <Sec>51</Sec>
        </Time>
    </General>
    <FrontBlock>
        <Pattern>
            <Bytes>CAFEBABE00</Bytes>
            <Pos>0</Pos>
        </Pattern>
        <Pattern>
            <Bytes>00</Bytes>
            <Pos>6</Pos>
        </Pattern>
        <Pattern>
            <Bytes>00</Bytes>
            <Pos>11</Pos>
        </Pattern>
```

```
    </FrontBlock>
    <GlobalStrings>
        <String>Code</String>
        <String>init</String>
        <String>java</String>
    </GlobalStrings>
</TrID>
```

定義ファイルのタグのうち、公式サイト[6]で説明されているものを表 2 に示す。

**表 2 定義ファイルのタグ**

| | 概要 |
|---|---|
| FileType | ファイルフォーマット |
| Ext | 拡張子 |
| User | 作成者 |
| E-mail | メールアドレス |
| Home | ホームページアドレス |
| Rem | ファイルフォーマットについての情報 |
| RefURL | 参照 URL |

## 2.3. ソースコード

TrID のソースコードは公開されていない。コマンドラインから起動する場合は、TrID のインストールディレクトリに移動して、以下のコマンドを入力する。

trid.exe input.pdf

この場合、input.pdf の識別結果が標準出力に表示される。

## 3．性能

### （1） 識別可能なフォーマット数

2009 年 10 月 26 日現在、サイト上で公開されている定義ファイルは 3,852 種類（同一拡張子に対する異なるバージョンのものを含む）である[9]。定義ファイルは、開発者自身が作成したものと第三者によって提供されたものがあり、随時追加・更新が行われている。

### （2） 実験結果

26 種類のフォーマットに対して 6 つのパターンを作成し、識別テストを行った。6 つのパターンとは、各テストフォーマットに対して、拡張子あり、拡張子なし、拡張子を偽装したものの 3 つのパターンと、これらのパターンに日本語名のみのファイル、英語名のみのファイルを用意したものである。結果は別添資料 1「060_03_識別性能比較」を参照。

識別した結果は、正しく識別された（第 1 位の識別成功率で判定された）場合に「識別可能」、間違ったフォーマットと識別された場合[4]と、識別結果が「Unknown」となった場合に「識別不可」の 2 つの基準を設けている。

識別結果が不可であったフォーマットのうち、「.jhd」、「.jsd」については、適切な定義ファイルが存在せず、さらにマジックナンバーのパターンが「Generic OLE2 / Multistream Compound File」と一致（D0CF11E0A1B11AE1）していることが原因である。「.ods」（Open Document Spreadsheet）については、定義ファイルのヘッダ情報の 4byte の開始パターンが「.odt」（Open Document Text）と同じため、登録されている定義ファイル数がより多い「.odt」と誤判定されたと思われる。「.txt」については、UTF エンコードのテキストフォーマットの定義ファイルが存在しており、UTF-8 及び UTF-16 の文字コードのテキストファイルをテストしたところ、正しく識別することが出来た。ASCII 及び Shift_JIS は定義ファイルが存在していなかったため、識別することができなかった。

識別不可だった各フォーマットに関して、再度テストを行った。当該フォーマットのファイル各 5 個から TrIDScan を用いて正しい定義ファイルを作成し、初回のテストと同じファイルを対象に実施した。その結果、初回と同じ判定結果及び識別成功率となり、新たに定義ファイルを追加しても正しく識別できなかった。

### （3） 定義ファイルと識別性能の関係

TrIDScan を利用してユーザーが独自に定義ファイルを作成した際、識別結果に与える影響を調査した。具体的には、以下の 2 点について検証を行った。

A　識別不可だったフォーマットを用いた検証

TrID に定義ファイルがなく識別できなかったフォーマットについて、ユーザーが TrIDScan を使って独自に定義ファイルを作成した場合、そのフォーマットを識別できるようになるか。また、定義ファイルに追加するファイルの数が増えると、識別成功率は向上するのか。

