Apache Directory Server

LDAP Stored Procedures and Triggers in ApacheDS

Ersin Er

ersiner@apache.org

Agenda

- Stored Procedures
  - Why do we need them in LDAP?
  - Representation and Execution
  - Security Issues
- Triggers
  - Why do we need them in LDAP?
  - Model of LDAP Triggers
  - Integration with LDAP Stored Procedures
  - Trigger Execution Domains – X.500 in Action

Why Stored Procedures in LDAP?

- Bulk processing / Performance
- Controlled by user
- Extending server's capability *easily*

- What about LDAP Extended Operations?
  - Hard to implement a new one
  - Not really for users

Stored Procedures – Questions
and answers to simple ones

- How to implement?
  - As a piece of code implemented in any technology
- Where to store and how to represent?
  - In DIT with schema elements (so can be manipulated via standard LDAP operations)
  - Location of/locating stored procedures in DIT is subject to vendor implementation
- How to execute?
  - Parameters?
  - Return value?
- Security?

Stored Procedures – Abstract Representation

```
objectclass ( storedProcUnit_oid
   NAME 'storedProcUnit'
   SUP top
   ABSTRACT
   MUST ( storedProcLangId $ storedProcUnitName ) )

attributetype ( storedProcLangId_oid
   NAME 'storedProcLangId'
   EQUALITY caseExactIA5Match
   SYNTAX IA5StringSyntax_oid
   SINGLE-VALUE )

attributetype ( storedProcUnitName_oid
   NAME 'storedProcUnitName'
   EQUALITY caseExactIA5Match
   SYNTAX IA5StringSyntax_oid
   SINGLE-VALUE )
```

Java Stored Procedures - Representation

- storedProcLangId → 'Java'
- storedProcUnitName → Fully qualified name of a Java class
  - Ex: com.example.ldap.util.sp.DITUtilities

```
Objectclass ( javaStoredProcUnit_oid
   NAME 'javaStoredProcUnit'
   SUP storedProcUnit
   STRUCTURAL
   MUST ( javaByteCode ) )

attributetype ( javaByteCode_oid
   NAME 'javaByteCode'
   SYNTAX binarySyntax_oid
   SINGLE-VALUE )
```

Execution of Stored Procedures

- Server Side vs. Client Side (External)
- An extended operation is defined for External Invocation:

```
StoredProcedureExecutionRequestValue ::= SEQUENCE {
        name IA5String,
        parameters SEQUENCE OF Parameter OPTIONAL }

Parameter ::= SEQUENCE OF {
        type OCTET STRING OPTIONAL,
        value [0] OCTET STRING }

StoredProcedureExecutionResponseValue ::= SEQUENCE {
        returnType OCTET STRING OPTIONAL,
        returnValue [0] OCTET STRING OPTIONAL }
```

- Encoding/decoding semantics of defined elements are subject to the implementation technology and vendor

Client-side Invocation of Java Stored Procedures

- StoredProcedureExecutionRequestValue.name
  - Fully qualified name of a Java class + name of a public static method of it
  - Ex: com.example.ldap.util.sp.DITUtilities:deleteSubtree
- Parameter.type
  - Unnecessary as Java has RTTI
- Parameter.value
  - Any Java-serialized object
- StoredProcedureExecutionResponseValue
  - .returnType : Unnecessary as Java has RTTI
  - .returnValue: Any Java-serialized object

A special parameter for Java SPs

- LdapContextParameter
  - Includes a string holding a DN
- ApacheDS supplies a JNDI context at the specified DN with the user's credentials
- *Why do we need it?*

Stored Procedures - Demo

- DelTree* for LDAP
- RFC4511: *Only leaf entries (those with no subordinate entries) can be deleted with this operation*
- Simple to implement it with Java (JNDI)
- Why not let the server run it?
- Let’s see Java LDAP Stored Procedures in Action

- * You remember the old MS-DOS command *DELTREE*?
  Or are you a *rm –rf* fan?

