LDAP Stored Procedures and Triggers Arrive in ApacheDS

• Originally presented at ApacheCon US 2006 in Austin, Texas
• Latest presentation materials are at http://people.apache.org/~ersiner
• Presented by Ersin Er, ersiner@apache.org
Agenda

• Stored Procedures
  – Why do we need them in LDAP?
  – Representing Stored Procedures
  – Executing Stored Procedures

• Triggers
  – Why do we need them in LDAP?
  – Model of LDAP Triggers
  – Integration with LDAP Stored Procedures

• Demos
Stored Procedures for LDAP (Why?)

- Bulk processing
- Controlled by user
- Extending server’s capability *easily*

- LDAP Extended Operations?
Model of LDAP Stored Procedures

- Implementation technology
- Storage place
- Storage format
- Storage method
- Calling
  - Parameters
  - Return value
- Security
What’s an LDAP stored procedure?

• A piece of code
• Implemented in any technology
• Stored in the Directory Information Tree
• Represented with schema elements
• Manipulated by standard LDAP operations (add, delete)
Stored Procedures in ApacheDS

- “Java” implementation of the generic model
- A “Java” LDAP stored procedure is
  - A public static method of a Java class
  - Represented by two attributes and an object class
  - Stored with its class (as expected) in compiled form (byte-code)
Calling a Stored Procedure

• Call from where?
  – Client side
  – Server side

• No standard SP Call operation

• For calling any LDAP stored procedure from client side
  – Use Stored Procedure Execution (Extended) Operation
Stored Procedure Execution
(Extended) Operation

• Name of the stored procedure
• Where to find the stored procedure (optional)
  – A base search context (DistinguishedName)
  – Search scope: base, one, whole (Optional)
• Parameters (optional)
  – type information (optional)
  – value
DEM0 1

• Let’s load a “Hello World” SP and call it!

```java
public class Greeter {
    public static void helloWorld() {
        System.out.println("Hello World!");
    }
}
```
DEMO 2 – SP Parameters and Return Values

- public class Greeter

  {  
      public static String sayHello( String who, Integer times )
      {
          StringBuffer buffer = new StringBuffer();
          for ( int i = 0; i < times.intValue(); i++ )
          {
              buffer.append( "Hello " );
          }
          buffer.append( who );
          buffer.append( '!' );
          return buffer.toString();
      }
  }
“Java” SP execution progress
(A reflection story)

• Find the SP entry
  – Use the SP name (what) and search context (where)
• Extract class name from SP name
• Load the class
• Extract method name from SP name
• Find the method in the class
  – Use method name and check parameters for assignment compatibility
• Call the method supplying parameters
• Return back the result Object
A special SP parameter

- type: “ldapContext”
- value: A distinguished name (as a String object)

- ApacheDS supplies a JNDI context at the specified DN with the user’s credentials

- Why do we need it?
• Let’s do a real world example
• With delete operation only a single entry can be deleted at once
• It’s a common requirement to delete a subtree at once
• There is a delete operation control for this but it is not adopted by the mainstream
• Let’s write our own deleteSubtree SP, load and call it!
Security Issues

• Directory operations on stored procedures
  – Who can do what on stored procedures
• Permissions used during execution
  – Executor’s verses owner’s
• Authorization for executing stored procedures
• Stored procedures’ capabilities within the server
Security Issues and ApacheDS

- Stored procedures
  - are standard user objects
  - any operation on them is possible
  - and subject to access control

- Stored procedures are executed with executor’s permissions

- Currently, who is authorized to read an SP is also authorized to execute it

- Currently, execution is not sandboxed
Stored Procedures – Briefly

• LDAP stored procedures allow users to effectively define their own extended operations without requiring any server software extensions
Triggers for LDAP (Why?)

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

<Trigger Specification> :
<Action Time>
<Trigger Event>
<Triggered Action>
An LDAP Trigger

- **Action Time**: AFTER
- **Trigger Event**: Change inducing LDAP operations
- **Triggered Action**: LDAP Stored Procedures!

