

# Atacama Large Millimeter

KGB-DOC-01/06

Revision: 2.0

2003-11-11

*Software  
Manual*

Matej Šekoranja

## Generating Device Beans

*Software Manual*

Matej Šekoranja ([matej.sekoranja@ijs.si](mailto:matej.sekoranja@ijs.si))

*KGB Team, Jozef Stefan Institute*

Gasper Tkačik ([gasper.tkacik@ijs.si](mailto:gasper.tkacik@ijs.si))

*KGB Team, Jozef Stefan Institute*

**Keywords:** KGB-DOC-01/06

Author Signature:

Date:

Approved by:

Signature:

Institute:

Date:

Released by:

Signature:

Institute:

Date:

## Change Record

[illegible]

# Table of Contents

<b>Introduction.....</b>	<b>3</b>
<a href="#">1.1 References.....</a>	<a href="#">3</a>
<a href="#">1.2 Overview.....</a>	<a href="#">3</a>
<b>Generating Abeans.....</b>	<b>5</b>
<a href="#">1.3 Abeans Generator.....</a>	<a href="#">5</a>
<a href="#">1.4 Generation procedure.....</a>	<a href="#">5</a>
<b>Input file and other command line parameters.....</b>	<b>6</b>
<b>BACI to Java Mapping.....</b>	<b>6</b>
<a href="#">1.5 Out, Inout and Sequence Parameter Types.....</a>	<a href="#">6</a>
<a href="#">1.6 Structs and Enums.....</a>	<a href="#">6</a>
<a href="#">1.7 BACI model generation.....</a>	<a href="#">7</a>
<b>Conclusion.....</b>	<b>7</b>

## Introduction

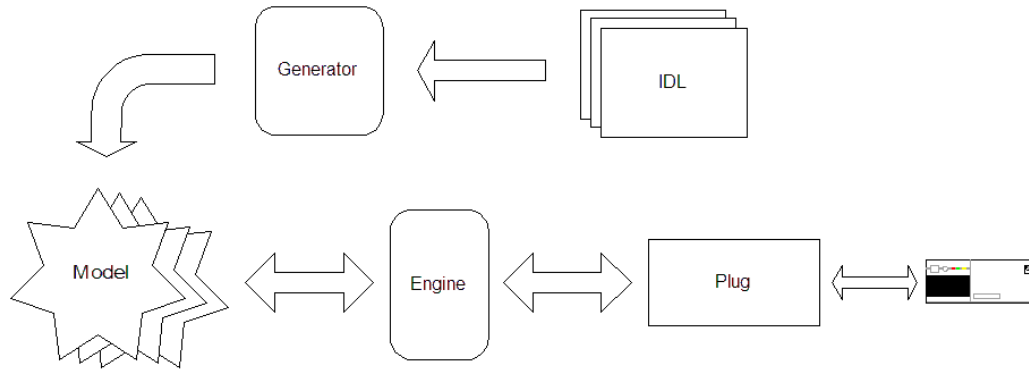
### 1.1 References

- [1] <http://abeans.cosylab.com/>
- [2] [http://www.cosylab.com/ProjectDocs/Abeans/Documentation/RFC-BACI Model and ACS Plug.html](http://www.cosylab.com/ProjectDocs/Abeans/Documentation/RFC-BACI%20Model%20and%20ACS%20Plug.html)
- [3] <http://openorb.sourceforge.net/>

### 1.2 Overview

Abeans R3 [1] is the next generation of Java based client framework for building control system applications. They provide application services and skeletons for the implementation of data flow between local client data structures and servers that access data to remote data sources, such as devices or processes. More specifically, this task is realized in two layers. Firstly, Abeans define a *model* that is a layer of Java Beans components that represent controlled objects. Usually, if controlled objects are remote devices, there might be one Java type per device type, and in an application instance a programmer might instantiate one instance of such Java bean per controlled device. Secondly, Abeans define a *plug* which is a driver layer, specific to a given model and an underlying communication system. A model and a plug in Abeans R3 communicate in a well defined way, through Abeans Engine.