Stored Procedures – Security Issues

- Directory operations on stored procedures
  - Who can do what on stored procedures
    - Can be managed by the access control subsystem
- Permissions used during invocation
  - Caller's verses owner's
    - Owner makes sense
- Authorization for invoking stored procedures
  - X.500's *grantInvoke* ?
- Stored procedures' capabilities within the server
  - Run in a sandbox

Stored Procedures *Briefly*

- Considering the external invocation capability,

  LDAP stored procedures allow users to effectively define their own *extended operations* without requiring any server software extensions

Why Triggers in LDAP?

- Tracking DN references (referential integrity)
- Custom action needs upon some operations on some entries especially for IdM systems (logging, firing an external process, cascaded operations)
- Existing solutions lacks some capabilities or are hard to use (e.g. requires server side plug-ins)

Specification of A Trigger

- Triggering Event
- Triggered Action
- Action Time (with respect to the Event time)
- Trigger Scope

Specification of An LDAP Trigger

- Triggering Event: Change inducing LDAP operations (Modify, Add, Delete, ModifyDN)
- Triggered Action: LDAP Stored Procedures!
- Action Time (with respect to the Event time): AFTER
- Trigger Scope: Individual entries or sets of entries...

LDAP Trigger Specification

- An LDAP Trigger Specification is represented as a String

```
AFTER Delete
  CALL “com.example.ldap.util.sp.BackupTools:backupDeletedEntry”
    ( $ldapContext “ou=backup,ou=system”, $name, $deletedEntry );
```

- To make the scope of the trigger an individual entry put this specification in an `entryTriggerSpecification` attribute in that entry

Leveraging Stored Procedures

- Stored Procedures invoked by Triggers can be supplied:
  - operation specific standard request parameters
    ($entry for Add, $modification for Modify, ...)
  - operation specific useful parameters
    ($deletedEntry for Delete, ...)
  - generic parameters
    ($ldapContext, $operationPrincipal, ...)
- For the Java implementation, all of the available parameters have predefined corresponding Java types

