Auditing in OpenEdge®

Product Readiness 10.1A FCS Training

Joe Genovese
Direct End-User Pre-Sales
March 2008

Agenda

- Overview
- Getting started
- Audit Policy Maintenance
- Authentication
- Database Events
- Application Events
- Internal Events

Auditing Overview

Mission Statement:

Provide an auditing framework that can supply an uninterrupted trail of an application client's access to its operations and data.

Auditing

Driving factors

- Regulatory compliance
  - Sarbanes-Oxley Act, CFR Part 11, HIPAA, European Union's Annex 11, European Union Data Protection Directive, etc
- Non-repudiation of Audit data
- Consistency
  - 4GL, SQL, database utilities
- Immediacy of Audit data

Auditing

Key features

- Provide an audit trail of
  - Application operations
  - Context
  - Data
- Performance, scalability, storage size
- Secure, tamper-resistant
- General purpose audit logging
  - Code coverage, debugging / tracing, event analysis

Auditing Capabilities

- Database Auditing
  - Record level events
    - Create, update, delete (CUD) operations
- Application Auditing
  - Contextual, event groups, operations
- Internal auditing
  - Tools, utilities, connections, schema changes

Authentication

Secure Auditing is key to compliance

- Audit trails can tell you who did what, when, where and how
- Must reflect the verifiable identity of the real application user
- Must be complete, accurate and non-refutable
  - Prove audit policy and data has not been tampered with

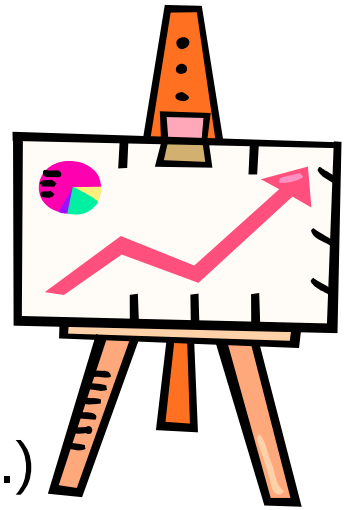
Security of Audit Data

- Separation of duty
  - Audit administrator
  - Application audit event inserter
  - Audit data archiver
  - Audit data reporter
- No updates to audit data – EVER!
- No deletion of defined *events*
- Audit data is sealed to prevent tampering
  - Within and outside of the database

Auditing

Why use it in place of your own solution?

- Common built-in auditing for both SQL/4GL clients
- Flexible audit policy management
- Secure audit data, policy and utilities
  - Separation of duty
  - Purposed audit permissions
  - Verified user identity
  - Secure utilities and sealed data
- Internal audit events (utilities, schema changes, etc.)
- Performance, performance, performance
- High performance archiving
- Multi-database, multi-platform, multi-application



Agenda

- Overview
- Getting started
- Audit Policy Maintenance
- Authentication
- Database Events
- Application Events
- Internal Events

Before You Start

Decide what to audit

- Consider your reporting needs
  - Database operations
  - Application operations
- How much information to record
  - Table and field level
  - Contextual information
- Which fields constitute unique identifier
- What changes cause event to be recorded

Auditing - Getting Started

Enabling auditing

- Disabled by default
- Upgrade client & database to 10.1A
- Create storage area(s) for audit data
  - Must be Type II storage area
- Enable auditing

```
Proutil dbname –C enableauditing area Data_Area [indexarea
Index_Area] [deactivateidx]
```


Defining Auditing Storage Areas

Example structure file

```
d "Audit_Data":20,32;512      .       f 40960
d "Audit_Data":20,32;512      .
#
d "Audit_Index":21,1;64       .       f 5120
d "Audit_Index":21,1;64       .
```

```
Prostrct add dbname <structurefile.st>
```


Auditing - Getting Started

Policies

- Connect to database as the DBA
- Set up database security key via Data Administration tool
- Edit audit permissions for users
  - Not tied to _User
- Optionally load / enable shipped policies
- Create your own events and policies

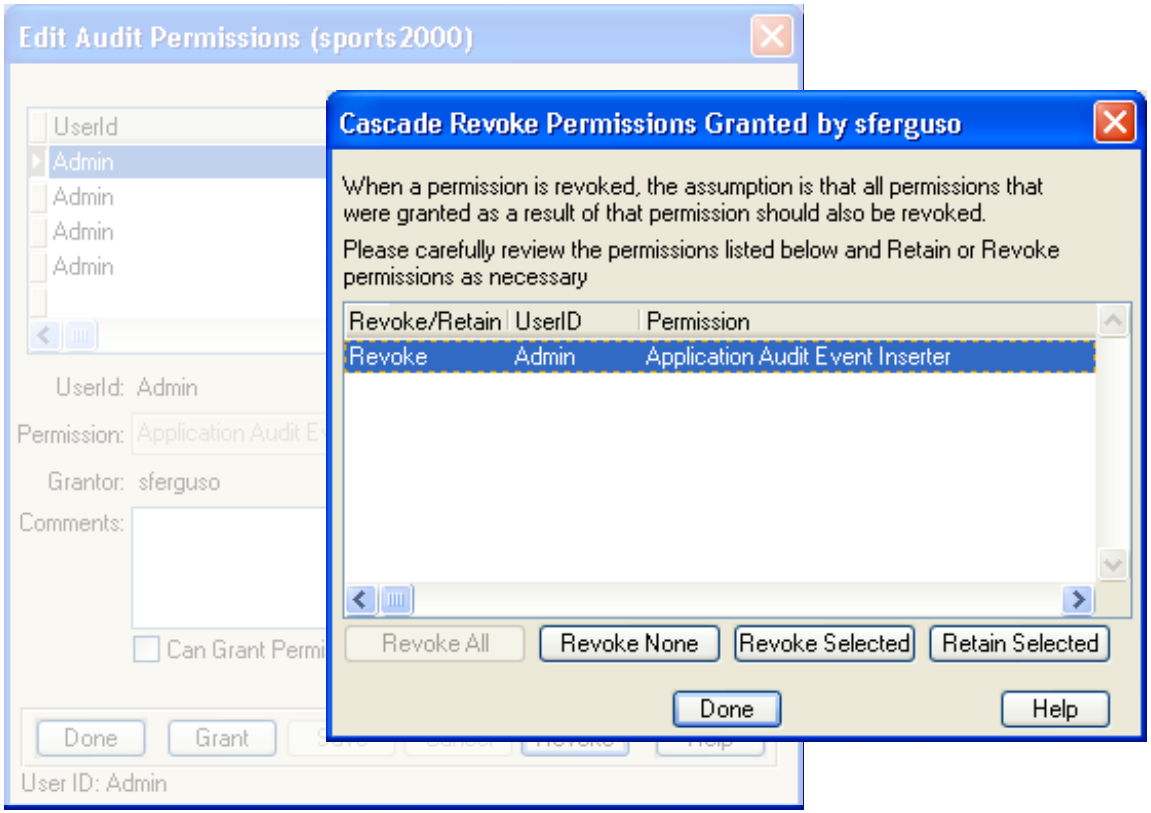
Create Audit Users

Separation of Duty

User	Description
Audit Administrator	Manage audit policies Grant auditing privileges
Audit Event Inserter	Can generate application audit events
Audit Data Archiver	Can archive & load audit data
Audit Data Reporter	Query and report on audit data

Manage Audit Permissions

Admin -> Security -> Edit Audit Permissions…



Auditing – Getting Started

Disabling auditing

- Disabling auditing

```
Proutil dbname –C disableauditing
```

- Does not remove anything
  - Policies, data, schema all remain
- Must be audit admin to disable
  - Event is audited

Getting Started – Lab 1.

Agenda

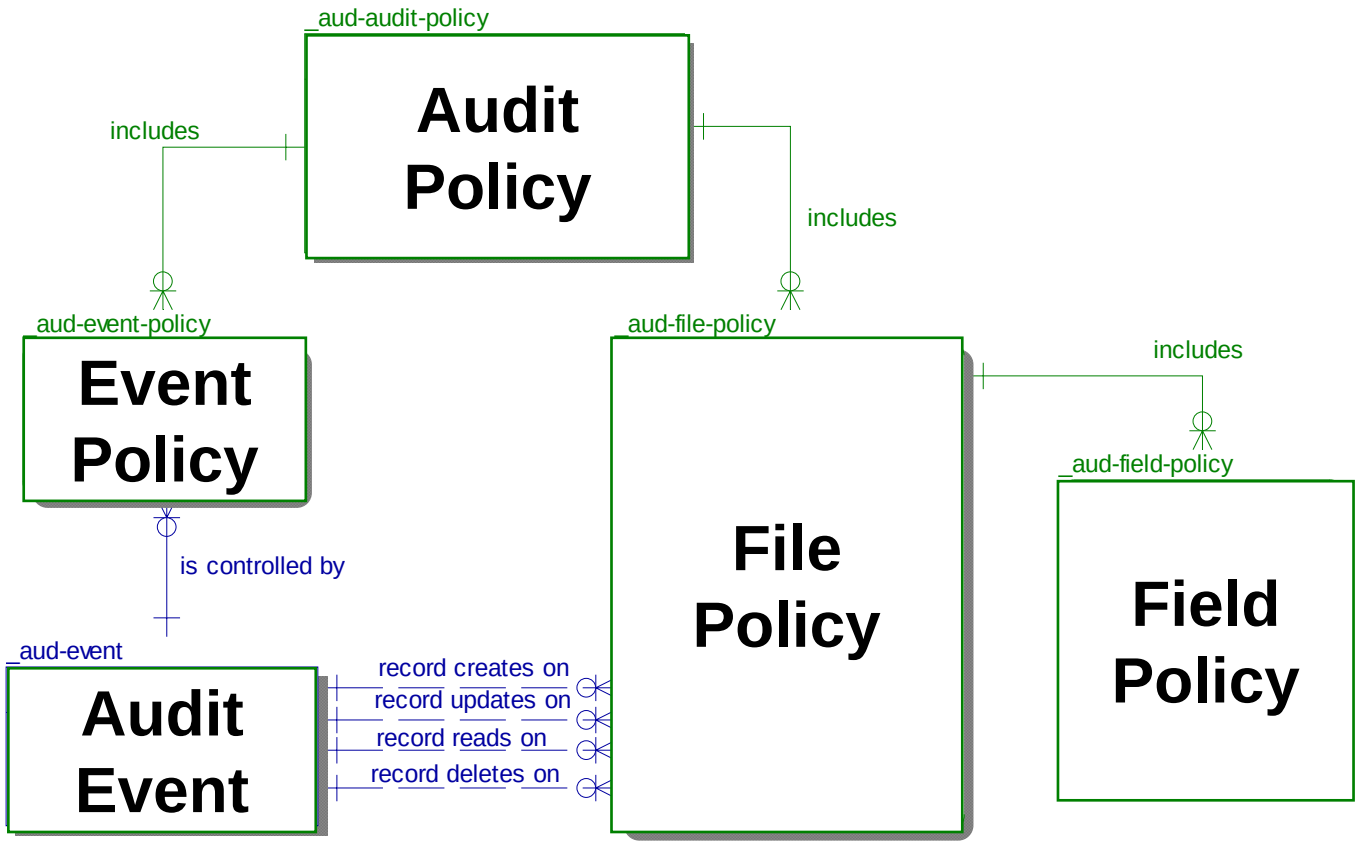
- Overview
- Getting started
- Audit Policy Maintenance
- Authentication
- Database Events
- Application Events
- Internal Events

