edu.mit.csail.aeolus.api
Class AeolusLib

```
java.lang.Object
  └ edu.mit.csail.aeolus.api.AeolusLib
```

```
public final class AeolusLibextends java.lang.Object
```

This class contains utilities for accessing and manipulating the labels of the current thread; it also provides launch, shutdown, registerService, getService, fork, call, getRoot and setRoot.

Method Summary	
static void	**addSecrecy**(AeolusTag t) Adds AeolusTag to the caller's secrecy label
static void	**addSecrecy**(java.util.List<AeolusTag> tags) Adds each AeolusTag in tags to the caller's secrecy label
static <T> T	**call**(java.util.concurrent.Callable<T> codeObj, PID calleePid) Invokes codeObj in the same user thread with the authority of calleePid.
static void	**createBlock**() Creates a new block for the storage of pids, tags, and labels
static EventID	**createEvent**(java.lang.String desc, java.util.List<java.lang.String> appArgs) Creates a new event in the Aeolus audit trails.
static EventID	**createEvent**(java.lang.String desc, java.util.List<java.lang.String> appArgs, java.util.List<EventID> elist) Creates a new event in the Aeolus audit trails.
static void	**declassify**(AeolusTag t) Remove AeolusTag from the caller's secrecy label
static void	**declassify**(java.util.List<AeolusTag> tags) Remove AeolusTags from the caller's secrecy label
static void	**endorse**(AeolusTag t) Add AeolusTag to the caller's integrity label
static void	**endorse**(java.util.List<AeolusTag> tags) Add AeolusTags in List to the caller's integrity label
static void	**fork**(java.lang.Runnable codeObj) Runs code in a different user thread.
static void	**fork**(java.lang.Runnable codeObj, PID calleePid) Runs code in a different user thread.

static java.util.List<java.lang.Integer>	**getBasis**() Returns caller's basis
static EventID	**getEventID**() Returns the EventID of the most recent event logged for the thread.
static AeolusLabel	**getIntegrity**() Returns caller's integrity label
static PID	**getPID**() Returns caller's PID
static AeolusShared	**getRoot**() Returns the shared state root.
static AeolusLabel	**getSecrecy**() Returns caller's secrecy label
static java.lang.Object	**getService**(java.lang.String remoteHostName, java.lang.String serviceName, java.lang.Class<?> serviceClass) Return service stub that implements the interface specified by serviceName.
static void	**launch**(java.lang.String hostname, PID pid, java.lang.String appName, java.lang.String appParams) Launches a new platform instance on the specified Aeolus node with the specified authority.
static void	**makeCurrentBlock**(PID p) Makes the block that contains p the current block for the storage of pids, tags, and labels
static void	**registerService**(java.lang.String serviceName, java.lang.Class<?> service) Registers a service with Aeolus, which allows it to be accessed remotely.
static void	**removeIntegrity**(AeolusTag t) Removes AeolusTag from the caller's integrity label
static void	**removeIntegrity**(java.util.List<AeolusTag> tags) Removes AeolusTags from the caller's integrity label
static void	**setRoot**(AeolusShared root) Set the shared state root.
static void	**shutdown**() Shuts down the virtual node of the caller

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Detail

addSecrecy

```
public static void addSecrecy(AeolusTag t)
                       throws AeolusException
```

Adds AeolusTag to the caller's secrecy label

Parameters:

`t` - - AeolusTag to add to the secrecy label

Throws:

`AeolusException` - - if an attempt to change a label occurs while the thread is running inside a shared object

---

addSecrecy

```
public static void addSecrecy(java.util.List<AeolusTag> tags)
                       throws AeolusException
```

Adds each AeolusTag in tags to the caller's secrecy label

Parameters:

`tags` - - List of AeolusTags to add to the secrecy label

Throws:

`AeolusException` - - if an attempt to change a label occurs while the thread is running inside a shared object

---

call

```
public static <T> T call(java.util.concurrent.Callable<T> codeObj,
                         PID calleePid)
              throws AuthorityException,
                     java.lang.Exception
```

Invokes codeObj in the same user thread with the authority of calleePid. Caller principal must be authorized to switch to calleePid. After invocation, the caller PID is restored, even if the call throws an exception.

Parameters:

`codeObj` - - code to invoke

`calleePid` - - principal to run the code on behalf of

Throws:

`AuthorityException` - - if caller principal cannot switch to calleePid

`java.lang.Exception` - - if an exception is thrown by the invoke method of the codeObj

---

createBlock

```
public static void createBlock()
                       throws InfoFlowControlException
```

Creates a new block for the storage of pids, tags, and labels

Throws:

`InfoFlowControlException` - if the thread's secrecy label is not null

---

createEvent

```
public static EventID createEvent(java.lang.String desc,
                                  java.util.List<java.lang.String> appArgs)
                           throws AeolusException
```

Creates a new event in the Aeolus audit trails. Sets the record fields to the provided arguments. This event will be automatically correlated with other events in the same user thread.

