Trusted Logic

CAP File Transformer

Reference Manual

Ref. PU-2001-RT-49/2 – September 2001

This page is intentionally left blank.

1 Overview

The Trusted Logic on-card bytecode verifier for Java Card requires an off-card preprocessing of the CAP files for packages that are to be verified on-card. The preprocessing consists in two steps:

- Transformation of the Java Card bytecode contained in the Method component into an equivalent bytecode that meets the additional restrictions on bytecode imposed by the on-card verifier. These restrictions, along with the code transformation process, are described in [1].
- For library packages only, addition of a custom component called the ExportDescriptor component. This custom component is a subset of the Descriptor component, containing the minimal information that the on-card verifier needs to verify clients of the library package.

These two preprocessing steps are performed by a tool called `captransf`, written in Java. It takes as inputs:

- The CAP file for the package to be processed, in JAR format, as produced by any Java Card converter.
- The export files for every package imported by the package to be processed. Since the bytecode transformation is type-directed, these export files are needed to provide type information on imported class members.
- If the package to be processed is a library package, its export file must also be provided.

The output of the `captransf` tool is a CAP file in JAR format, ready for on-card verification and loading.

2 Using the `captransf` tool

Synopsis

`java -jar captransf.jar [`*options*`]` *export-files CAP-file*

Command-line arguments

CAP-file

The CAP file for the package to be processed (`.cap` file). This must be a complete CAP file, including the Descriptor component.

export-files

The export files (`.exp` files) for every package imported by *CAP-file*. These export files must be identical to those used for converting the package and building *CAP-file*. If the package is a library package, please provide also the export file for the package, as created by the converter.

Instead of listing export files individually, one or several directory names can be provided instead. These directories will be searched recursively for `.exp` files.

Options

`-aid` *AID-for-ExportDescriptor-component*
: For library packages only: specify the AID for the custom ExportDescriptor component. If not specified, the AID defaults to the package AID with the tag of the ExportDescriptor component appended. (See the `-tag` option.) The syntax for the AID is a colon-separated list of integer literals, possibly starting with `0x` to indicate hexadecimal. Example: `0xa0:0:0:0:0x62:0:1:2`.

`-noint`
: Specify that the output CAP file must not use the JCVM `int` instructions. By default, the transformer can generate `dup_x` and `swap` JCVM instructions that are supported only if the Java Card supports the `int` type. (It will then set the `ACC_INT` flag on the output CAP file, of course.) This ensures that the shortest, most efficient code sequences are generated by the transformer, but may turn a package that runs on any Java Card into one that runs only on `int`-supporting Java Cards. If this behavior is undesirable, specify the `-noint` flag. The transformer will then generate slightly less efficient code that is guaranteed to run on any Java Card, even those that do not support the `int` type.

`-o` *output-CAP-file*
: Specify the name of the output CAP file. If not specified, the name of the output CAP file defaults to the name of the input CAP file with a `.transf` suffix appended.

`-s` Print statistics on the transformation. With this option, the transformer prints the sizes of modified CAP components before and after the transformation, thus showing the impact of the transformation on the size of the package.

`-tag` *tag-for-ExportDescriptor-component*
: For library packages only: specify the tag for the custom ExportDescriptor component. If not specified, the tag defaults to 128.

`-version`
: Print the version number of the CAP transformer, and exit.

Examples

```
java -jar \path\to\captransf.jar
  \exportfiles\lang.exp \exportfiles\framework.exp
  com\sun\javacard\HelloWorld\javacard\HelloWorld.cap
```

Transform the `com.sun.javacard.HelloWorld` applet. The transformed CAP file is stored in `com\sun\javacard\HelloWorld\javacard\HelloWorld.cap.transf`.

```
java -jar \path\to\captransf.jar
  \exportfiles\lang.exp \exportfiles\framework.exp
  com\sun\javacard\SampleLibrary\javacard\SampleLibrary.exp
  -tag 0x84 -aid 0xa0:0x0:0x0:0x0:0x62:0x3:0x1:0x6:0x1
  -o SampleLibrary.cap
  com\sun\javacard\SampleLibrary\javacard\SampleLibrary.cap
```

Transform the `com.sun.javacard.SampleLibrary` library package. The ExportDescriptor custom component is given tag `84` in hexadecimal and AID `0xa0:0x0:0x0:0x0:0x62:0x3:0x1:0x6:0x1`. Since this is a library package, its export file must also be provided. The transformed CAP file is stored in the file `SampleLibrary.cap` in the current directory.

```
java -jar \path\to\captransf.jar
  \exportfiles\lang.exp \exportfiles\framework.exp
  com\sun\javacard\SampleLibrary\javacard\SampleLibrary.exp
  -noint -o JavaPurse.cap
  com\sun\javacard\JavaPurse\javacard\JavaPurse.cap
```

Transform the `com.sun.javacard.JavaPurse` applet. Since it imports the `com.sun.javacard.SampleLibrary` library package, the export file of the latter is also given on the command line. The `-noint` flag is set, guaranteeing that the transformed package does not require `int` support. The transformed CAP file is stored in the file `JavaPurse.cap` in the current directory.

3 Software license

4 History and feedback

Version 1.0, 2001-09-12. First public release.
Version 1.1, 2001-09-26. Fixed issue with `int` method parameters; cleaned error report.
Version 1.2, 2002-06-13. Fixed incorrect type offsets in generated `Descriptor` component.
Version 1.3, 2002-06-14. Fixed issue with CAP files containing several applets. Added recursive searching of directory arguments for `.exp` files.
Version 1.4, 2003-11-05. Correct treatment of additional, non-components files contained in input CAP file (such as the `MANIFEST` file added by the converter from the Java Card JDK 2.2). These non-component files are copied verbatim from the input CAP file to the generated CAP file.
Version 1.5, 2004-06-09. Fixed non-termination on CAP files containing empty infinite loops (such as those obtained from Java sources of the form `while (true) ;`).

Bug reports and user feedback should be e-mailed to `support.captransf@trusted-logic.fr`. If at all possible, please attach the input export files and CAP file that reproduce the unexpected behavior.

References

[1] *Protocole de gestion, procédé de vérification et de transformation d'un fragment de programme téléchargé et systèmes correspondants*, French patent application number 99 10697, INPI, published March 2nd 2001 with number 2797963. PCT extension under examination.