Audit Policies

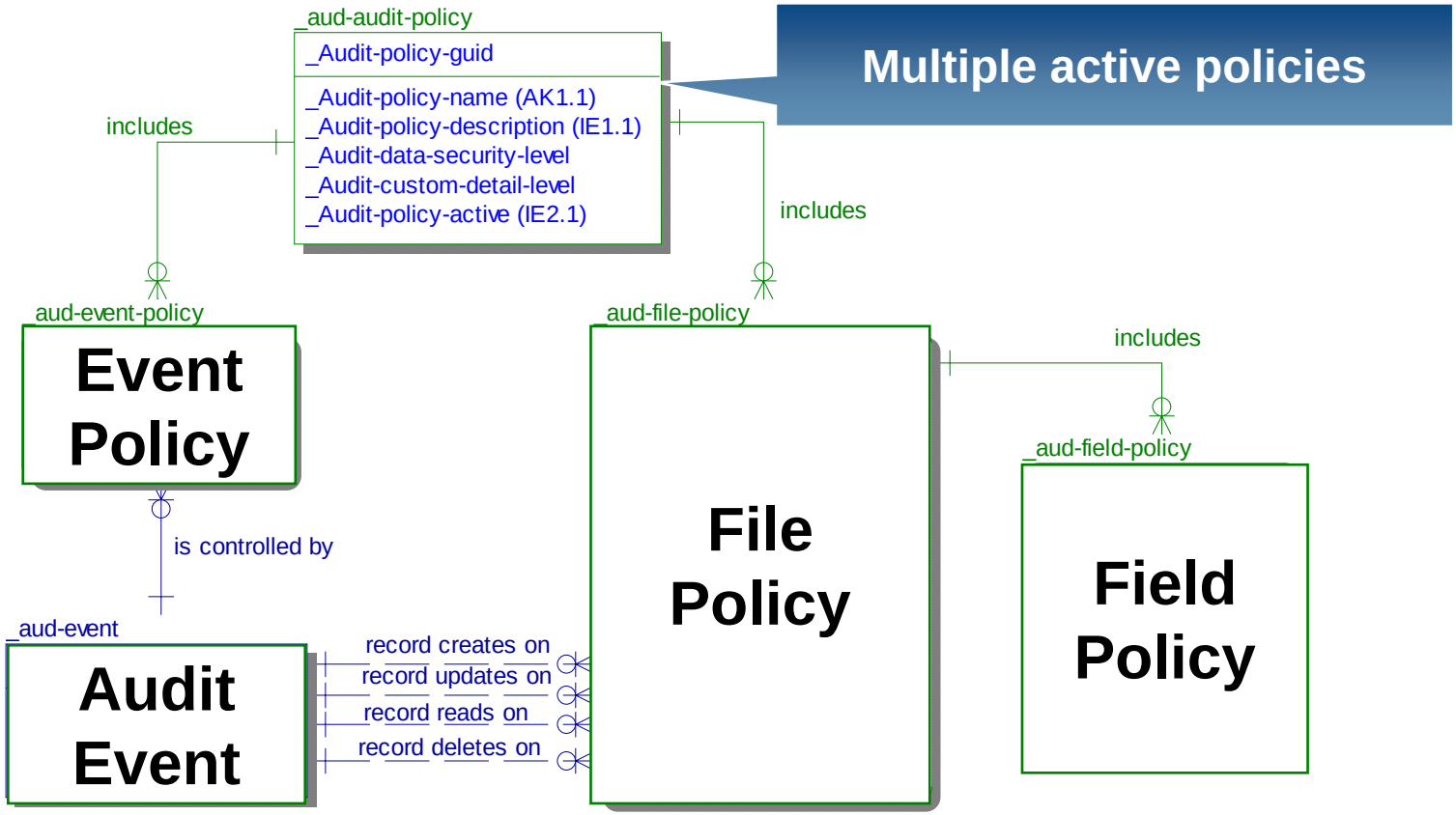
Definition

- An Audit Policy is
  - A named collection of audit configuration settings
  - Required for all audit operations
    - Database, application and internal
  - Applied at run time
- Multiple audit policies are supported
- Activate/deactivate required policies
- Manage event records