B　識別可能だったフォーマットを用いた検証

定義ファイルが既に登録されており正しく識別できたフォーマットについて、ユーザーが TrIDScan で誤った定義ファイルを新たに追加した場合、識別できなくなるのか。また、定義ファイルに追加する誤った情報をもつファイルの数が増えると、識別成功率は下降するのか。

[4] 正しい識別結果が 2 位以降で出力された場合も、識別不可としている。

検証のために、3 種類のテストを行った[5]。①は A について、②、③は B についての検証である。いずれのテストにおいても、定義ファイル作成に使用するファイルと識別対象ファイルはすべて異なるファイルである。また、識別テストの対象ファイルは、すべて中身が日本語でファイル名が英語のファイルを使用した。

① - TrID に定義ファイルが登録されていない「花子」フォーマットのファイル（.jhd）に対して、以下の 5 種類の定義ファイル（TRD 形式）を用意し、各定義ファイルを利用した場合の識別結果を比較する。

(ア) 新規作成したデフォルトの定義ファイル

(イ) （ア）と、2 個の「.jhd」ファイルから作成した定義ファイルをマージした新定義ファイル

(ウ) （ア）と、3 個の「.jhd」ファイルから作成した定義ファイルをマージした新定義ファイル

(エ) （ア）と、4 個の「.jhd」ファイルから作成した定義ファイルをマージした新定義ファイル

(オ) （ア）と、5 個の「.jhd」ファイルから作成した定義ファイルをマージした新定義ファイル

② 正しく識別できる（定義ファイルが既に登録されている）「一太郎」フォーマットのファイル（.jtd）に対して、以下の 5 種類の定義ファイル（TRD 形式）を用意し、各定義ファイルを利用した場合の識別結果を比較する。（ア）は正しい定義ファイル、（イ）～（オ）は「一太郎」フォーマットに関する誤った情報を追加した定義ファイルである。

(ア) デフォルトの定義ファイル（定義ファイル：jtd.trid.xml）

(イ) （ア）と、2 個の「一太郎」フォーマットのファイル（拡張子を「.abc」に変更）から作成した定義ファイルをマージした新定義ファイル

(ウ) （ア）と、3 個の「一太郎」フォーマットのファイル（拡張子を「.abc」に変更）から作成した定義ファイルをマージした新定義ファイル

(エ) （ア）と、4 個の「一太郎」フォーマットのファイル（拡張子を「.abc」に変更）から作成した定義ファイルをマージした新定義ファイル

(オ) （ア）と、5 個の「一太郎」フォーマットのファイル（拡張子を「.abc」に変更）から作成した定義ファイルをマージした新定義ファイル

③ 正しく識別できる（定義ファイルが既に登録されている）「Word」フォーマットのファイル（.doc）に対して、以下の 5 種類の定義ファイル（TRD 形式）を用意し、各定義ファイルを利用した場合の識別結果を比較する。（ア）は正しい定義ファイル、（イ）～（オ）は「Word」フォーマットに関する誤った情報を追加した定義ファイルである。また、「Word」フォーマットはバージョン 2000 を使用した。

(ア) デフォルトの定義ファイル（定義ファイル：doc-ms-word.trid.xml）

(イ) （ア）と、2 個の「Word」フォーマットのファイル（拡張子を「.abc」に変更）から作成した定義ファイルをマージした新定義ファイル

(ウ) （ア）と、3 個の「Word」フォーマットのファイル（拡張子を「.abc」に変更）から作成した定義ファイルをマージした新定義ファイル

(エ) （ア）と、4 個の「Word」フォーマットのファイル（拡張子を「.abc」に変更）から作成した定義ファイルをマージした新定義ファイル

[5] 上記（2）のテストで識別不可となった各フォーマット（ods、txt、jsd、wma）についても定義ファイルを作成して①のテストを実施したが、同様に識別できなかった。

(オ) （ア）と、5 個の「Word」フォーマットのファイル（拡張子を「.abc」に変更）から作成した定義ファイルをマージした新定義ファイル

①～③のテスト結果をそれぞれ表 3～5 に示す。

**表 3 ①のテスト結果**

| 入力 | | | 出力 | | 結果 |
|---|---|---|---|---|---|
| | | フォーマット | | | |
| （ア） | test.jhd | JHD | (.) Generic OLE2 / Multistream Compound File (8000/1) | 100.0% | 識別不可 |
| （イ） | test.jhd | JHD | (.) Generic OLE2 / Multistream Compound File (8000/1) | 100.0% | 識別不可 |
| （ウ） | test.jhd | JHD | (.) Generic OLE2 / Multistream Compound File (8000/1) | 100.0% | 識別不可 |