Triggers - Demo

- Task: Make an entry backed up upon deletion

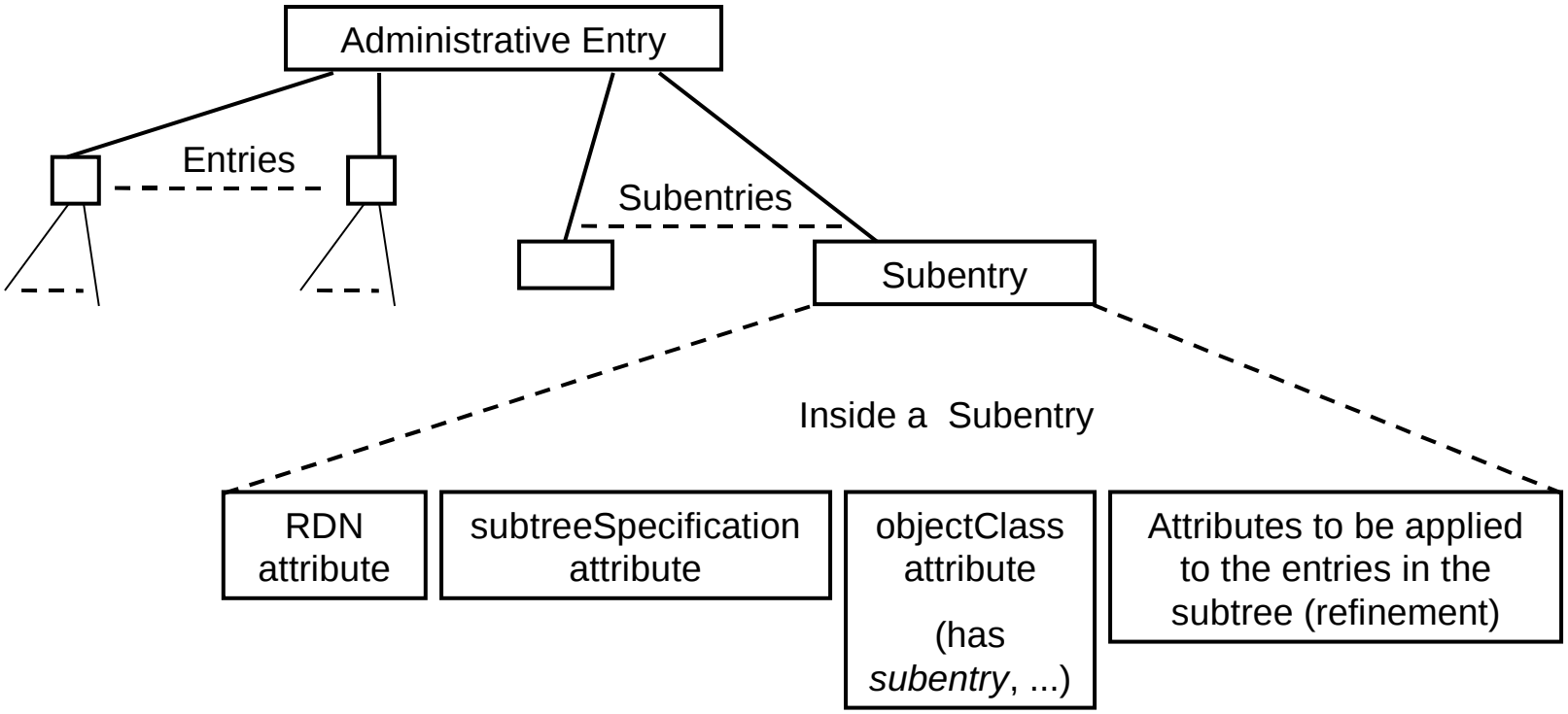
Problems with Individual Entry Scope

- The trigger was effective only on a single entry
- After a delete operation even the trigger specification is lost!