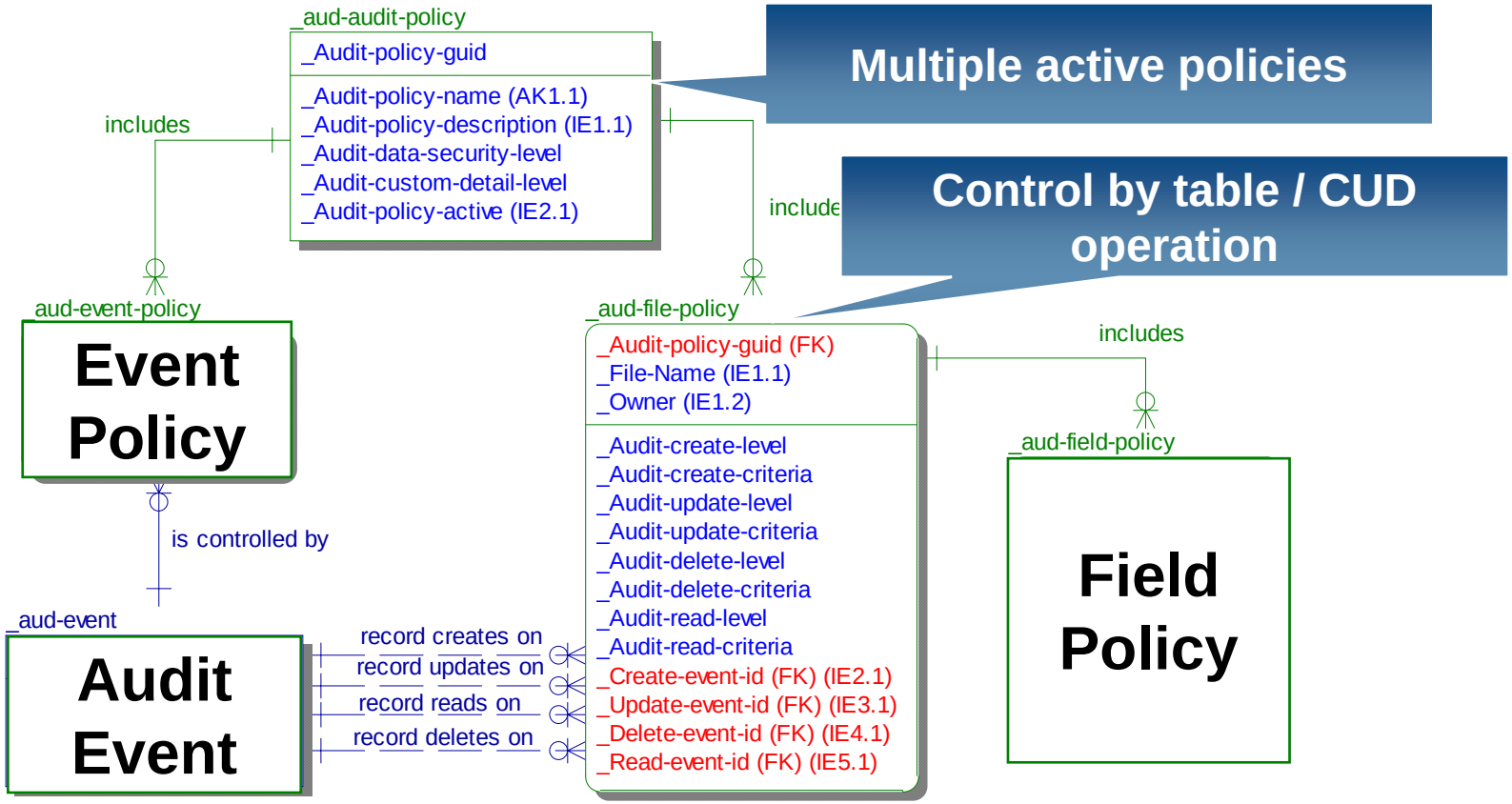
Audit Policy MetaSchema



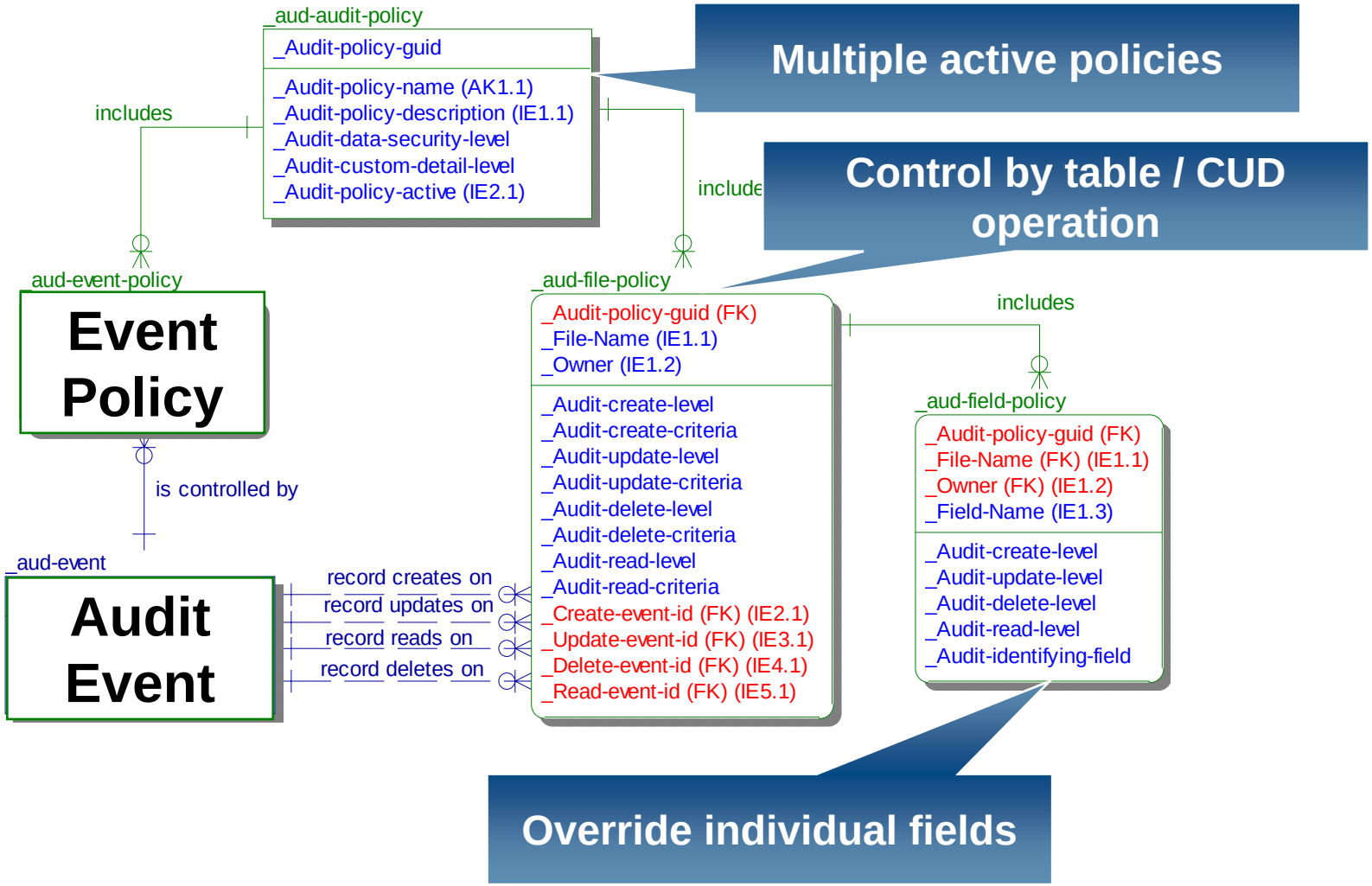
Audit Policy MetaSchema



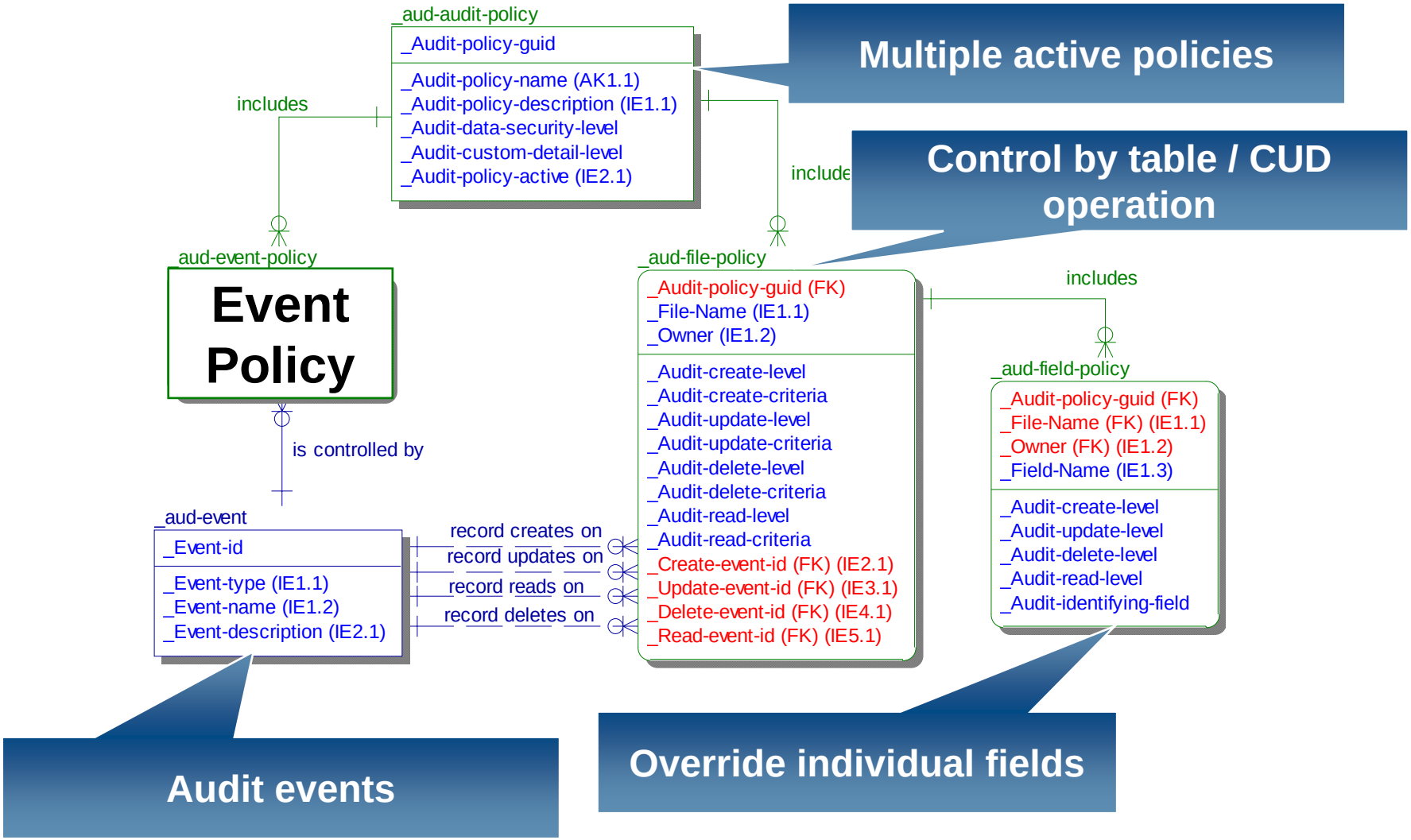
Audit Policy MetaSchema



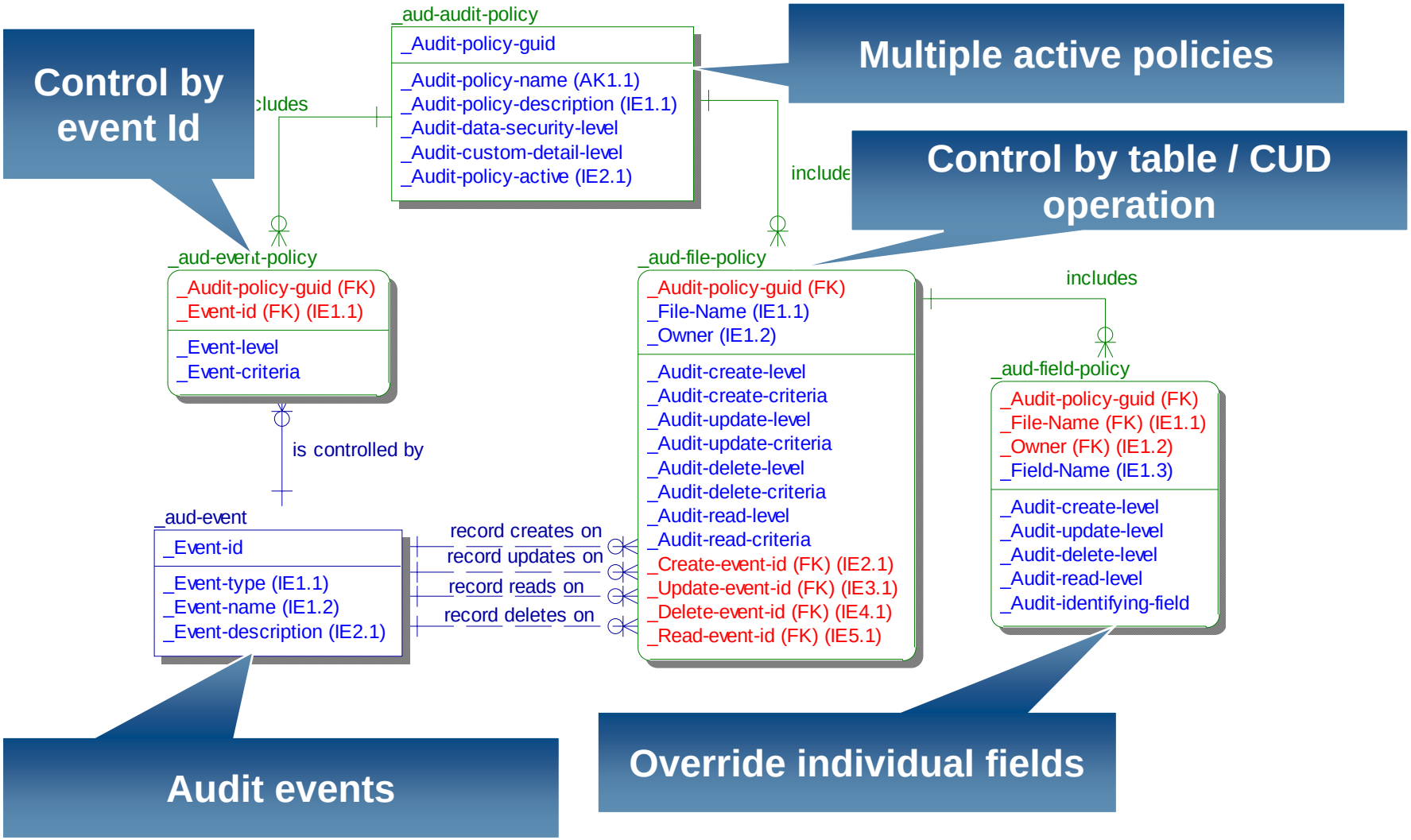
Audit Policy MetaSchema



Audit Policy MetaSchema



Audit Policy MetaSchema

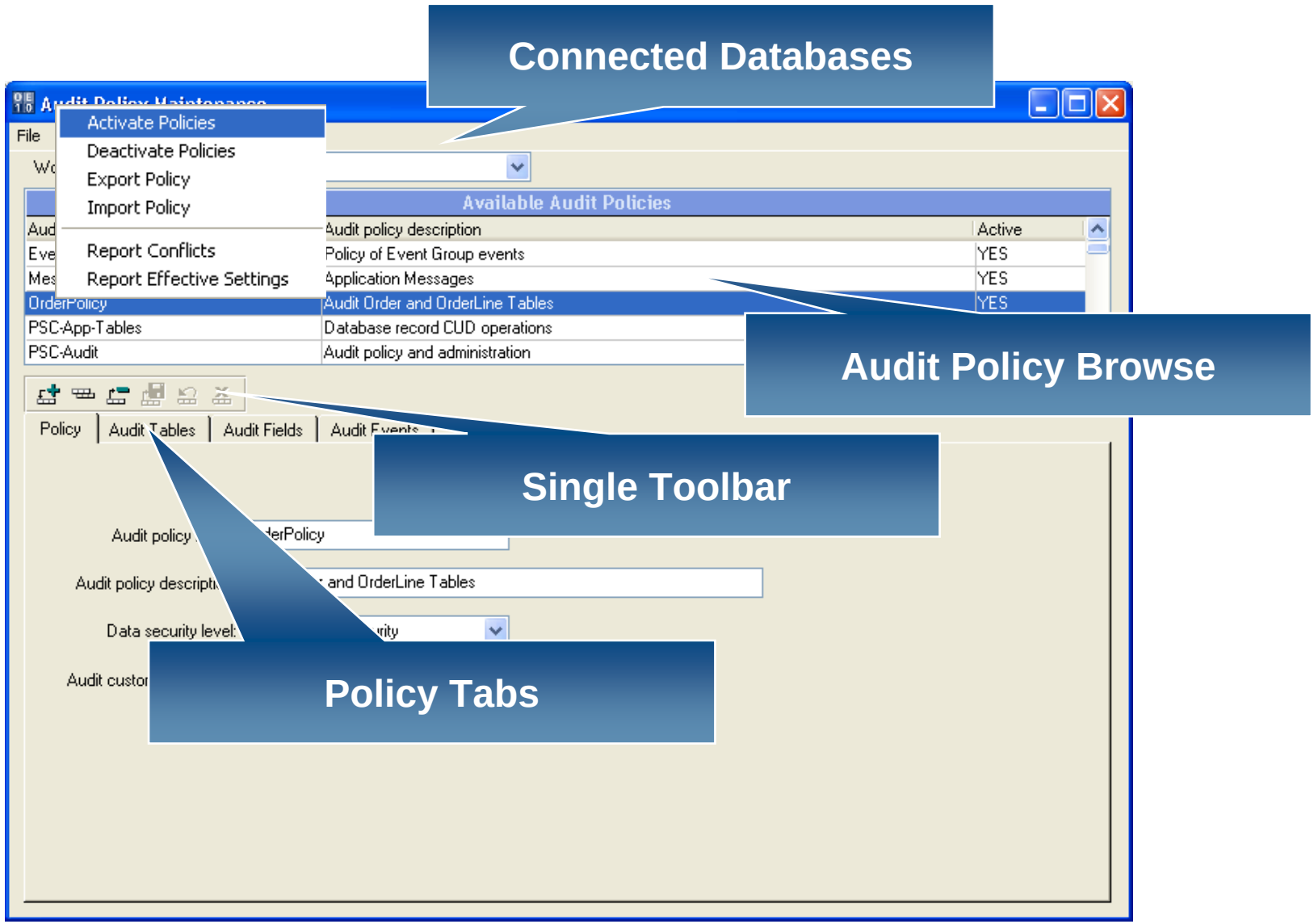


Audit Policy Maintenance

Primarily a developers tool

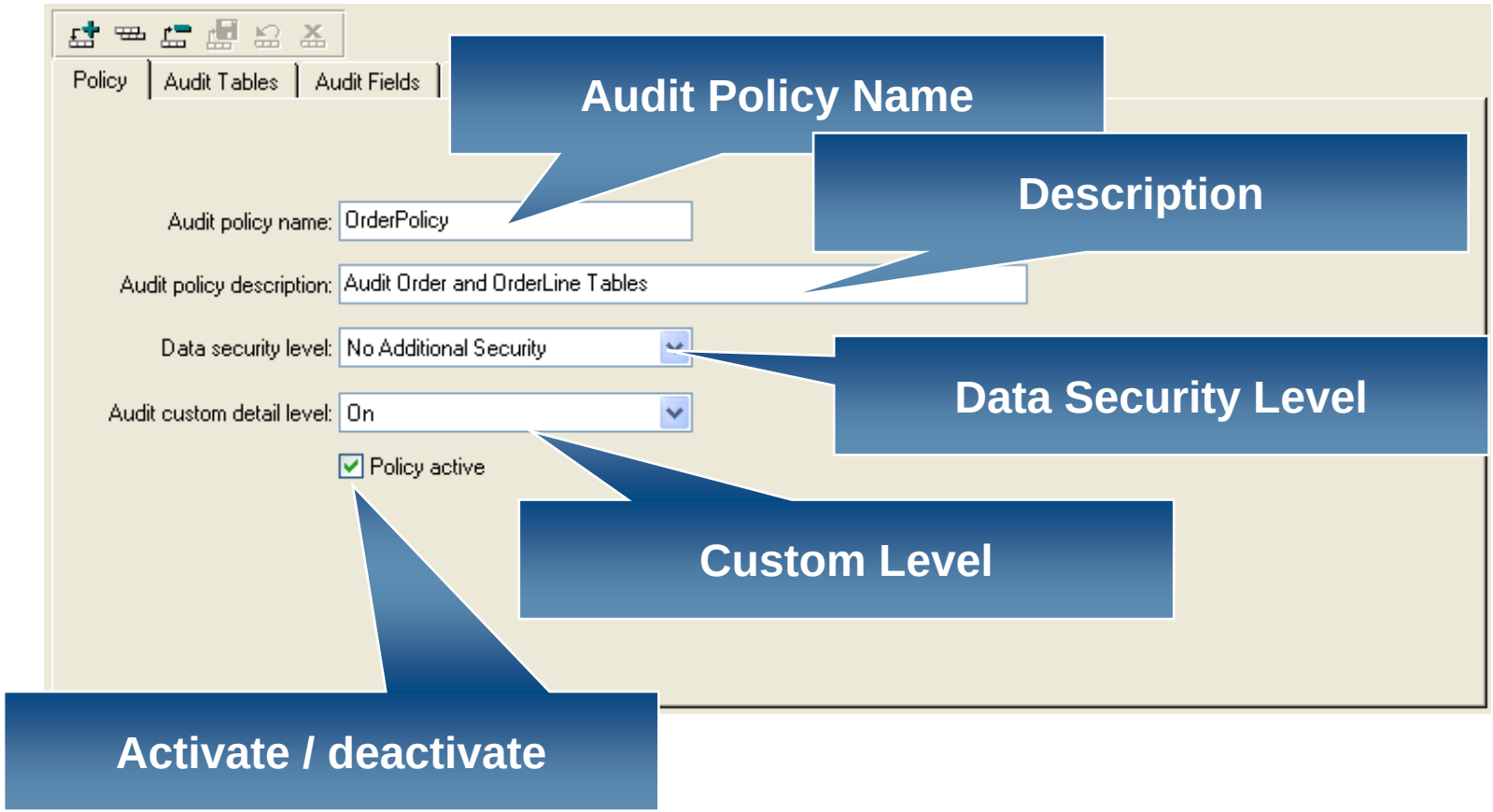
- Provides basic functionality
- A starting point to build your own
  - Source code is provided
  - Re-write as required
  - APIs provided
- Not translated
- Located in “DLC/auditing” directory
- Independent of other OpenEdge tools

