

How to use the Bit-efficient ACL (BE-ACL) encoding with JADE

Author: [Heikki Helin](#) ([Sonera](#)), Mikko Laukkanen ([Sonera](#)),

Date: March 29, 2001

Java platform: Sun JDK 1.2 Linux

[JADE](#) version 2.2

Since JADE 2.2, FIPA-ACL Message codecs can be plugged and activated via the command line on any JADE container. By default, the platform uses a String ACL codec. However, Bit-efficient codec can be used as an alternative FIPA-ACL encoding. This tutorial describes how to install and use the bit-efficient ACL codec with JADE. The bit-efficient ACL codec is implementation of experimental FIPA standard number [00069](#) (FIPA ACL Message Representation in Bit-Efficient Specification).

Installation

In order to install BE-ACL the following steps must be performed:

- The BE-ACL must be downloaded from the [JADE](#) download page.
- after downloading you **MUST** unzip the BE-ACL package under the root of the jade distribution tree. You should end having a hierarchy like jade/add-ons/BEFipaMessage

Compiling

The Jade Makefile rules don't take the BE-ACL into account. For handling the compilation process of the BE-ACL you have to use the 'build.xml' ant-file located in the BEFipaMessage directory. The following rules are available:

- ant compile - compiles the BE-ACL classes
- ant lib - creates the BEFipaMessage.jar archive
- ant clean - removes the compiled classes and the .jar archive
- ant doc - creates javadoc of BE-ACL
- ant examples - compiles the BE-ACL examples.

Configuration and Usage

In order to use BE-ACL codec, the BEFipaMessage.jar must be added to the classpath when starting(either by including it into the \$CLASSPATH environment variable - %CLASSPATH% under windows or by specifying it on the command line). Notice that the BE-ACL codec must be launched on all the JADE containers

Here is an example of how you would start the platform assuming you are in the root of the Jade directory:

```
java -classpath ./lib/jade.jar:./lib/jadeTools.jar:./lib/iiop.jar:./add-ons
/BEFipaMessage/lib/BEFipaMessage.jar jade.Boot -aclcodec
sonera.fipa.acl.BitEffACLCodec (for Unix)
```

or

```
java -classpath ./lib/jade.jar;./lib/jadeTools.jar;./lib/iiop.jar;./add-ons
/BEFipaMessage/lib/BEFipaMessage.jar jade.Boot -aclcodec
sonera.fipa.acl.BitEffACLCodec (for Windows)
```

More aclcodecs can be indicated separated by a ','

Examples

Using BE-ACL in Jade

When an agent wants to send a bit-efficiently encoded message, all it has to do is to indicate this in the message envelope. For example (assuming `e` is the message envelope):

```
e.setAclRepresentation("fipa.acl.rep.bitefficient.std");
```

and Jade takes care of the rest. When receiving bit-efficiently encoded messages, the agent does not have to do anything, Jade is working for him.

Using off-line examples

The BE-ACL package comes with simple off-line Encoder and Decoder that can be used to explore features BE-ACL encoding. The DummyEncoder reads ACL messages from standard input and writes bit-efficiently encoded messages to standard output. For example, assume that file `test.msg` contains one or more ACL messages encoded using string-based representation, then

```
java DummyEncoder < test.msg > be.msg
```

encodes these messages to file `be.msg`. (Notice that `BEFipaMessage.jar` and `jade.jar` must be added to the classpath).

Similarly, the DummyDecoder reads bit-efficiently encoded messages from standard input and writes the messages to standard output using string-based encoding. For example,

```
java DummyDecoder < be.msg > new.msg
```

writes the messages to file `new.msg`. The contents of `test.msg` and `new.msg` should be the same. However, this is not necessarily true. The number of white space might be different as well as the order of message parameters (if `new.msg` is encoded and decoded again, then the new result should be exactly the same).

By default, these applications does not use dynamic codetable. However, it is possible to give an additional command line parameter to each of these programs that defines the size of the code table. For example:

```
java DummyEncoder 9 < test.msg > be.msg
```

and

```
java DummyDecoder 9 < be.msg > new.msg
```

In the example above, code table size 9 is used (i.e., 2^9 entries in the code table). The size must be between 8 and 16. Notice that exactly same code table size must be given to both applications, otherwise decoding fails.

JADE is a trademark of CSELT. JADE has been developed jointly by CSELT and the Computer Engineering Group of the University of Parma.

The BE-ACL codec implementation was developed by Sonera Corporation.