Parameters:

`desc` - - a String describing the event

`appArgs` - - a list of Strings for this event chosen by the application

Returns:

the EventID for the new event

Throws:

`AeolusException`

---

createEvent

```
public static EventID createEvent(java.lang.String desc,
                                  java.util.List<java.lang.String> appArgs,
                                  java.util.List<EventID> elist)
                           throws AeolusException
```

Creates a new event in the Aeolus audit trails. Sets the record fields to the provided arguments. This event will be automatically correlated with other events in the same user thread.

Parameters:

`desc` - - a String describing the event

`appArgs` - - a list of Strings for this event chosen by the application

`elist` - - a list of predecessors for this event

Returns:

the EventID for the new event

Throws:

`AeolusException`

---

declassify

```
public static void declassify(AeolusTag t)
                       throws AeolusException
```

Remove AeolusTag from the caller's secrecy label

Parameters:

`t` - - AeolusTag to remove from the secrecy label

Throws:

`AuthorityException` - - if caller principal does not have authority for t

`AeolusException` - - if an attempt to change a label occurs while the thread is running inside a shared object

---

declassify

```
public static void declassify(java.util.List<AeolusTag> tags)
                       throws AuthorityException,
                              AeolusException
```

Remove AeolusTags from the caller's secrecy label

Parameters:

`tags` - - List of AeolusTags to remove from the secrecy label

Throws:

`AuthorityException` - - if caller principal does not have authority for each tag in tags

`AeolusException` - - if an attempt to change a label occurs while the thread is running inside a shared object

---

endorse

```
public static void endorse(AeolusTag t)
                    throws AuthorityException,
                           AeolusException
```

Add AeolusTag to the caller's integrity label

Parameters:

`t` - - AeolusTag to add to the integrity label

Throws:

`AuthorityException` - - if caller principal does not have authority for t

`AeolusException` - - if caller is inside a shared state object

---

endorse

```
public static void endorse(java.util.List<AeolusTag> tags)
                    throws AuthorityException,
                           AeolusException
```

Add AeolusTags in List to the caller's integrity label

Parameters:

`tags` - - List of AeolusTags to add to the integrity label

Throws:

`AuthorityException` - - if caller principal does not have authority for each AeolusTag

`AeolusException` - - if caller is inside a shared state object

---

fork

```
public static void fork(java.lang.Runnable codeObj)
                 throws AeolusException
```

Runs code in a different user thread. The new thread runs with the authority of the caller. The code object, codeObj, is copied to the new process.

Parameters:

`codeObj` - - code to invoke

Throws:

`AuthorityException` - - if thread can not switch to pid

`AeolusException` - - if caller running in shared state

---

fork

```
public static void fork(java.lang.Runnable codeObj,
                        PID calleePid)
                 throws AuthorityException,
                        AeolusException
```

Runs code in a different user thread. The new thread runs with the authority of calleePid. Caller must be authorized to switch to calleePid. The code object, codeObj, is copied to the new process.

Parameters:

`codeObj` - - code to invoke

`calleePid` - - principal to run the new user thread with

Throws:

`AuthorityException` - - if caller principal cannot switch to calleePid

`AeolusException` - - if a platform failure occurs when starting the new user thread

---

getBasis

```
public static java.util.List<java.lang.Integer> getBasis()
```

Returns caller's basis

---

getEventID

```
public static EventID getEventID()
                          throws AeolusException
```

Returns the EventID of the most recent event logged for the thread.

Throws:

`AeolusException` - - if logging is not enabled.

---

getIntegrity

```
public static AeolusLabel getIntegrity()
```

Returns caller's integrity label

---

getPID

```
public static PID getPID()
```

Returns caller's PID

---

getRoot

```
public static AeolusShared getRoot()
```

Returns the shared state root. Returns null if the root has not been set.

---

getSecrecy

```
public static AeolusLabel getSecrecy()
```

Returns caller's secrecy label

---

getService

```
public static java.lang.Object getService(java.lang.String remoteHostName,
                                          java.lang.String serviceName,
                                          java.lang.Class<?> serviceClass)
```

Return service stub that implements the interface specified by serviceName. Remote methods should be invoked using the service stub by casting the returned object to the remote service interface; for example:

Foo service = (Foo) getService("localhost", "path.to.Foo", serviceClass);

Serializable z = service.methodA(x, y);

A method invocation on the service stub will result in a remote procedure call to the remote host.