Audit Policy Maintenance



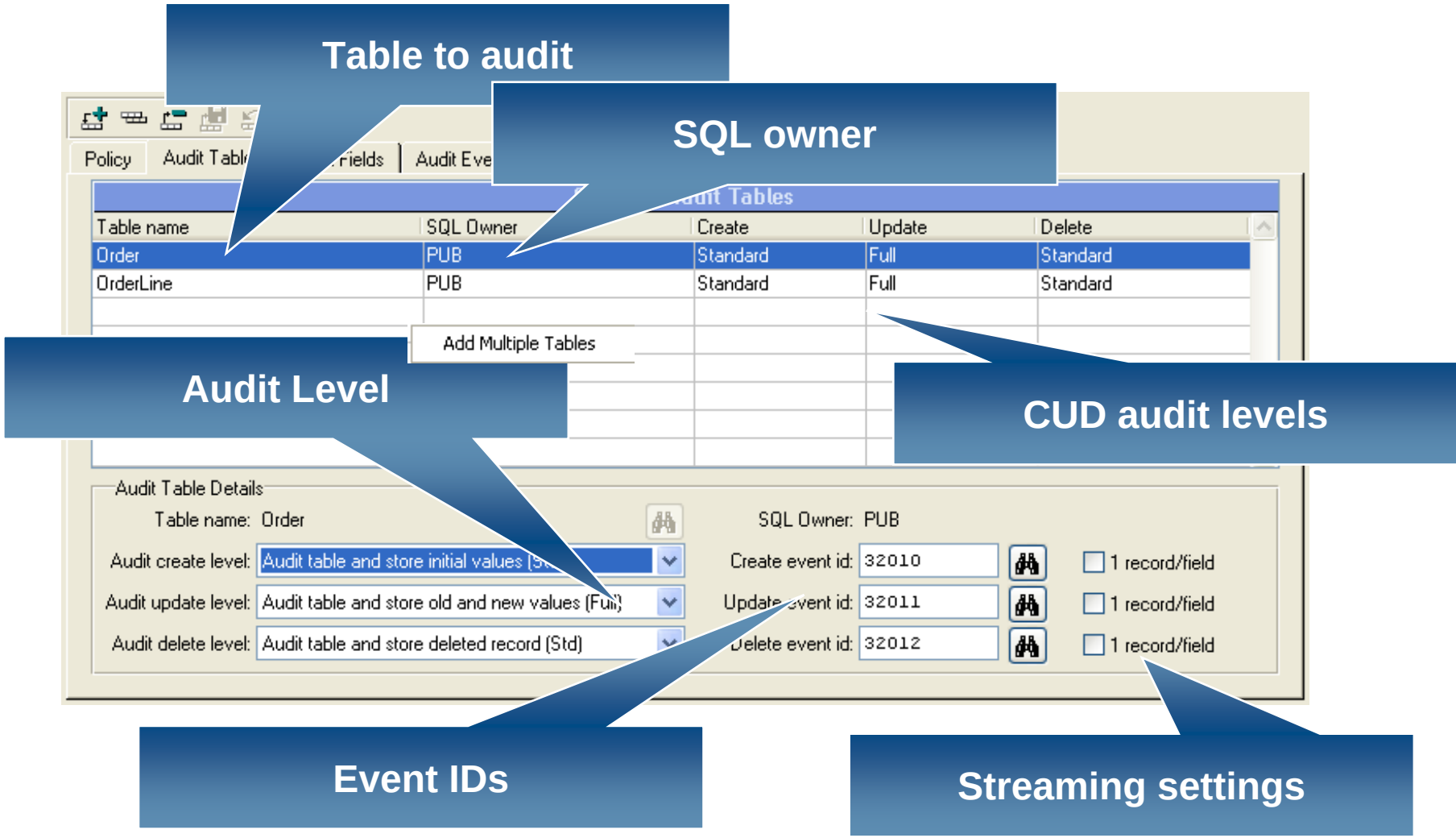
Audit Policy Maintenance - Policy Tab

Create, update, delete policy



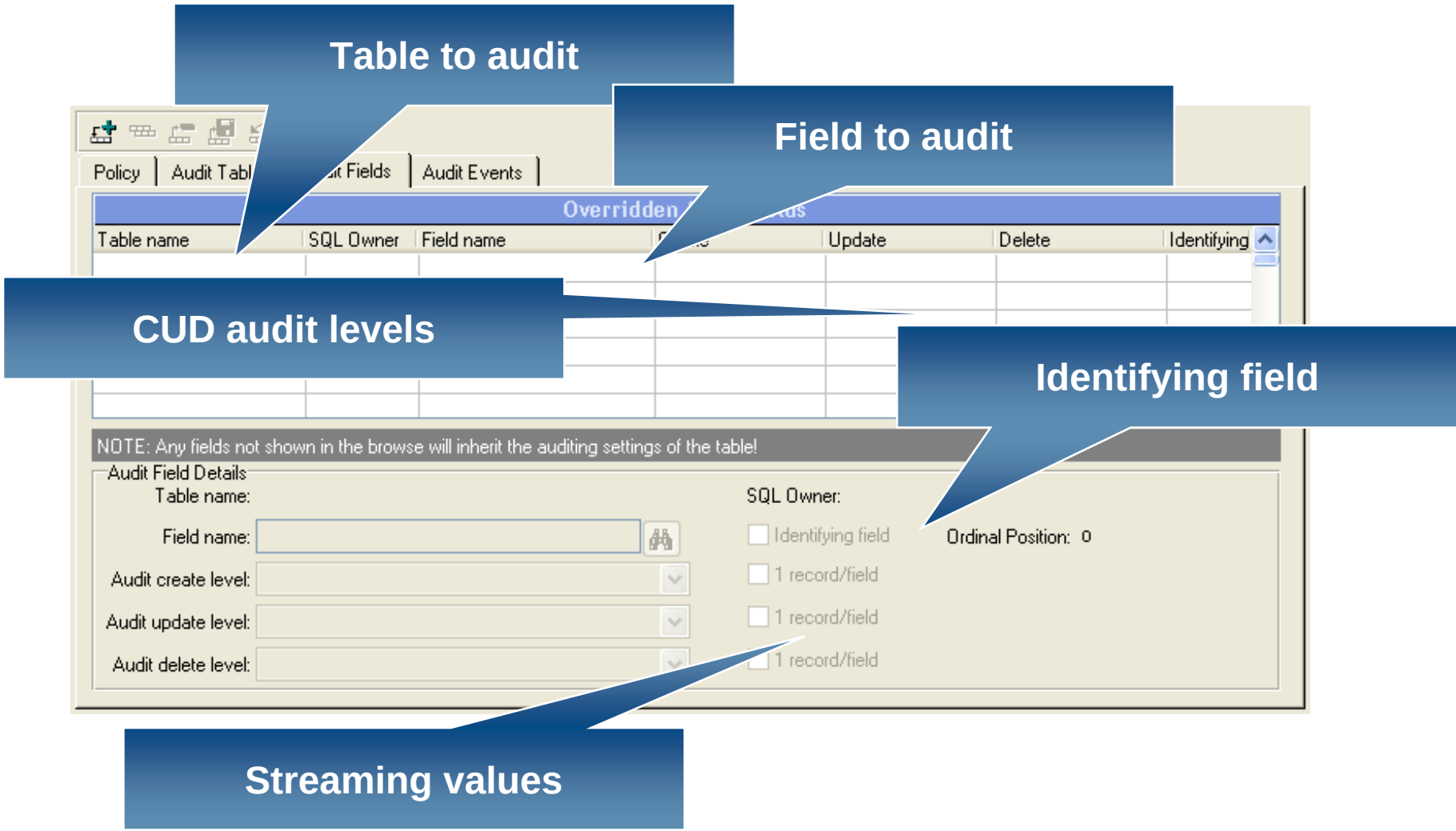
Audit Policy Maintenance - Audit Tables Tab

View, configure auditing for tables



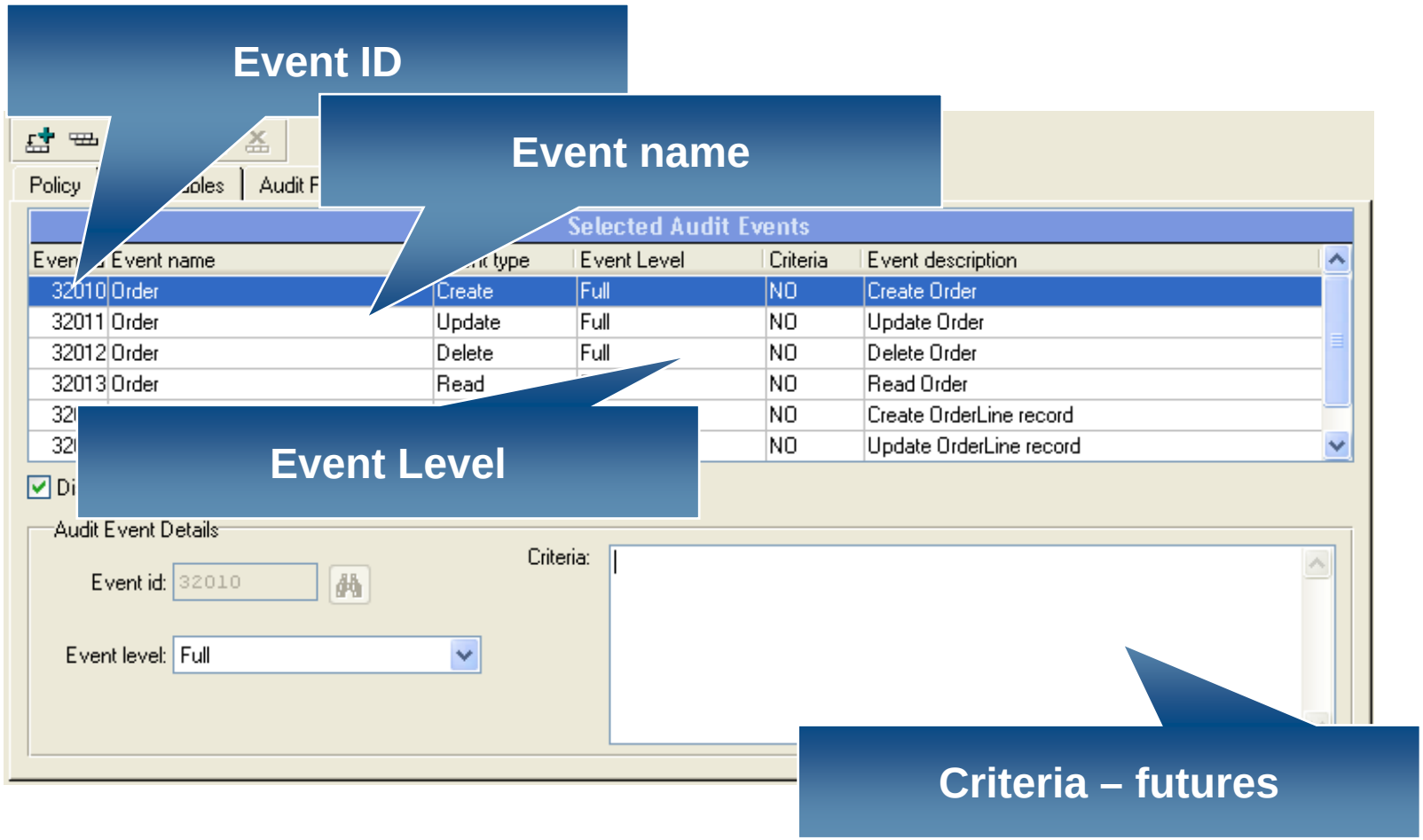
Audit Policy Maintenance - Audit Fields Tab