| （エ） | test.jhd | JHD | (.) Generic OLE2 / Multistream Compound File (8000/1) | 100.0% | 識別不可 |
| （オ） | test.jhd | JHD | (.) Generic OLE2 / Multistream Compound File (8000/1) | 100.0% | 識別不可 |

**表 4 ②のテスト結果**

| 入力 | | | 出力 | | 結果 |
|---|---|---|---|---|---|
| | | フォーマット | | | |
| （ア） | test.jtd | JTD | (.JTD) Ichitaro document (151564/3/14) | 94.9% | 識別可能 |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 5.0% | |
| （イ） | test.jtd | JTD | (.ABC) Enter a useful file type description (1044950/21/43) | 86.7% | 識別不可 |
| | | | (.JTD) Ichitaro document (151564/3/14) | 12.5% | |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 0.6% | |
| （ウ） | test.jtd | JTD | (.ABC) Enter a useful file type description (1044950/21/43) | 86.7% | 識別不可 |
| | | | (.JTD) Ichitaro document (151564/3/14) | 12.5% | |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 0.6% | |
| （エ） | test.jtd | JTD | (.ABC) Enter a useful file type description (1044950/21/43) | 86.7% | 識別不可 |
| | | | (.JTD) Ichitaro document (151564/3/14) | 12.5% | |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 0.6% | |
| （オ） | test.jtd | JTD | (.ABC) Enter a useful file type description (1044950/21 | 86.7% | 識別不可 |
| | | | (.JTD) Ichitaro document (151564/3/14) | 12.5% | |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 0.6% | |

**表 5 ③のテスト結果**

| 入力 | | | 出力 | | 結果 |
|---|---|---|---|---|---|
| | | フォーマット | | | |
| （ア） | test.doc | DOC | (.DOC) Microsoft Word document (32000/1/3) | 80.0% | 識別可能 |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 20.0% | |
| （イ） | test.doc | DOC | (.DOC) Microsoft Word document (32000/1/3) | 80.0% | 識別可能 |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 20.0% | |
| （ウ） | test.doc | DOC | (.DOC) Microsoft Word document (32000/1/3) | 80.0% | 識別可能 |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 20.0% | |
| （エ） | test.doc | DOC | (.DOC) Microsoft Word document (32000/1/3) | 80.0% | 識別可能 |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 20.0% | |
| （オ） | test.doc | DOC | (.DOC) Microsoft Word document (32000/1/3) | 80.0% | 識別可能 |
| | | | (.) Generic OLE2 / Multistream Compound File (8000/1) | 20.0% | |

表 3 から、「.jhd」の定義ファイルを追加しても、識別結果に影響を与えておらず、OLE2.0 のコンテナとして誤った判定をしている。識別アルゴリズムが不明であるため原因は分からないが、マジックナンバーが他のフォーマットと同じ場合、定義ファイルを追加しても正しく識別できるとは限らないと言える。