Parameters:

`remoteHostName` - - node where the desired remote service was registered
`serviceName` - - name that the service is bound to on the remote host
`serviceClass` - - local Class which defines the interface of the service

Returns:

service stub, must be cast to the interface specified by serviceName

Throws:

`java.lang.NullPointerException` - - if serviceClass is null or any of its elements are null
`java.lang.IllegalArgumentException` - - if any of the restrictions on the

getProxyClass method in Java's Proxy class are violated.

---

launch

```
public static void launch(java.lang.String hostname,
                          PID pid,
                          java.lang.String appName,
                          java.lang.String appParams)
                   throws InfoFlowControlException,
                          AuthorityException,
                          AeolusException,
                          RpcException,
                          java.lang.reflect.InvocationTargetException,
                          java.lang.ClassNotFoundException,
                          java.lang.NoSuchMethodException
```

Launches a new platform instance on the specified Aeolus node with the specified authority.

This instance starts executing the application of the given name. Caller's secrecy label must be null to execute this request. Caller principal must act for the specified pid.

Parameters:

`hostname` - - node where the platform instance should be launched
`pid` - - principal to launch the platform instance on behalf of
`appName` - - application to start running in the new platform instance
`appParams` - - parameters to the application

Throws:

`InfoFlowControlException` - - if caller's secrecy label is not null
`AuthorityException` - - if caller does not act for pid
`RpcException` - - if failure in communicating with hostname
`java.lang.ClassNotFoundException` - - if application class was not found on hostname
`java.lang.NoSuchMethodException` - - if main method was not found in the application
`java.lang.reflect.InvocationTargetException` - - if an exception was thrown by the main method (declared or runtime exception), this exception is a wrapper; use e.getCause() to view the actual thrown exception
`AeolusException` - - if any other failure while launching the application or if an unexpected platform failure occurred

---

makeCurrentBlock

```
public static void makeCurrentBlock(PID p)
                            throws InfoFlowControlException,
                                   AuthorityException
```

Makes the block that contains p the current block for the storage of pids, tags, and labels

Throws:

`InfoFlowControlException` - if the thread's secrecy label is not null
`AuthorityException` - if the thread does not act-for p

registerService

```
public static void registerService(java.lang.String serviceName,
                                   java.lang.Class<?> service)
                            throws InfoFlowControlException,
                                   java.lang.NullPointerException
```

Registers a service with Aeolus, which allows it to be accessed remotely. The service must implement an interface listing any methods that it exposes remotely. The name of this interface must be provided as the name of the service. Any parameters and return values from the methods of this interface must be Serializable. The service name is bound to the virtual node (VN) where the service is registered. Any invocations of the remote methods provided by this service will run in this VN with the authority of the VN principal. Every time a remote method of the service is invoked, a new instance of the service will be created from the specified service class and the method will be executed by this service instance. If a service by the same name exists, it will be replaced.

Parameters:

`serviceName` - - name of the interface implemented by the service

`service` - - class implementing the service (must implement the service interface)

Throws:

`InfoFlowControlException` - - if caller's secrecy label is not null

`java.lang.NullPointerException` - - if service is null

---

removeIntegrity

```
public static void removeIntegrity(AeolusTag t)
                            throws AeolusException
```

Removes AeolusTag from the caller's integrity label

Parameters:

`t` - - AeolusTag to remove from integrity label

Throws:

`AeolusException` - - if an attempt to change a label occurs while the thread is running inside a shared object

---

removeIntegrity

```
public static void removeIntegrity(java.util.List<AeolusTag> tags)
                            throws AuthorityException,
                                   AeolusException
```

Removes AeolusTags from the caller's integrity label

Parameters:

`tags` - - List of AeolusTag to remove from integrity label

Throws:

`AuthorityException` - - if an attempt to use authority occurs while in shared state

`AeolusException` - - if caller is inside a shared state object

setRoot

```
public static void setRoot(AeolusShared root)
                    throws InfoFlowControlException
```

Set the shared state root. Only a process with an empty secrecy label can set the root.

Throws:

`InfoFlowControlException` - - if caller's secrecy label isn't null

shutdown

```
public static void shutdown()
                     throws InfoFlowControlException
```

Shuts down the virtual node of the caller

Any threads running in the virtual node (VN) will be terminated immediately. The VN's service will be deregistered and the VN will be shutdown. Caller's secrecy label must be null to execute this request.

Throws:

`InfoFlowControlException` - - if caller's seclabels are not null