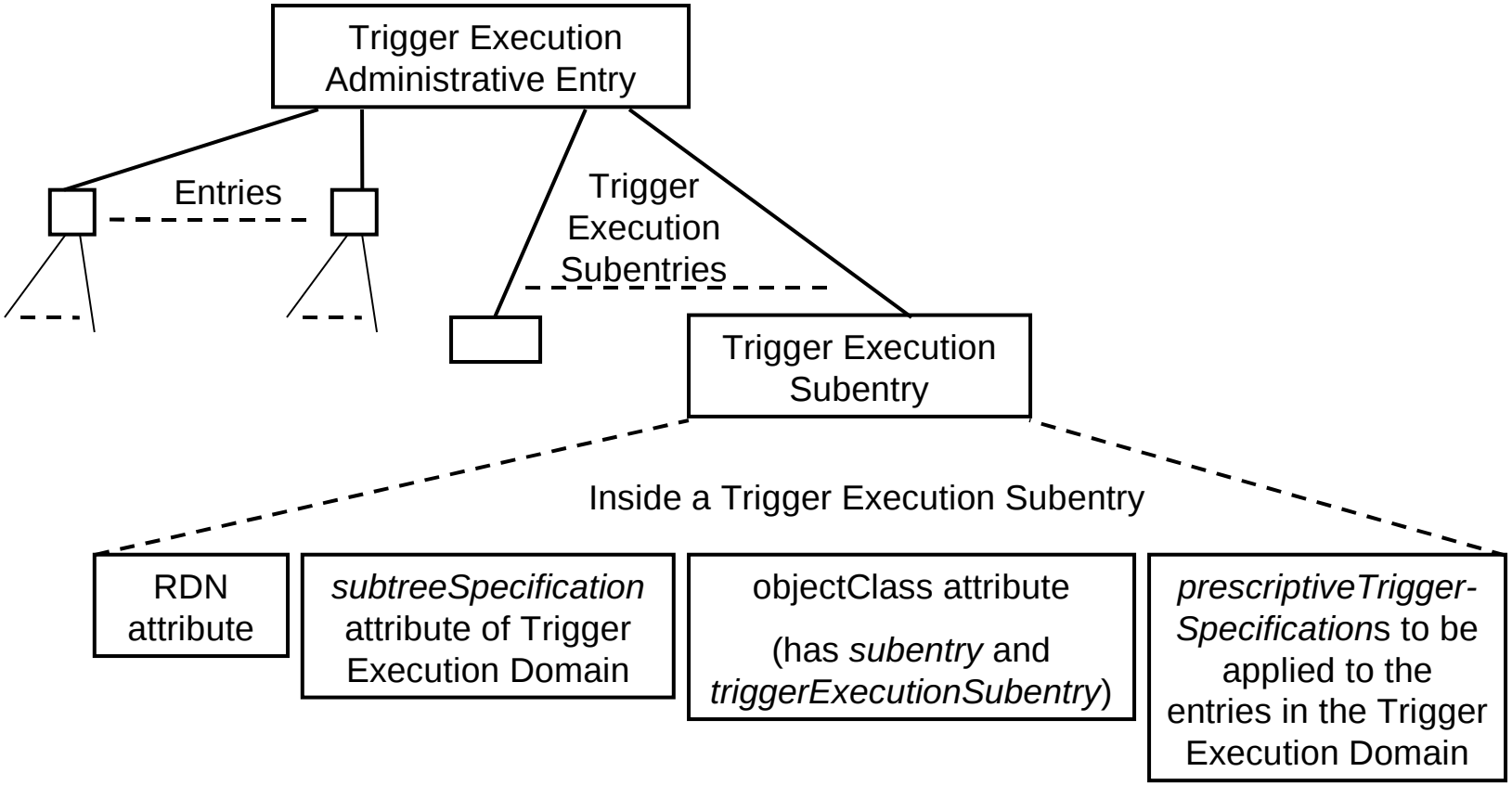
Trigger Execution Domains

- X.500 Subentries and subtreeSpecification
  - A Subentry holds a subtreeSpecification attribute
  - subtreeSpecification allows specifying a *subtree of entries with chop specifications and refinements*
  - Other attributes in the Subentry are *applied* to the selection of entries according to the administrative aspect associated with the subentry
  - A building block of X.500 Administrative Model
  - RFC 3672 - Subentries in the Lightweight Directory Access Protocol
- Trigger Execution Domains
  - Instead of *entryTriggerSpecification*,
  - use *prescriptiveTriggerSpecification* in *triggerExecutionSubentry*
  - to define triggers on a *set of entries*

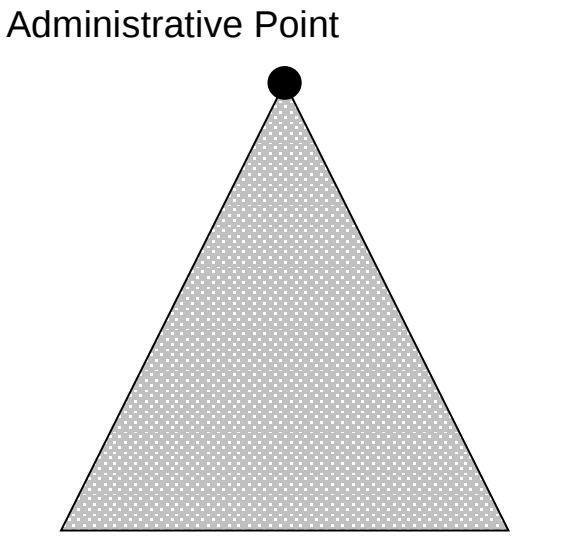
X.500 Administrative Model



X.500 Administrative Model – Trigger Execution Aspect

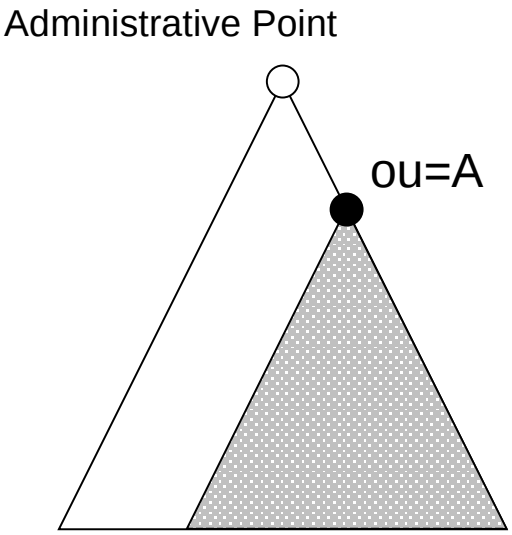


What can be specified *(How a TED can be specified)* with a subtreeSpecification ? (1)