次に表 4 から、定義ファイル（ア）と（イ）を使った場合を比較して、誤った定義ファイルを追加した場合に誤ったフォーマット（.abc）と判定されていることから、判定結果に影響を与えていることが分かる。また、（イ）～（オ）の定義ファイルでは識別結果の識別成功率が変わらないことから、定義ファイル作成に使用するファイル数と識別成功率との間に相関関係はないと言える。

しかし、③のテスト結果である表 5 では、②と同様に誤った定義ファイルを使用したが、判定結果は変わらず、正しく識別できた。この理由として、②、③の（ア）で使用したデフォルトの定義ファイルに使用しているファイル数（「Word」は 570 個、「一太郎」は 26 個）[9]が関係していると推測されるが、識別アルゴリズムが不明であるため、詳細な考察はできない。

テスト実行環境

OS　　Windows XP

## 4. まとめ

TrID は、フォーマットの定義ファイルを作成することで識別可能なフォーマットの対象数を増やすことができる点において、拡張性の高いフォーマット識別ツールである。公開されている定義ファイルの追加・更新も随時行われている。一方、ソースコードは公開されていないため、識別アルゴリズムの変更等を行うことはできず、識別機能を他のアプリケーションに組み込むには、DLL（TrID Lib）を利用するしかない。

識別テストの結果から、定義ファイルが用意されていないフォーマットについては識別できないことが分かる。また、定義ファイルが用意されているものでも、ヘッダ情報の先頭のビットパターンが同一のフォーマットの定義ファイルも登録されている場合、誤って識別することも分かった。固有のヘッダ情報を持つフォーマットについては、正しく識別できている。識別速度も特に問題はない。

しかし、定義ファイルに使用するファイルの内容に識別精度が影響される点や、公式サイトで公開されている定義ファイルに関して、誰でも作成できるため、その作成方法や正当性及び信頼性が明らかでない点などから、識別結果に対する信頼性は高くないと言える。

## 5．参考文献

[1] Marco Pontello's Home – Software – TrID,
http://mark0.net/soft-trid-e.html
（最終アクセス日　2009 年 10 月 26 日） -

[2] Marco Pontello's Home – Code & Libs – TrIDLib,
http://mark0.net/code-tridlib-e.html
（最終アクセス日　2009 年 10 月 26 日） -

[3] trid_w32.zip 内 readme_e.txt
（最終アクセス日　2009 年 10 月 26 日） -

[4] tridlib-free 内 TrIDLib-License.txt
（最終アクセス日　2009 年 10 月 26 日） -

[5] Marco Pontello's Home – Software – TrIDNet,
http://mark0.net/soft-tridnet-e.html
（最終アクセス日　2009 年 10 月 26 日） -

[6] Marco Pontello's Home – Software – TrIDScan,
http://mark0.net/soft-tridscan-e.html
（最終アクセス日　2009 年 10 月 26 日） -

[7] Marco Pontello's Home – Software – TrIDEngine,
http://mark0.net/code-tridengine-e.html
（最終アクセス日　2009 年 10 月 26 日） -

[8] TrIDDefsPack v1.10,
http://mark0.net/download/triddefspack.zip
（最終アクセス日　2009 年 10 月 26 日） -

[9] Marco Pontello's Home – Software – TrID – Files Extensions and File Type Definitions list,
http://mark0.net/soft-trid-deflist.html
（最終アクセス日　2009 年 10 月 26 日）

# 極窓

第 1.0 版

平成 22 年 3 月 31 日

## 履歴

| 版数 | 改版日 | 内容 | 備考 |
|---|---|---|---|
| 1.0 | 2010/03/31 | （初版） | |

# 目　次

# 別添資料

- 別添資料 1　060_03_識別性能比較

# 1. 極窓について

**本資料は 極窓 v25.10、WinExChange v7.06、WINEX32.DLL v1.44 を対象とする。**
**極窓には複数の機能があるが、今回は主に識別機能についてのみ調査対象とする。**

## 1.1. 種別

極窓はファイルフォーマット識別ツールである。