Field level auditing – overrides table settings



Audit Policy Maintenance - Audit Events

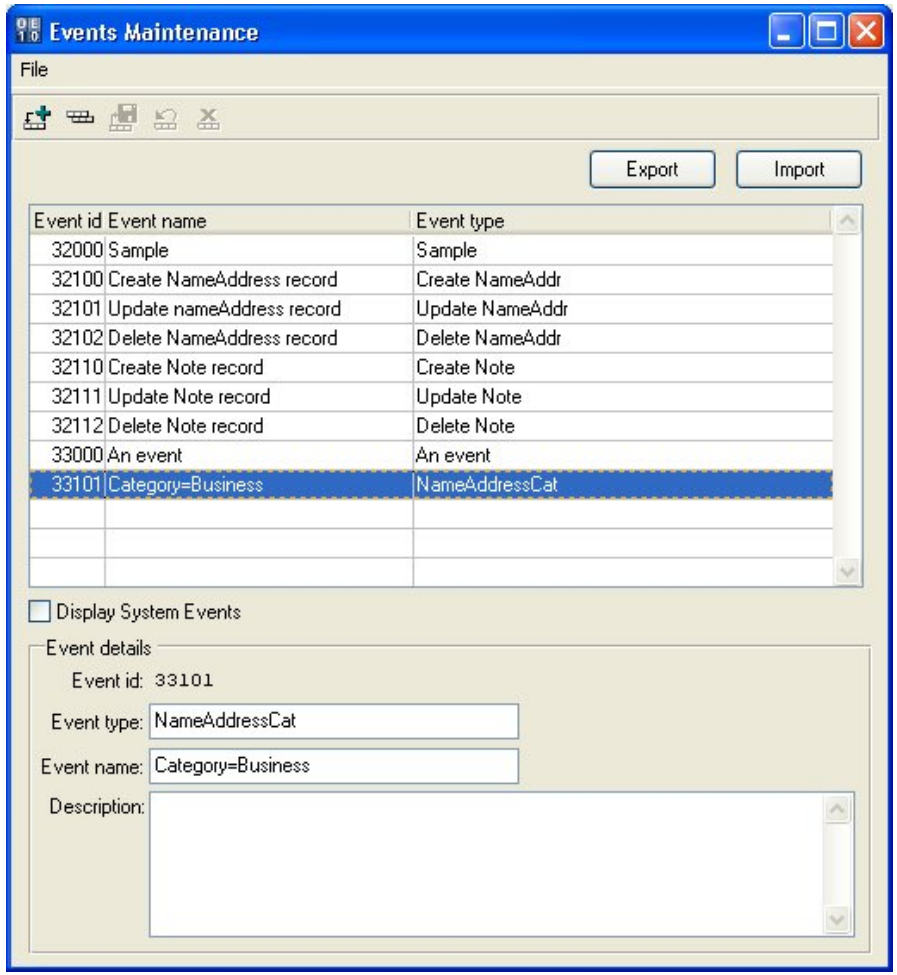
Event level auditing



Audit Policy Maintenance
Events Maintenance

File -> Events Maintenance…

- Cannot be deleted
- Can be renamed
- Copy allowed
- Changes committed on Save
- Cannot edit events below 32000



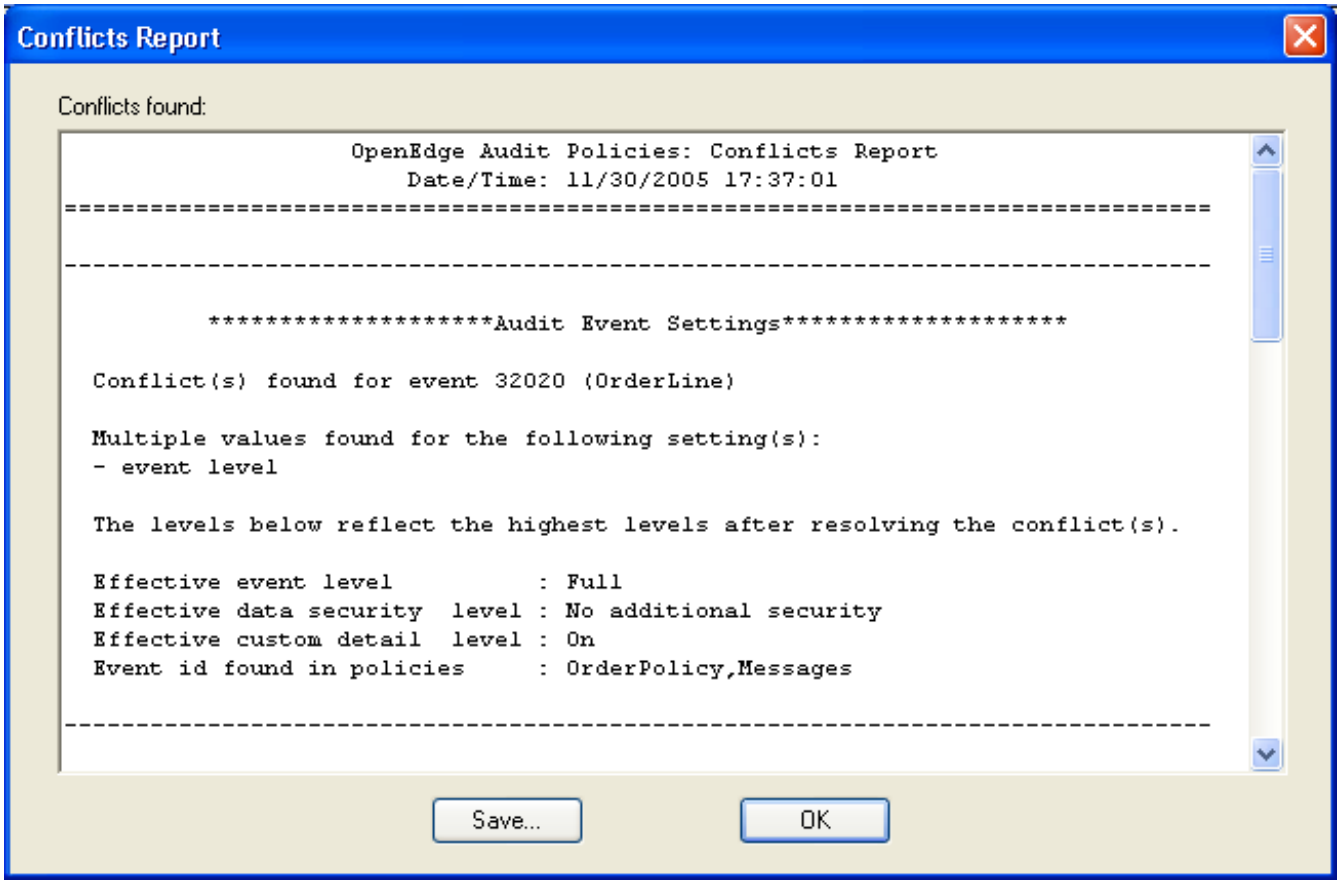
Audit Policy Maintenance

Additional features

- Import / export policies
  - As XML or dump files
- Import / export events
  - User defined events
- Also available from Data Admin tool
  - Supports multi-selection
- Use Audit Policy Maintenance API's to automate

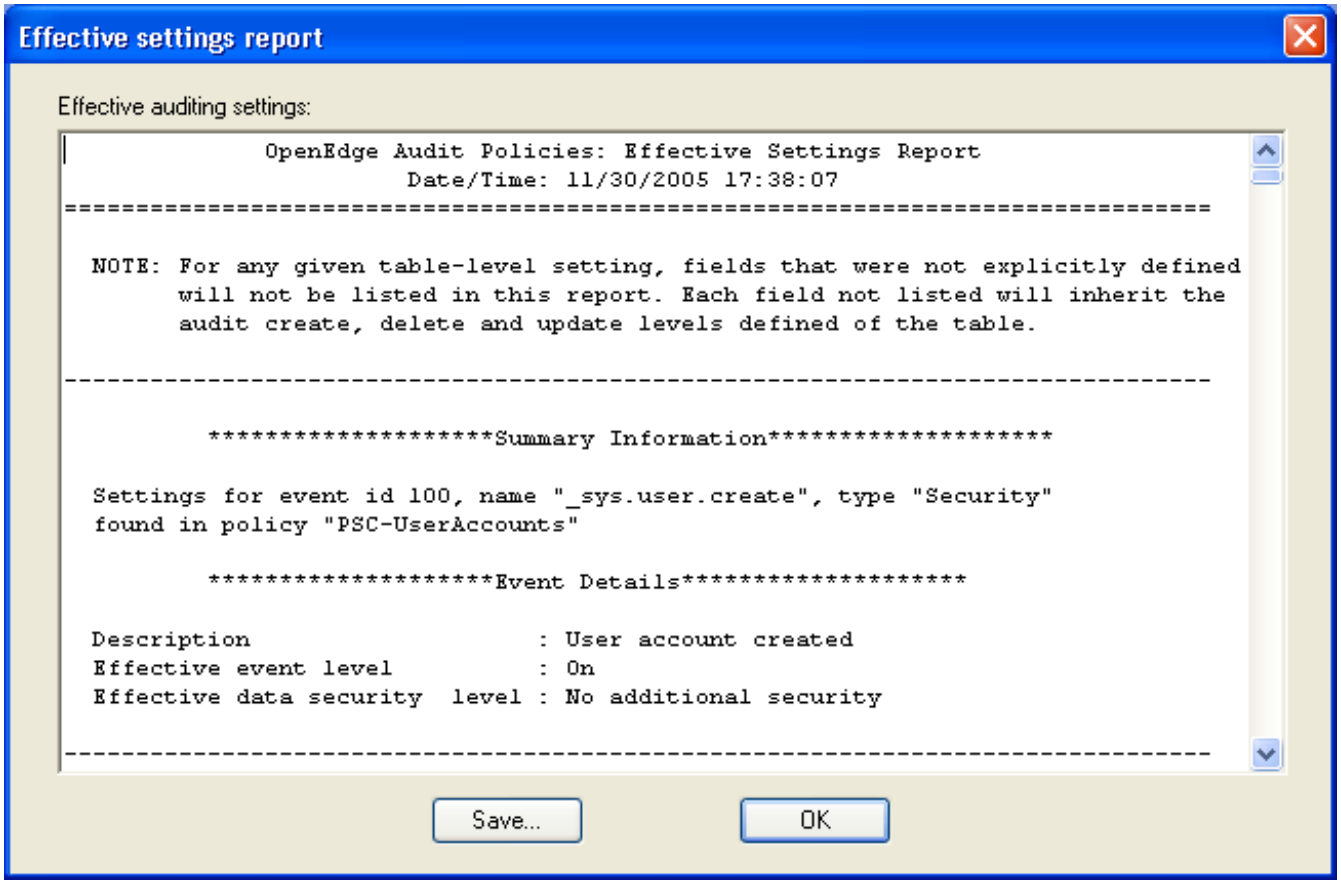
Conflict Reporting

Policy -> Report Conflicts



Report Effective Settings

Policy -> Report Effective Settings



Policy Updates At Runtime

- Audit policies are cached in memory
- Updates cause a rebuild
  - Database is stalled until rebuild complete

Audit Policy Maintenance – Demo. Lab 2.

Agenda

- Overview
- Getting started
- Audit Policy Maintenance
- Authentication
- Database Events
- Application Events
- Internal Events

The OpenEdge User Identity Challenge

Prior to 10.1A

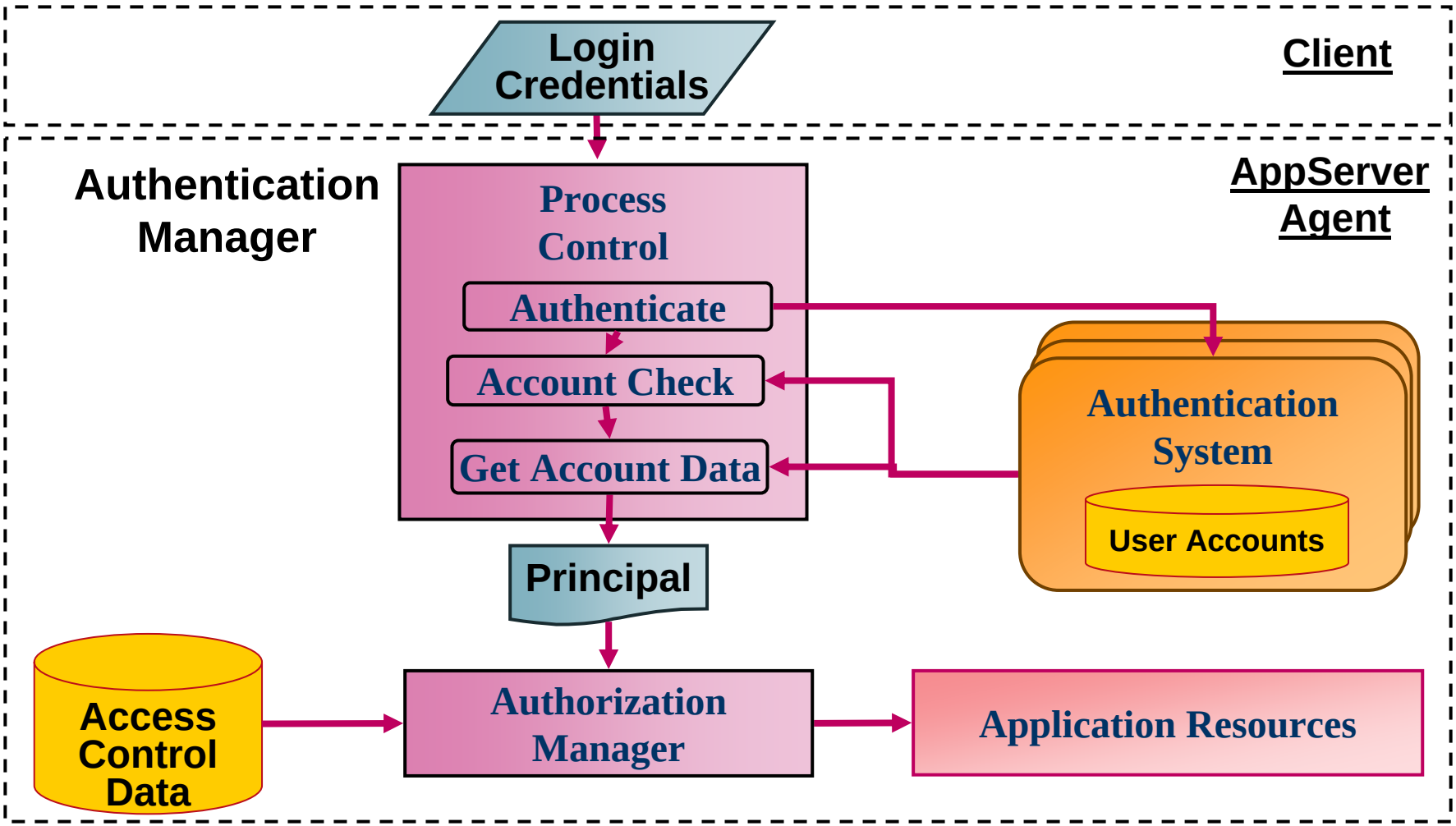
- _User table is the only *trusted* user-id source
- Almost no 4GL applications use the _User table
  - No way for 4GL application to tell OpenEdge that it is a trusted authentication source
  - No way for OpenEdge to validate that a user-id came from a *trusted* 4GL application source
- Solution: allow a 4GL application to become a *trusted* source of user authentication