- Which entries is a trigger defined *on*?
  - A specific entry
  - *Trigger Execution Domains*

- All these information are stored as regular schema objects (so can be browsed, replicated, etc.)
Trigger Specification Examples

- **AFTER Delete**
  
  CALL "BackupUtilities.backupDeletedEntry" ($ldapContext(""), $name, $deletedEntry)

- **AFTER Add**
  
  CALL "Logger.logAddOperation" ($entry, $attributes, $operationPrincipal)
Stored Procedures – Triggers Integration

- SPs can be supplied parameters like:
  - operation specific standard request parameters ($entry for Add, $name for Delete, ...)
  - operation specific useful parameters ($deletedEntry for Delete, ...)
  - generic parameters ($ldapContext, $operationPrincipal, ...)

- All available parameters have predefined corresponding Java types

- SP call options are supported as specified in the SP Execution Operation
DEMO 1

• Let’s backup an entry when it’s deleted

• Write a Java stored procedure and load it

• Put an entryTriggerSpecification attribute in an entry
  – AFTER Delete
  – CALL "BackupUtilities.backupDeletedEntry"
  – ($ldapContext(""),$name,$deletedEntry)
Was it impressive?

- Not very much!
- The trigger was effective only on a single entry
- And even our trigger specification has been deleted!
Trigger Execution Domains (TED)

- **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
  - A building block of X.500 Administrative Model

- **Trigger Execution Domains**
  - Instead of entryTriggerSpecification,
  - use *prescriptiveTriggerSpecification* in *triggerExecutionSubentry*
  - to define triggers on a set of entries
X.500 Administrative Model

- Administrative Entry
  - Entries
  - Subentries
  - Subentry
    - RDN attribute
    - subtreeSpecification attribute
    - objectClass attribute (has subentry, ...)
    - Attributes to be applied to the entries in the subtree (refinement)

Inside a Subentry
X.500 Administrative Model – Trigger Execution Aspect

Trigger Execution Administrative Entry

Entries

Trigger Execution Subentries

Trigger Execution Subentry

Inside a Trigger Execution Subentry

RDN attribute

subtreeSpecification attribute of Trigger Execution Domain

objectClass attribute (has subentry and triggerExecutionSubentry)

prescriptiveTriggerSpecifications to be applied to the entries in the Trigger Execution Domain
What can be specified
(How a TED can be specified)
with a subtreeSpecification? (1)

Administrative Point

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

```plaintext
Administrative Point

subtreeSpecification=
{ base "ou=A" }
```
What can be specified
(*How a TED can be specified*)
with a subtreeSpecification? (3)

Administrative Point

```
subtreeSpecification=
{ specificExclusions { chopAfter: "ou=A" } }
```
What can be specified *(How a TED can be specified)*
with a subtreeSpecifications? (4)

subtreeSpecifications =

{ specificExclusions { chopBefore: "ou=A" } }
What can be specified
(How a TED can be specified)
with a subtreeSpecification? (5)

Administrative Point

subtreeSpecification=
{ base “ou=A”, minimum 1, maximum 3 }
What can be specified
(How a TED can be specified)
with a subtreeSpecification? (6)

Administrative Point

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 } }
 DEMO 2

• Let’s backup *any* entry when it’s deleted

• `triggerExecutionSubentry`
  – `subtreeSpecification`
  – `prescriptiveTriggerSpecification`
What’s coming next?

• BEFORE and INSTEADOF Triggers
• Search Operation Triggers
• Mutable parameters for Stored Procedures called from Triggers
• Fine grained security control

• and more: dev@directory.apache.org
LDAP Stored Procedures and Triggers Arrive in ApacheDS

• Originally presented at ApacheCon US 2006 in Austin, Texas

• Latest presentation materials are at http://people.apache.org/~ersiner

• Presented by Ersin Er, ersiner@apache.org