## 1.2. ライセンス

極窓とWinExChangeはフリーソフトであり、アーカイブの内容の変更は禁止されているが、転載は可能である。WINEX32.DLLは、使用条件などは特になく自己責任の範囲で自由に使用可能である。

## 1.3. 開発・運用主体

（1） 開発者

NTSOFT 氏[1]が開発を行っている。NTSOFT 氏は極窓以外にも ZIP 書庫画像ビュアーなどのフリーソフトを複数開発、運用している。

（2） 現在の運用者

NTSOFT 氏。

## 1.4. 概要

（1） 概要

極窓は NTSOFT 氏が公開しているファイルフォーマット識別・変換ツールである。国内では幅広く利用されており、対応するファイルフォーマット数も 2009 年 11 月 24 日現在 1,388 種類（拡張子のみ）[1]である。機能面ではファイルフォーマット識別機能だけでなく、ファイルの圧縮解凍、ファイルのリネーム、ファイルの属性変更、ファイルのタイムスタンプ変更やファイルを簡易的に表示するビュアー機能など多機能なソフトであり、中でもファイルの中身を解析して本来のファイルフォーマットを識別する機能に優れているとされている[2][3]。

しかし、極窓はフリーソフトでありオープンソースではない。そのためソースコードやモデル図などは公開されていないが、極窓のファイルフォーマット識別ルーチンのみを VC++ に移植した「WinExChange」を DLL 化した「WINEX32.DLL」が開発者向けに公開されており、この DLL を使用することで、各種システムがファイルフォーマット識別機能を実装することができる[4]。極窓の各種機能のうち、DLL として提供されているものは、この「WINEX32.DLL」のみである。

（2） 開発の経緯・目的

フォーマットが不明または偽装されているファイルのフォーマットを特定することを目的に作成されている。

[1] 対応している拡張子は拡張子博物館[8]参照

## 1.5. 開発状況

（1） 開発進捗状況

現在も極窓とWinExChange、WINEX32.DLLの開発は続いており、年に数回アップデートがされている。

（2） 調査時点でリリースされている最新の安定バージョンとその情報

- 極窓

2009年3月30日

バージョン25.10（最新の安定版）が公開される

- WinExChange

2009年9月10日

バージョン7.06（最新の安定版）が公開される

- WINEX32.DLL

2009年9月28日

バージョン 1.44（最新の安定版）が公開される

（3） 最新の開発バージョンとその情報

上記（2）と同じ。

（4） 今後のロードマップ

Windows 7 の発売日（2009 年 10 月 22 日）に合わせて、.NET 版の極窓.NET がリリース予定[5]となっているが、2009 年 11 月末現在、まだリリースされていない。

## 1.6. システム要件

（1） 必須環境

極窓、WinExChange、WINEX32.DLL はすべて Windows OS で動作する。また、極窓の動作には、VB6.0（SP5）ランタイムが必要となる。

（2） 推奨環境

特に指定はない。

# 2. 機能・構成

## 2.1. 機能概要

極窓は以下の機能を提供している[2]。

- 2009 年 11 月 24 日現在 1,388 種類、1,585 パターンの拡張子のファイルフォーマットを識別
- 指定したファイルフォーマットに複数ファイルを一括で変換
- 複数ファイルの拡張子を一括して識別変換
- ファイル名の置換機能（正規表現対応）
- ファイル属性の変更機能
- ファイルの分割/指定部分分割/結合機能
- ファイルの暗号化/復号化機能
- ファイルタイムスタンプの変更機能
- ファイル名の連番機能
- ファイル名一覧のテキスト出力/印刷/CSV 出力
- ファイル名の大文字・小文字の変更
- フォルダ連番作成機能
- ダミーファイルの作成機能
- マルチビュアー搭載[2]
- 書庫[3]の解凍[4]
- 書庫の圧縮[5]、自己解凍形式[6]、パスワード形式[7]
- 書庫チェック機能
- 重複ファイルチェック機能
- マックバイナリ形式用の識別機能
- マックバイナリのヘッダ除去
- Windows Media Player に対応したファイル再生機能
- JIS から Shift-JIS への文字コード変換機能
- ファイル整理機能
- ファイルの CRC[8]計算機能
- CRC ファイル作成及びチェック機能

プログラムに組み込むためには「WINEX32.DLL」を使用する。また「WINEX32.DLL」は識別機能のみを提供するもので、他の機能を組み込むことは不可能である。

ファイルフォーマット識別機能は、フォーマットのバージョンの識別までは行わず、拡張子単位で識別する。

識別ロジックについては不明である。

---

[2] 音声、画像、動画、書庫、HTML、スクリーンセーバー、ダンプ、ファイル情報、テキストに対応