subtreeSpecification=
{ }

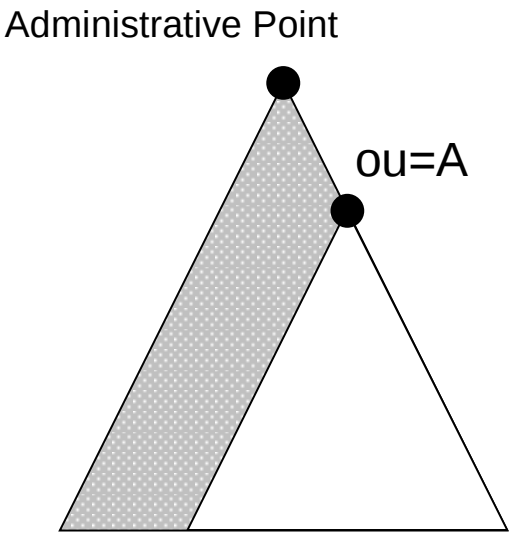
What can be specified *(How a TED can be specified)* with a subtreeSpecification ? (2)



subtreeSpecification=

{ base “ou=A” }

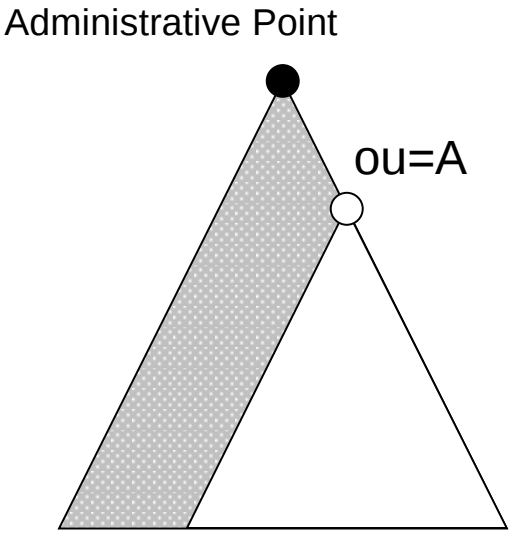
What can be specified *(How a TED can be specified)* with a subtreeSpecification ? (3)



subtreeSpecification=

{ specificExclusions { chopAfter: “ou=A” } }

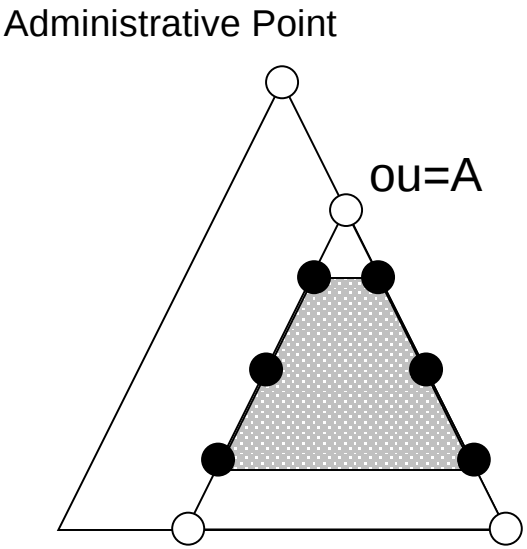
What can be specified *(How a TED can be specified)* with a subtreeSpecification ? (4)



subtreeSpecification=

{ specificExclusions { chopBefore: “ou=A” } }

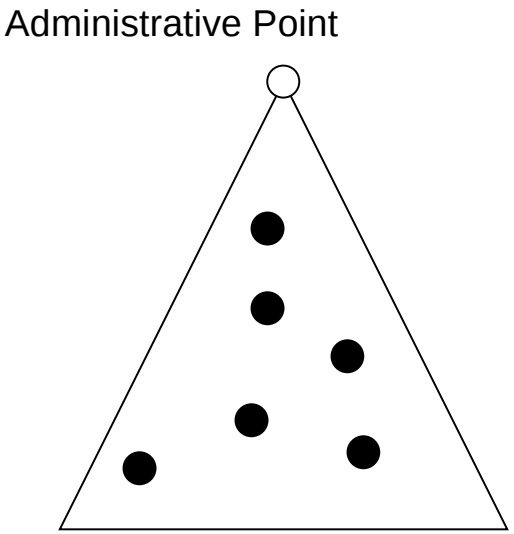
What can be specified *(How a TED can be specified)* with a subtreeSpecification ? (5)