10.1A - What Has Not Changed…

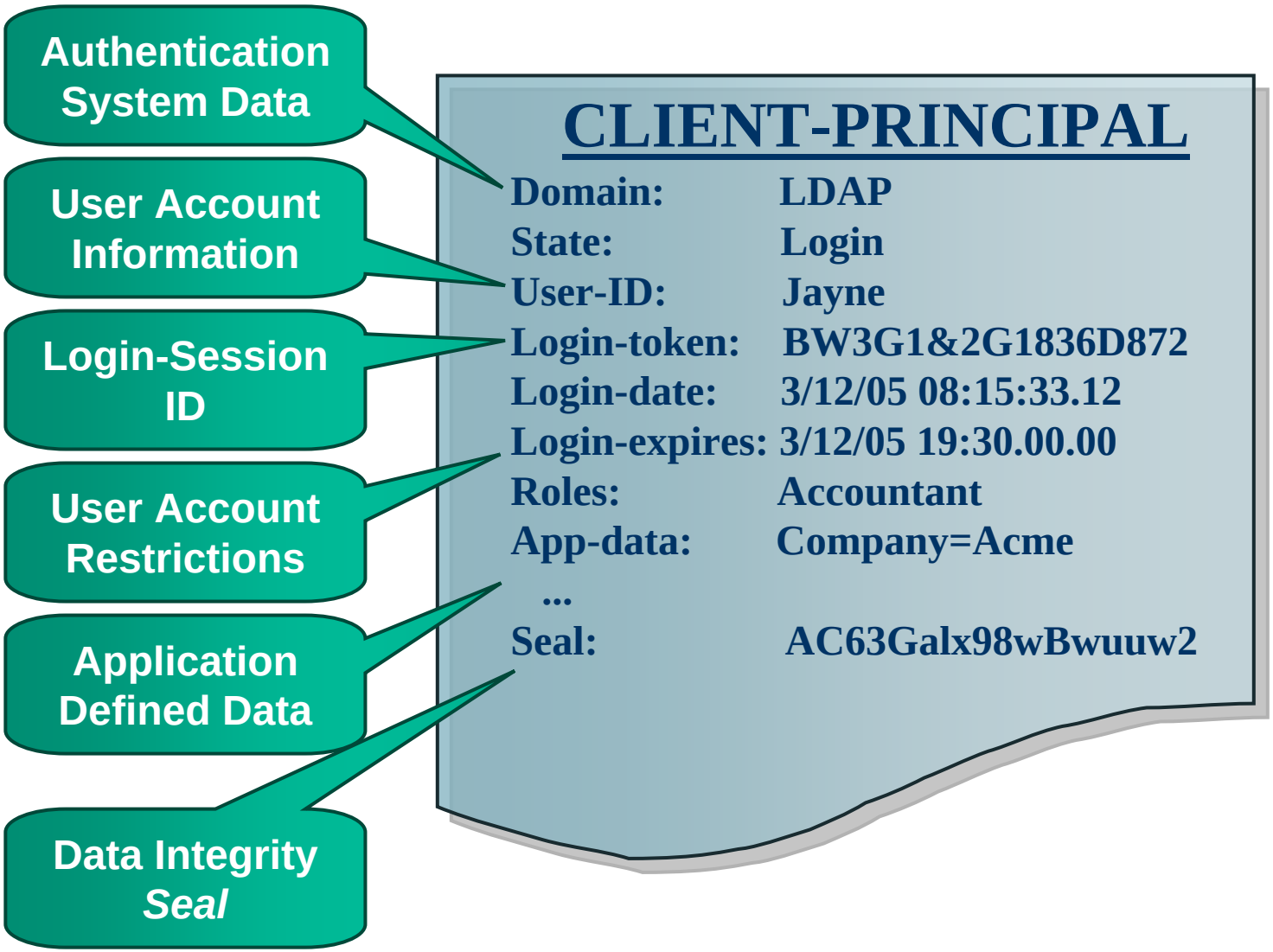
- Can still connect to OpenEdge database using –U –P
  - OpenEdge will require the _User table **
- SETUSERID() **
  - Authenticate and set the user-id for a database connection
- OpenEdge SQL requires using the _User table

**** Audited by OpenEdge auditing service**

Authentication and Authorization Process



The Principal



New OpenEdge 10.1A Features

- 4GL-session can have a default user-id
- CLIENT-PRINCIPAL 4GL object
- Secure client identity validation and auditing options
- Trusted Authentication Registry
- 4GL Language extensions
- AUDIT-CONTROL 4GL session handle
- AUDIT-POLICY 4GL session handle

4GL CLIENT-PRINCIPAL Object

- Created and managed by 4GL application
  - After user account has been authenticated
- Represents a single user login session
- Can be shared for single sign-on purposes
  - Between application servers
  - Between application server agents
  - Transport cross-platform binary value
- Set the current user-id for
  - The 4GL application [& all database connections]
  - Individual OpenEdge database connection
- Automatically audits login-logout operations
- CLIENT-PRINCIPAL’s user-id can be used for run-time permission checking

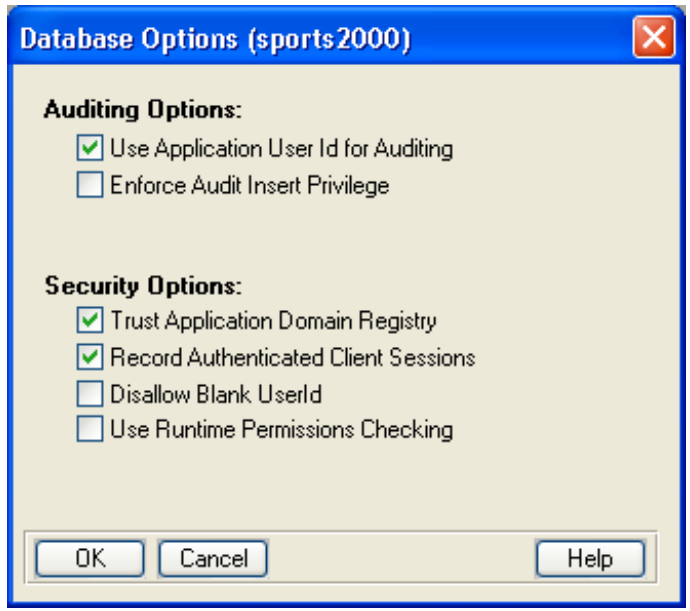
Trusted Authentication System Registry

- Used to validate CLIENT-PRINCIPAL object
  - Originating from trusted 4GL user authentication module
  - Checks integrity of user identity data
  - Validation uses symmetric key cryptography and HMAC technologies
- Contents loaded from
  - Application code using SECURITY-POLICY object
  - OpenEdge database tables
    - _sec-authentication-system
    - _sec-authentication-domain

User Identity Configuration Options

Database options

- Stored in _db-option table
- Used by 10.1A+ 4GL OpenEdge clients
- Managed via Data Administration tool



4GL Language Extensions

- SECURITY-POLICY object extensions
  - SET-CLIENT (hClientPrincipal).
  - LOAD-DOMAINS (dbAlias).
  - REGISTER-DOMAIN (“domain-name”, …).
    - LOCK-REGISTRATION ().

Auditing User-id Strategies

- Custom application design & implementation
- OpenEdge auditing service
  - Use SETUSERID() to built-in _User table
    - 👍 No changes needed if already in use
    - 👍 Can use AUDIT-CONTROL object
    - 👍 No extra configuration and deployment setup
    - 👎 No user login-logout or session information
    - 👎 Replicate _User table for multiple databases
  - Use 10.1A CLIENT-PRINCIPAL identity extensions
    - 👍 Use existing 4GL authentication modules
    - 👍 User login-logout and session information
    - 👍 Single sign-on between 4GL products
    - 👎 Requires code additions
    - 👎 Extra configuration and deployment setup

User Identity Strategies

Steps

- Define and deploy application supported user authentication system types and domains
  - _sec-authentication-system table
    Ex: 4GL procedure, LDAP, Kerberos, …
  - _sec-authentication-domain table
    Ex: Built-in, Default-LDAP, Default-Kerberos, …
- Configure/enable domains at production site
- Define and deploy user identity and validation options
  - Data Administration

Load / Register Domains

Example

```
FOR EACH trusted-reg NO-LOCK:
        SECURITY-POLICY:REGISTER-DOMAIN
        (trusted-reg.cDomainName,
         trusted-reg.cDomainKey,
         trusted-reg.cDomainDescr,
         trusted-reg.cDomainType).

END.
SECURITY-POLICY:LOCK-REGISTRATION.
```

```
LOAD-DOMAINS  ("Sports2000").
```


Creating a CLIENT-PRINCIPAL Object

```
CREATE CLIENT-PRINCIPAL hCp.
hCp:USER-ID = cUserId.
hCp:DOMAIN-NAME = cDomainName.
IF (SESSION:REMOTE) THEN DO:
    IF (SESSION:SERVER-OPERATING-MODE = "State-free") THEN DO:
            cSessionID = BASE64-ENCODE(GENERATE-UUID).
            hCp:SESSION-ID  = SUBSTRING(cSessionID, 1, 22).
        END. ELSE DO:
              hCp:SESSION-ID = SESSION:SERVER-CONNECTION-ID.
        END.
    END. ELSE DO:
         cSessionID = BASE64-ENCODE(GENERATE-UUID).
        hCp:SESSION-ID  = SUBSTRING(cSessionID, 1, 22).
    END.
END.
hCp:AUDIT-EVENT-CONTEXT =  cUserId + "@" + cDomainName.
hCp:CLIENT-TTY = SESSION:CLIENT-TYPE + "." +
        SESSION:DISPLAY-TYPE.
```

Insert user & domain

Insert login session ID

Insert optional information

Insert Audit-Event context

Managing a CLIENT-PRINCIPAL Object

- Successful user authentication

```
lRetValue = p_hCP:SEAL(“Gy23800xxYzthslie4yyslzekeylqql”).
```

- Unsuccessful user authentication

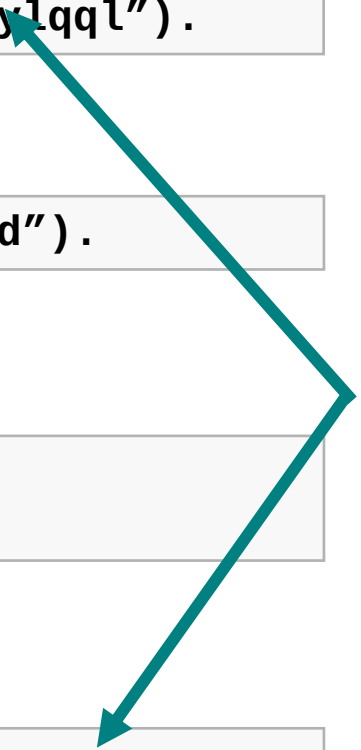
```
m_hCP:AUTHENTICATION-FAILED(“User account not found”).
```