[3] 複数ファイルを一つにまとめたファイル。アーカイブ。
[4] JZH,ZIP,JAR,CAB,TAR,TGZ,TAZ,TBZ,GZ,Z,BZ2,ISH,ARJ,MS-Compress,YZ1,YZ2,BZA,GZA,GCA,ACE,NOA,7Z,DGC ファイルに対応
[5] LZH,ZIP,CAB,YZ1,YZ2,GZA,BZA,NOA,7Z ファイルに対応
[6] LHA,ZIP,CAB,7Z ファイルに対応
[7] ZIP,YZ1,YZ2,7Z ファイルに対応
[8] Cyclic Redundancy Check の略。巡回冗長検査。データ転送時のエラーチェック法の一つ。32bit,16bit に対応

図 1 に WinExChange の出力結果画面を示す。ファイル「サンプル.pdf」のフォーマットを PDF と識別して出力した例（赤枠内）である。


**図1 WinExChange の出力結果（例：PDF）**

## 2.2. 設計

### 2.2.1. コンポーネント構成

不明である。

### 2.2.2. 外部インタフェース

（1） 他システムと連携して利用することの可否

他システムと連携することが可能である。

（2） 連携方法

WINEX32.DLL を使用することで、極窓の識別機能を使用できる。

（3） 実際の利用例

以下に WINEX32.DLL を C++で使用し、識別結果を取得・表示する例を記す。WINEX32.DLL の入出力データ構造を表 1 に、また、メソッドを表 2 及び表 3、表 4 に記す[6]。

```
HMODULE hDLL;
tExInfo tData;
int (WINAPI * WinExGetInfo)(LPCSTR, int, tExInfo*, DWORD);

hDLL = ::LoadLibrary("WINEX32.DLL");
if(hDLL == NULL)
{
        ::MessageBox(NULL, "WinEx32.DLL が見つかりません", "エラー", MB_OK);
        return;
}
WinExGetInfo = (int(WINAPI*)(LPCSTR, int, tExInfo*,
DWORD))::GetProcAddress(hDLL, "WinExGetInfo");

WinExGetInfo(argv[1], 3000, &tData, 0);

::MessageBox(NULL, tData.szFileEx, "判別結果", MB_OK);

::FreeLibrary(hDLL);
```

**表 1 WINEX32.DLL の入出力データ構造**

| | | | 登録データ例 |
|---|---|---|---|
| 入力 | File | 解析対象ファイル | gokumado.jpg |
| 出力 | szFileEx | ファイル名 | gokumado.jpg |
| | szExInfo1 | ファイル情報 | Joint Photographic Coding Experts Group File<br>「画像」 |
| | szExInfo2 | ファイル付属情報 | 空欄 |

表 2 WINEX32.DLL のメソッド

| | | 機能 |
|---|---|---|
| WinExGetInfo(LPCSTR pcszFileName, int nBuffLen, tExInfo *data, DWORD dwFlag) | int | 拡張子を判別し、構造体に情報を返す |
| WinExGetVersion() | int | WinEx32.DLL のバージョンを返す |
| WinExGetExNumKind() | int | WinEx32.DLL が対応している拡張子の種類 |
| WinExGetExNumPattern() | int | WinEx32.DLL が対応している拡張子のパターン数 |
| WINAPI WinExGetExtList(LPSTR szExList) | int | 対応している拡張子の一覧を CSV 形式で取得する<br>※受け取るサイズは、5,000Byte 程度必要 |
| WINAPI WinExGetExtListLength() | int | 受け取るバッファサイズを返す [v1.20 以降] |
| WinExGetFileCRC(LPCSTR pcszFileName) | int | 指定したファイルの CRC（16bit）計算の値を返す |
| WinExGetFileCRC32(LPCSTR pcszFileName) | int | 指定したファイルの CRC（32bit）計算の値を返す |

表 3 WinExGetInfo メソッド詳細（戻り値あり）

| | | 機能 |
|---|---|---|
| WinExGetInfo (LPCSTR pcszFileName, int nBuffLen, tExInfo *data, DWORD dwFlag) | int | 拡張子を判別し、構造体に情報を返す |