The task of Abeans R3 generator is to generate model [2], i.e. modeling layer, which is actually set of Java classes following Java Beans standard. Java classes have to be generated for each ACS component including all its defined data structures.



## Generating Abeans

### 1.3 Abeans Generator

Abeans Generator is a Java application located in `abeansgen` module. It is based on OpenORB IDL Compiler [3].

Inputs are CORBA IDL files and outputs are Java classes modeling interfaces defined in the input IDL files. All generated packages are postfixed with the `abeans` package, e.g. `alma.ACS` to `alma.ACS.abeans`.

Instead of running generator directly a helper script called `abeansgen` was written. It does the IDL preprocessing, generation, compilation, packaging and at the end also cleans up compiled Java code. The final result is Java Archive (JAR) file containing generated model. JAR file names are postfixed using “Abeans” string, e.g. if IDL file name is `acsexmplHelloWorld.idl`, script will generate `acsexmplHelloWorldAbeans.jar` library.

Usage of `abeansgen` script:

```
abeansgen <IDL file>
```

Script automatically includes IDL files located in

```
$ACSWROOT/lib, $INTROOT/lib, $ALMASW_ROOTDIR/  
$ALMASW_RELEASE/JacORB/idl/jacorb, $ALMASW_ROOTDIR/  
$ALMASW_RELEASE/JacORB/idl/omg
```

directories.

If `$INSTALLDIR` environment variable is defined, generated JAR libraries are installed there, default is current directory.

### 1.4 Generation procedure

Generation procedure consists of the following steps:

1. Write (BACI compliant) IDL.
2. Compile this IDL with IDL to Java compiler to generate Java stubs and other necessary CORBA classes.
3. Use `abeansgen` script to generate JAR libraries containing compiled model of entities defined in the IDL file.

## Input file and other command line parameters

Generator is run with a number of input parameters, summarized in the following usage printout:

```
IDL to Abeans R3 BACI model compiler v1.0
based on
OpenORB IDL To Java Compiler (c) 2002 The Community OpenORB

Options
  -all
    Generate mapping for included files.
  -d directory_name
    Provide a way to specify the ouput dir. This option
    will not use the 'generated' directory.
    For example :
      abeans.models.acs.baci.generator.IDLtoAbeansCompiler demo.idl -
d /home/matej
  -D
    Define a symbol. It is equivalent to #define.
  -importLink link
  -I
    Allow specification of include directory.
    Example:
      abeans.models.acs.baci.generator.IDLtoAbeansCompiler demo.idl -
I /home/matej/idl -I ../other
  -release
    Show version number.
  -silent
    Suppress any output.
  -verbose
    Show debug output.
```

## BACI to Java Mapping

### 1.5 Out, Inout and Sequence Parameter Types

CORBA to Java mapping specifies that each for each out parameter a Java Holder instance of the appropriate type must be used for parameter passing. This requirement stems from the fact that Java passes all objects by value and not by reference (so it is impossible to pass a pointer to an integer as a parameter and have the function modify the integer). In Java it is necessary to wrap the data type into another (Holder) instance, pass that instance and let the method modify the instance member. Generator adopts the same approach and generates wraps out/inout parameters into a Holder class. Holder classes for primitives are already defined in `abeans/src/abeans.models.acs.baci.util`.

### 1.6 Structs and Enums

Generator supports generation of user-defined structures and enumerations, Holder and Helper classes are generated as well.

## 1.7 BACI model generation

Generator is also designed to generate BACI model. This means it contains some internal knowledge of BACI and requires BACI property naming convention to be followed, i.e. P<name>, ..., RO<name>, ..., RW<name>, ..., naming convention.

## Conclusion

The current limitations of the generator are:

- Generator cannot generate events declared by devices.
- Generator does not generate `BeanInfo` classes.
- Generator does not support multiple inheritance.
- Generator support only multi-dimensional arrays/sequences for IDL primitives.

While the second issue is easily resolved, other issues will be addressed when needed.