subtreeSpecification=

{ base “ou=A”, minimum 1, maximum 3 }

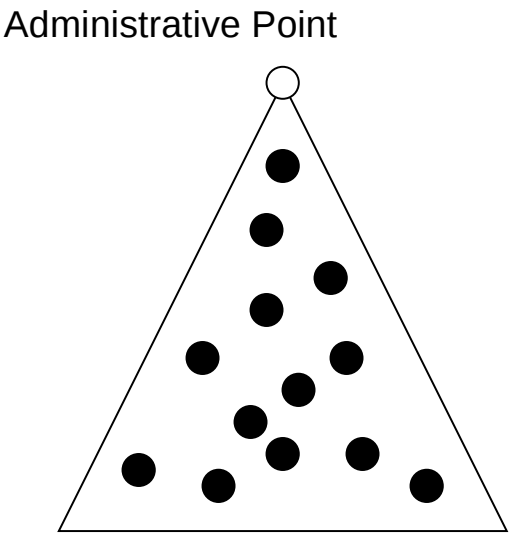
What can be specified *(How a TED can be specified)* with a subtreeSpecification ? (6)



subtreeSpecification=

{ specificationFilter item:student }

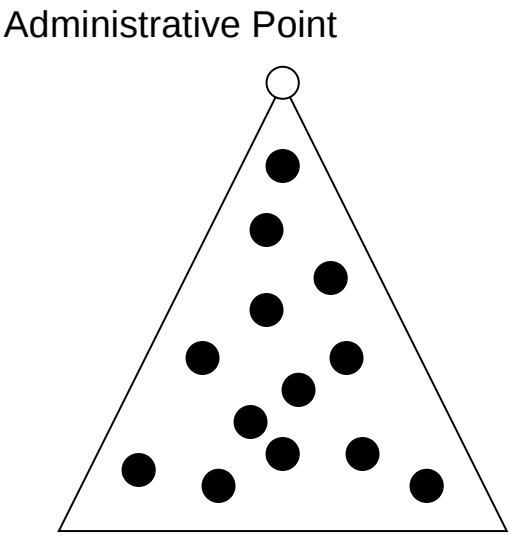
What can be specified *(How a TED can be specified)* with a subtreeSpecification ? (7)



subtreeSpecification=

{ specificationFilter or: { item:student, item:faculty } }

What can be specified *(How a TED can be specified)* with a subtreeSpecification ? (8)



subtreeSpecification=

{ specificationFilter (&(objectClass=person)(title=engineer)) }

Triggers – Demo with TED

- Task: Backup engineer users' entries when they are deleted

Triggers *Briefly*

- Change inducing operations as triggering events
- LDAP Stored Procedures as triggered events
- Individual or sets of entries as Trigger Scope. Powerful entry set selection via X.500 Administrative Model.
- Many triggers can effective on a single entry upon the same type or different operations.
- Many stored procedures can be run with a single trigger. Guaranteed order of execution.
- Operation specific or generic parameter injection to stored procedures. (Completely transparent to the trigger specifications.)

Future

- Support for Java based scripting languages for stored procedures
- Triggers for search operations
- Before and InsteadOf triggers
- GUI support via Apache Directory Studio

Apache Directory Server – In a Nutshell

- Pure Java and Embeddable
- Very extendible architecture
- X.500 Administrative Model and Access Control, Collective Attributes, Triggers built on top of it
- Pluggable protocol providers for LDAP, Kerberos 5, Change Password
- OpenGroup certified as a LDAPv3 compliant server (Renewed the certification just today for one year more!)
- Step by Step User's Guide (See it!)
- Great community

Questions? Thank you!

LDAP Stored Procedures and Triggers in ApacheDS

Ersin Er

ersiner@apache.org

http://people.apache.org/~ersiner