表 4 WinExGetInfo メソッド詳細（戻り値なし）

| | 概要 |
|---|---|
| pcszFileName | ファイル名「フルパス」 |
| nBuffLen | 調べるバイト数（BUFFSIZE_????）<br>※「????の数値が大きいほど識別精度が上がる」 |
| data | データを受け取る構造体（struct ExInfo）※ |
| dwFlag | 下記の値で識別オプションを指定する<br><br>0 = 標準識別<br>1 = マックバイナリを使用して識別<br>2 = フッターで識別<br>4 = 特殊識別<br>8 = メッセージボックスを出力しない |

※ExInfo 構造体

| | |
|---|---|
| char szFileEx[512] | 判別結果の拡張子（複数の場合は CSV 形式） |
| char szExInfo1[256] | 判別結果の拡張子の簡単な説明 |
| char szExInfo2[512] | 判別結果の拡張子の付属情報（対応していれば） |

### 2.2.3. データモデル

データモデルは公開されていない。入出力データ構造は表 1 の通りである。

## 2.3. ソースコード

ソースコードは開示されていないが、「WINEX32.DLL」で使用できるメソッドについては開示されている（表 2～4 を参照）。

## 3．性能

性能の調査は、組み込みに必要なWINEX32.DLLを対象とする。

（1）識別可能なフォーマット数

WINEX32.DLL の識別可能なフォーマット数は、2009 年 11 月 24 日現在 1388 種類、1,585 パターンの拡張子である[9][7]。日本独自の拡張子にも対応しており、一太郎のファイルはほぼ全てのバージョンに対応済みである[10]。

（2）実験結果

26 種類のフォーマットに対して 6 つのパターンを作成し、識別テスト行った。6 つのパターンとは、各テストフォーマットに対して、拡張子あり、拡張子なし、拡張子を偽装したものの 3 つのパターンと、これらのパターンにファイル名が日本語名のみのファイル、英語名のみのファイルをそれぞれ用意したものである。結果は別添資料 1「060_03_識別性能比較」を参照。

識別した結果は、正しく識別できた場合に「識別可能」、間違ったフォーマットに識別した場合と識別結果が返されなかった場合に「識別不可」の 2 つの基準を設けている。

今回作成するプロトタイプシステムに極窓のフォーマット識別機能を組み込むには WINEX32.DLL が必要である。WINEX32.DLL は極窓のファイルフォーマット識別ルーチンを VC++に移植した WinExChange の DLL であるので、識別精度テストは WinExChange を使用している。

結果として、HTML、XML 形式のファイルの拡張子を「.txt」に偽装した場合、両形式とも極窓で対応済みとされているにも関わらず、プレーンテキスト形式に誤まって識別された。

これは、HTML 及び XML がテキストファイルの一種であるためにプレーンテキストと識別されたと考え、下記の 4 パターンで検証を行った。

①. 正しいフォーマットが XML で、拡張子が「.xml」のファイル
②. 正しいフォーマットがプレーンテキスト（TEXT 形式）で、拡張子が「.txt」のファイル
③. 正しいフォーマットが XML で、拡張子が「.txt」のファイル
④. 正しいフォーマットがプレーンテキスト（TEXT 形式）で、拡張子が「.xml」のファイル

**表 5 検証結果**

| | | | 識別結果 |
|---|---|---|---|
| ① | フォーマット XML<br>拡張子.xml | ExtensibleMarkup Language File「言語」 | 識別可能 |
| ② | フォーマットプレーンテキスト（TEXT形式）<br>拡張子.txt | Text File「文章」 | 識別可能 |
| ③ | フォーマット XML<br>拡張子.txt | Text File「文章」 | 識別不可 |
| ④ | フォーマットプレーンテキスト（TEXT形式）<br>拡張子.xml | 識別せず（空白） | 識別不可 |

以上の結果から、テキストファイルに関しては、（ファイルフォーマット識別機能を持っているが）拡張子情報のみで判定している、または判定の際に拡張子情報がその他のフォー

[9] 拡張子博物館に収録されている拡張子は 12 月 17 日時点では 1341 種類となっている。
[10] 拡張子博物館 -「j」で検索[8]

マット情報（ヘッダ情報等）より優先されている可能性が高いと考えられる。

一方、バイナリファイル（DOC 形式や JPEG 形式等のヘッダ情報があるファイル）については、XLSX 形式及び OpenOffice 形式を除き、拡張子を偽装したファイルや拡張子のないファイルでも識別に成功している。