- Ending the user’s login session

```
m_hCP:LOGOUT.
DELETE OBJECT m_hCP.
```

- Validating the CLIENT-PRINCIPAL

```
lRetValue = p_hCP:VALIDATE-SEAL
       (“Gy23800xxYzthslie4yyslzekeylqql”).
```



Moving the CLIENT-PRINCIPAL

Import and Export

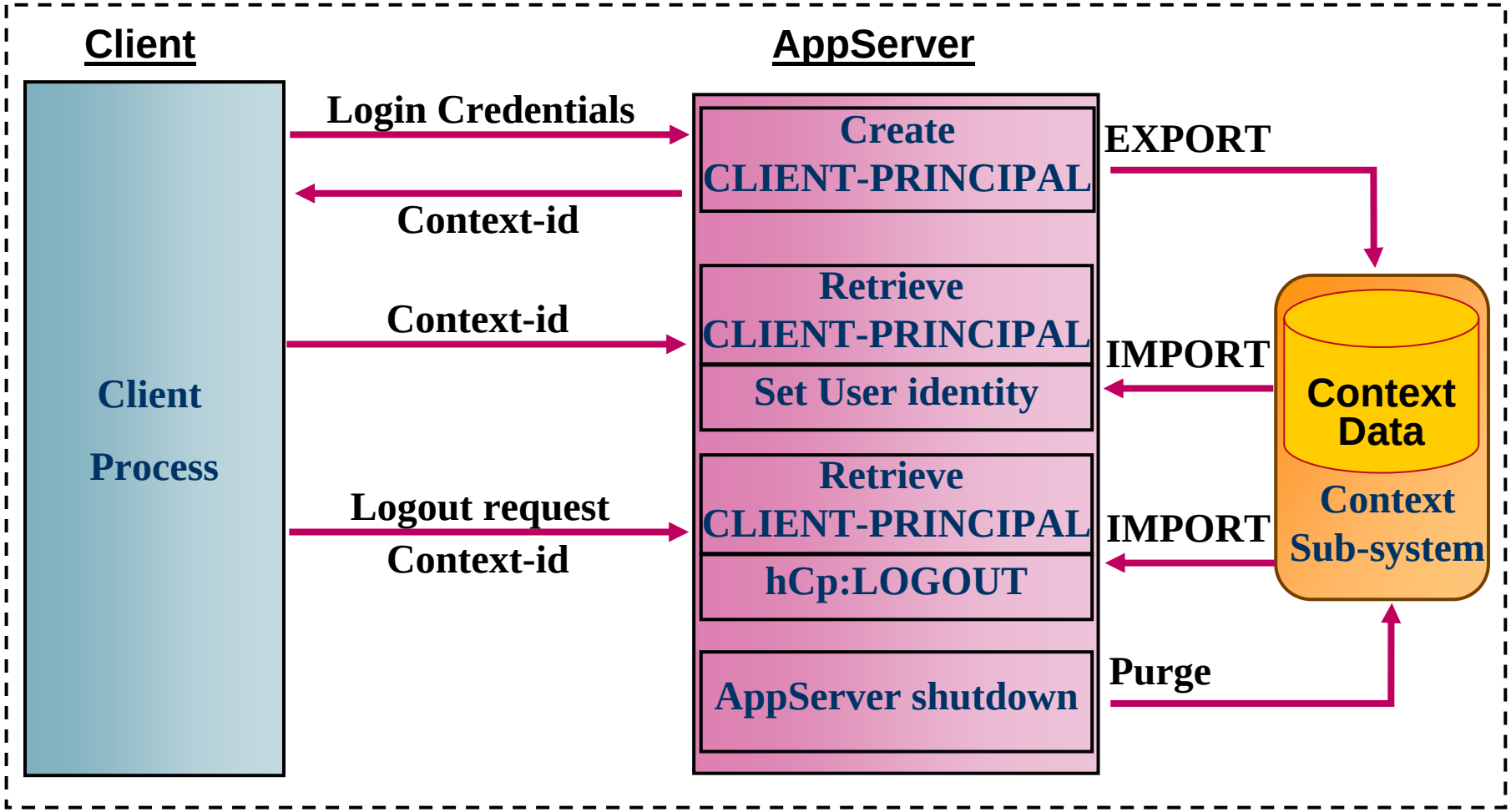
- Exporting for storage or transport

```
UsrCtxDB.rawClientPrincipal = m_hCP:EXPORT-PRINCIPAL.
```

- Importing from storage or transport

```
m_hCP:IMPORT-PRINCIPAL(UsrCtxDB.rawClientPrincipal).
```


AppServer Identity Management Tasks



Authentication – Demo. Lab3

Agenda

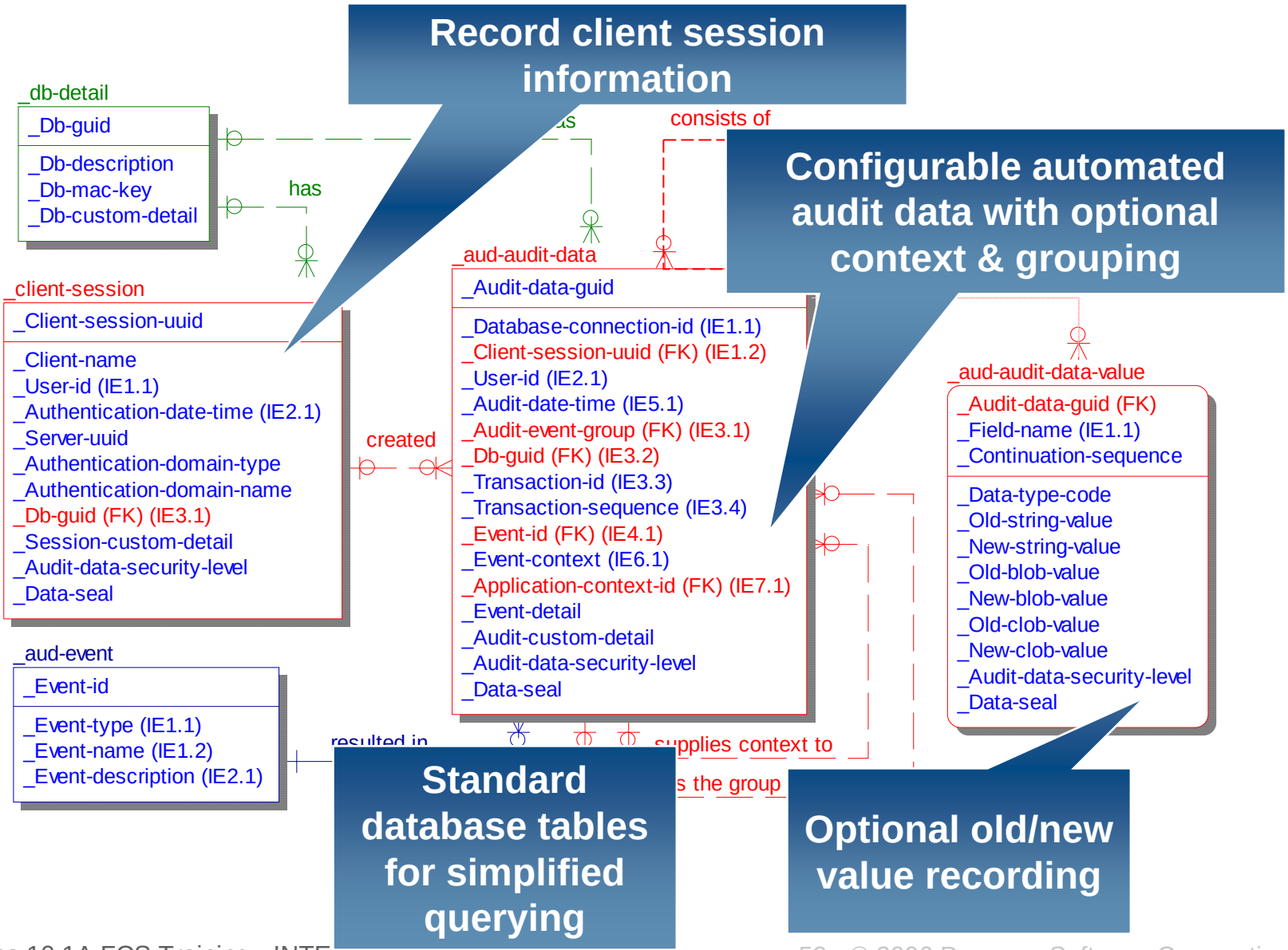
- Overview
- Getting started
- Audit Policy Maintenance
- Authentication
- Database Events
- Application Events
- Internal Events

Database events

What gets Audited?

- Record level events
  - Create
  - Update
  - Delete
- Controlled through file / field policy
- Old/New values
  - Stored as character
  - American format dates and numeric values

Audit Data Schema



Overridden Audit Fields

- File level policy is the default for fields
  - Set according to majority of fields
- Individual fields may be overridden
- When explicitly auditing fields
  - Consider schema changes

Field Value Recording

Performance vs. field reporting

- One record per field
  - Easy to report on individual field changes
  - Resource intensive
- Streamed
  - Pack as many field values into a single audit record
  - Reduced number of database writes

Streamed Field Values

- Values stored in _aud-audit-data
  - _Event-detail field
- Character format

```
field-name + chr(6) + data-type + chr(6) +
[old-value] + chr(6) + new-value + chr(7)[…]
```

  - chr(8) delimits array elements
- Must be enough space for field value
  - Otherwise written to _aud-audit-data-value
- Order of fields is arbitrary

Reporting on Streamed Field Values

- Unlikely to be able to use SQL reports
  - Unless specifically coded for OpenEdge auditing records
- 4GL reports must deal with streamed data
  - Arbitrary order
- Use a ProDataSet for queries and parse streamed field values

Streamed Values

Consider

- Store large CHARACTER and RAW fields individually
  - Maximizes smaller fields being compressed
- Reporting requirements
  - Individual fields

Old/New Field Values

- No old/new value support in 10.1A for LOBs
- There may be continuation records
  - Where _continuation-sequence > 0
- Arrays
  - Changed elements saved in same field
  - Each element is identified by “E[n]:”
  - Separator is chr(7)

```
chr(7)E[8]:AugustE[11]:November
```

Identifying Fields

Event context field _aud-audit-data._event-context

- Primary way of locating application audit data for specific *record* level events
  - E.g. for Customer number 10
- Default is primary index
- Avoid changing format over time
  - Makes it difficult to find records
- Stored in _aud-audit-data._event-context
- Format is

```
<owner>.<table>chr(6)<id-fld-val>[chr(7)<id-fld-val>.. ]
```


Agenda

- Overview
- Getting started
- Audit Policy Maintenance
- Authentication
- Database Events
- Application Events
- Internal Events

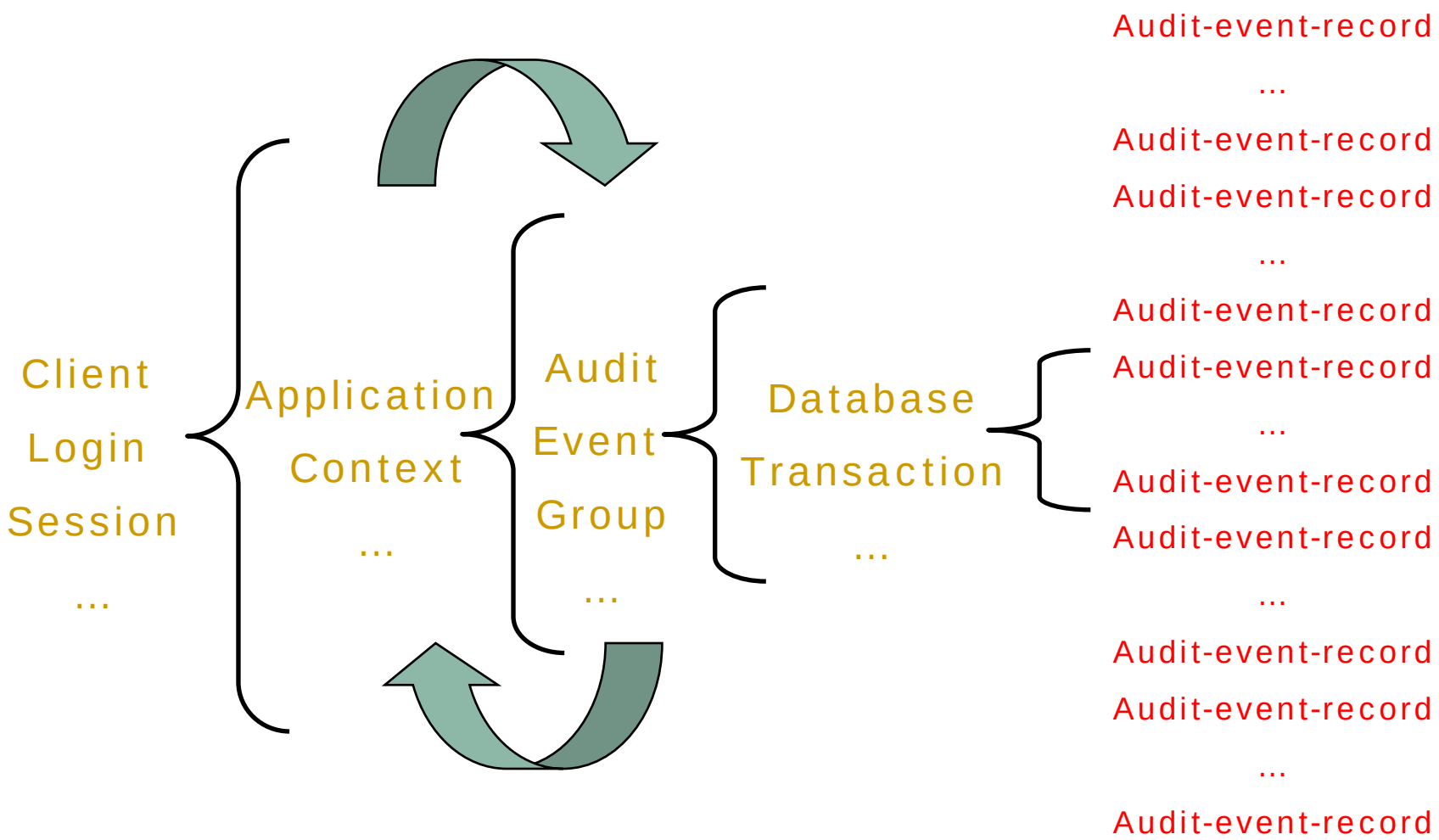
Application Defined Events

- Events with no corresponding database operation
- Context describes why the data was audited
  - Gives meaning to record level auditing
  - Event ID >= 32000
- Fully control granularity and detail
  - Example: 1 audit record for dispatch of an order
- Group into ranges to simplify reporting

Application Context

- Provides contextual information
  - When, where and why of changes
- Types of contextual information
  - Database transactions and sequence
  - Client login sessions
  - Application Context
  - Application Event Groups (AEG)

Types of Scope and Auditing Context



Log an Audit Event

AUDIT-CONTROL:LOG-AUDIT-EVENT method

- Creates an application defined event
  - In all audit-enabled databases with the event enabled
- A supporting active policy must exist
- Can write directly to the long-term storage
- Can be used for read auditing

Log Audit Event - Example

```
…
Ctx-id = AUDIT-CONTROL:LOG-AUDIT-EVENT
        (32530, "Starting Procedure: " +
        PROGRAM-NAME(1), cDetail, cUserData).
…
/* READ auditing */
Ctx-id = AUDIT-CONTROL:LOG-AUDIT-EVENT
    (32003, "Customer Enquiry",
     {&FIELDS-IN-FRAME-{&FRAME-NAME}}).
…
```


Set Application Context

AUDIT-CONTROL:SET-APPL-CONTEXT method

- Sets application context
  - Sent to all audit-enabled databases
- UUID used as context ID
  - Recorded with all subsequent audit events
    - _aud-audit-data. _application-context-id
- Event context cannot be unknown value
- Application context does not support nesting

Clearing Application Context

AUDIT-CONTROL:CLEAR-APPL-CONTEXT

- Clears an application context event-id
  - For all audit enabled databases
  - No context-id written in subsequent records
- No audit event generated

Application Context - Example

```
DEF VAR ctx-id as CHAR.


…
ctx-id = AUDIT-CONTROL:SET-APPL-CONTEXT
        (PROGRAM-NAME(1) + " Context",
        "Start Customer Enquiry Context").
…

AUDIT-CONTROL:CLEAR-APPL-CONTEXT.
```


Reporting on Event Context

AUDIT-CONTROL:SET-APPL-CONTEXT

- Application context record (parent)
  - Event ID = 31998
  - Unique guid in _Audit-data-guid
- Audit data records within context
  - Secondary read required
  - _Application-context-id = guid of parent
- Recursive join on _aud-audit-data

Audit Event Groups

AUDIT-CONTROL:BEGIN-EVENT-GROUP method

- Indicates beginning of a sequence of ‘batched’ operations
  - Sent to all audit-enabled databases
  - Can group multi-database transaction events
- UUID used as context ID
  - Recorded with all subsequent audit events
    - _aud-audit-data. _audit-event-group
- Cannot be nested
- Event context argument cannot be unknown value

End The Event Group

AUDIT-CONTROL:END-EVENT-GROUP method

- Ends an application event group
  - Sent to all audit-enabled databases
- Does not generate an event

```
Ctx-id = AUDIT-CONTROL:BEGIN-EVENT-GROUP
("Save Order Details-EVENT GROUP",
 "Data-set SAVE-ROW-CHANGES",
 cUserData).
…
AUDIT-CONTROL:END-EVENT-GROUP.
```


Reporting on Event Groups

AUDIT-CONTROL:BEGIN-EVENT-GROUP

- Event group record (parent)
  - Event ID = 31999
  - Unique guid in _Audit-data-guid
- Audit data records within context
  - Secondary read required
  - _Audit-event-group = guid of parent
- Recursive join on _aud-audit-data

OpenEdge SQL Application Auditing

- Log audit events

```
AUDIT INSERT ( event_id,
               [ event_context | NULL ],
               [ event_detail  | NULL ]);
```

- Set context and begin groups

```
AUDIT SET APPLICATION_CONTEXT | EVENT_GROUP
                [ Context | NULL ];
```

AUDIT-CONTROL System Handle

Attributes

- APPL-CONTEXT-ID
  - Returns UUID of current context

```
Ctx-id = AUDIT-CONTROL:APPL-CONTEXT-ID.
```

- EVENT-GROUP-ID
  - Returns UUID of current event group

```
Grp-id = AUDIT-CONTROL:EVENT-GROUP-ID.
```


Functions – AUDIT-ENABLED

- Queries audit-enabled status

```
logEnabled = AUDIT-ENABLED.
```

  - Returns YES if any connected db is audit enabled

```
logEnabled = AUDIT-ENABLED (logical-dbname).
```

  - Returns YES if *logical-dbname* is audit enabled

Functions – GENERATE-UUID

- Generates a Universal Unique Identifier

```
DEFINE VARIABLE rawUUID     AS RAW  NO-UNDO.
DEFINE VARIABLE cBase64UUID AS CHAR NO-UNDO.

DO WHILE TRUE:
   ASSIGN rawUUID = GENERATE-UUID
          cBase64UUID = BASE64-ENCODE(rawUUID).
   DISPLAY cBase64UUID.
END.
```


Functions - GUID

- Formats a 16byte raw UUID
  - XXXXXXXX—XXXX—XXXX—XXXX—XXXXXXXXXXXX

```
DEFINE VARIABLE MyUUID AS RAW         NO-UNDO.
DEFINE VARIABLE vGUID  AS CHARACTER NO-UNDO.
DEFINE VARIABLE vGUID2 AS CHARACTER NO-UNDO.

ASSIGN MyUUID = GENERATE-UUID
       vGUID  = GUID (MyUUID)
       vGUID2 = GUID. /* Calls GENERATE-UUID */
```


Application Events and Context – Demo. Lab4

Agenda

- Overview
- Getting started
- Audit Policy Maintenance
- Authentication
- Database Events
- Application Events
- Internal Events

What gets Audited?

Internal events

- Authentication (login)
- Database connections
- Schema changes
- Audit policy administration
- Security administration
- Database utilities
- Audit archiving

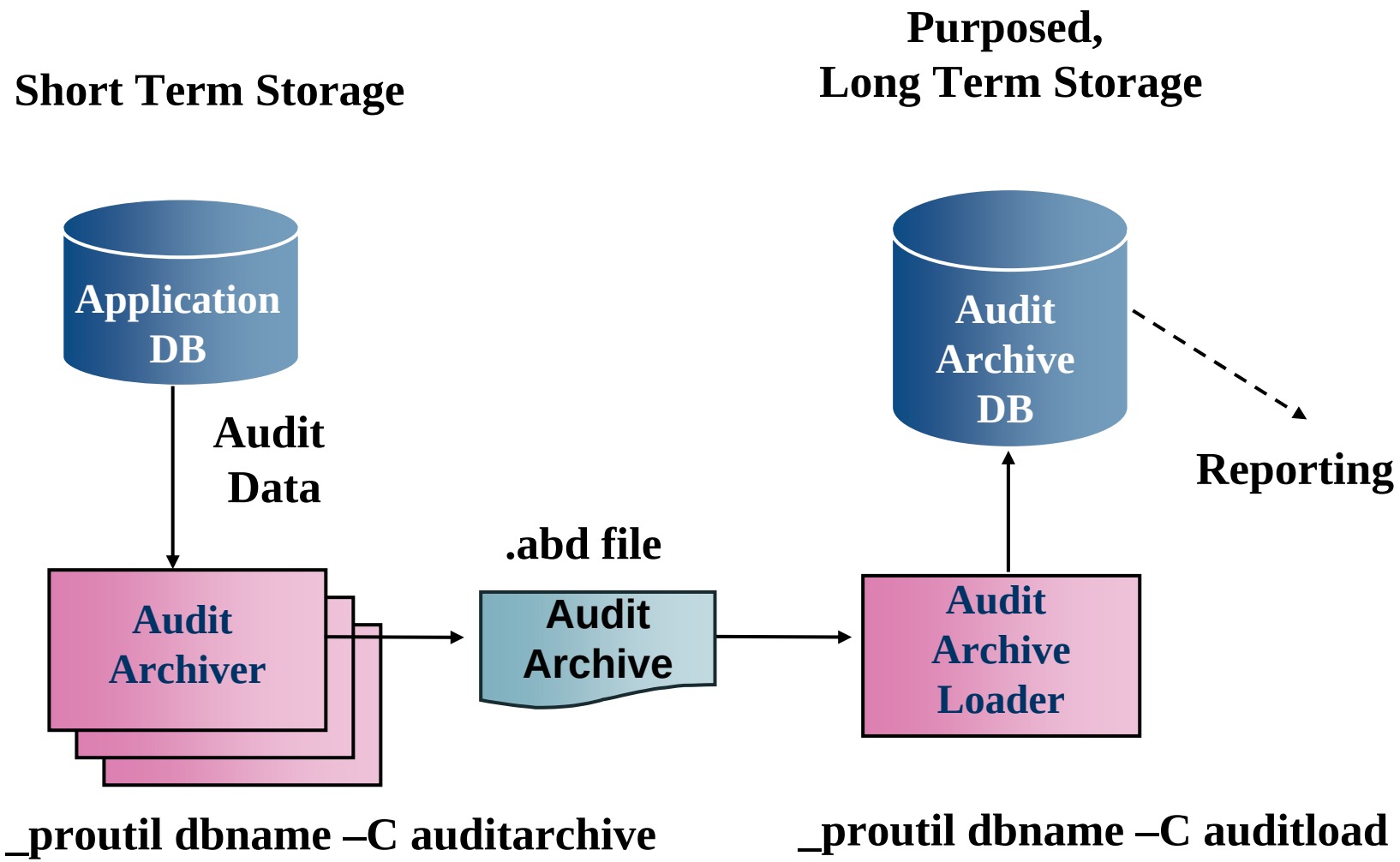
What is NOT Audited?

Database utilities

- Non record based utilities
  - Prolog, prostrct, …
- Probkup, prorest, procopy
- Proutil
  - Idxcheck, idxfix, index deactivate

Audit Archival Utility

Internal events



Audit Data Archival Utility

Archiving audit data

- Must have Audit Archive privilege to run
- May be scheduled, e.g. CRON
- Fast binary dump / load using .abd file
- Optional delete of source audit data on dump
- Supports
  - Multiple simultaneous invocation online
  - Online operation
- Is an auditable event

Audit Data Archival Utility

Audit Archive - command line syntax

```
_proutil <dbname> -C auditarchive
        [date-range [date-range2]] [-recs num-recs]
        [–nodelete] [-directory directory | /dev/null ]
        [-userid userid –password password]
        [-checkseal]
```

- Date range format
  - “MM-DD-YYYY HH:MM:SS.SSS+HH:MM”
  - Must be quoted
- Records deleted *num-recs* at a time

Archive Load Operation

Loading audit data - command line syntax

```
_proutil <dbname> -C auditload
        audit-archive-file-name [-recs num-recs]
        [-userid userid –password password]
        [-checkseal]
```

- Records loaded *num-recs* at a time
- Duplicates are ignored

Auditing - Summary

- 10.1A provides uninterrupted trail of audit events
  - Database, application, internal
- Secure, tamper resistant audit data and policies
- Flexible and scalable
- Built-in auditing for 4GL and SQL clients
- High performance

Documentation and Education

- OpenEdge
  Getting Started:
  Core Business Services
- Webpapers
- Education
  What's New 10.1 – Auditing