以上のことから極窓のファイルフォーマット識別機能では、バイナリファイルについてはヘッダ情報を見て識別しており、XML や HTML 形式といったテキストファイルは、ヘッダ情報では判断できないため、拡張子と XML では XML 宣言、HTML 形式では DOCTYPE 宣言といった情報を見て識別していると推察できる。また TEXT 形式を「.xml」に偽装した場合に識別結果が空白で返る原因としては、正しいフォーマットがプレーンテキスト（TEXT 形式）で、拡張子が「.xml」のファイルに XML 形式の要素が見つからず、またプレーンテキストとも判断し切れなかったため、このような結果になったと思われる。

なお、Office 2007 形式に関してはすべて対応済み[8]となっており、DOCX 形式では全パターンで識別可能であったが、XLSX 形式では、拡張子を「.zip」に偽装した場合と拡張子がない場合に識別できなかった。同じ Microsoft の表計算ソフトでも、XLS 形式の拡張子を「.doc」に偽装した場合や拡張子がない場合に正しく識別することが出来たため、Office 2007 形式の全てのフォーマットに対してまだ対応しているわけではないと思われる。

また、OpenOffice にも対応済みとあった[8]が、「.ods」「.odp」「.odt」ファイルをテストしたところ、「Open Publication Structure eBook File（.epub）」[11]と判定され、正しく識別できなかった。従って、拡張子博物館で極窓等に対応済みと記載されていても、必ず識別できるわけではないことが判明した。

日本独自のフォーマット「一太郎（.jtd）」、「花子（.jhd）」、「三四郎（.jsd）」のファイルについては、「一太郎」、「花子」に関しては正しく識別することが出来たが、「三四郎」は拡張子博物館に登録されておらず未対応のため、JSR 形式（楽々はがき 住所録ファイル）に誤識別された。

また、156 個のファイルに対しての解析処理の実効時間は約 3 秒であり、高速な処理がなされた。

[11] International Digital Publishing Forum（IDPF）による規格であり、XHTML をベースとしたテキストデータ形式の電子書籍のファイルフォーマット。拡張子は「.epub」。

# 4. まとめ

極窓は FOSS ではなく、今回作成するプロトタイプシステムで使用するためには WINEX32.DLL が必要である。極窓の特徴として、JHOVE や DROID より対応可能な拡張子の数が多く、JHOVE や DROID では対応していない日本独自フォーマットの「一太郎（.jtd)」や「花子（.jhd)」も識別できる点が挙げられる。

識別精度に関しては、今回比較した DROID、JHOVE、TrID File Identifier、Metadata Extraction Tool、極窓の 5 つのツールの中では、2 番目に識別可能数が多かったが、識別できなかった数が最も多く、精度が高いとは言い難い。また、対応しているはず[8]の Office 2007 形式のフォーマットの拡張子を「.zip」に偽装したファイルや、OpenOffice 系のフォーマットのファイルの識別が出来ていない。

識別精度の向上や識別可能なフォーマット数の増加に対応するには、極窓、WinExChange 又は WINEX32.DLL の中で使用するツール（今回では WINEX32.DLL）のアップデートを待つ以外の方法がない[12]。

[12] 拡張子博物館には投稿フォームから拡張子の追加申請をすることができる。

# 5．参考文献

[1] 極－NTSOFT－55555
http://www.55555.to/
（最終アクセス日　2009年10月28日）

[2] Vector：極窓 （Windows95/98/Me / ユーティリティ） - ソフトの詳細
http://www.vector.co.jp/soft/win95/util/se085018.html
（最終アクセス日　2009年10月28日）

[3] 極窓- Wikipedia
http://ja.wikipedia.org/wiki/%E6%A5%B5%E7%AA%93
（最終アクセス日　2009年10月28日）

[4] General Purpose Library: WinEx32.DLL
http://www.csdinc.co.jp/archiver/lib/winex32.html
（最終アクセス日　2009年10月28日）

[5] 極-NTSOFT-ブログ - Windows Live
http://ntsoft.spaces.live.com/
（最終アクセス日　2009年10月28日）

[6] WINEX32.DLLのsdk.txt
（最終アクセス日　2009年10月28日）

[7] WINEX32.DLLのWinEx32.txt
（最終アクセス日　2009年10月28日）

[8] 拡張子博物館
http://www.55555.to/ext/
（最終アクセス日　2009年12月17日）