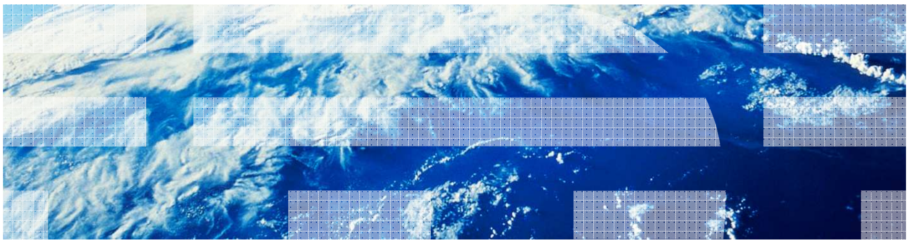
*Gareth Jones*
*DB2 for z/OS Development*
*jonesgth@uk.ibm.com*

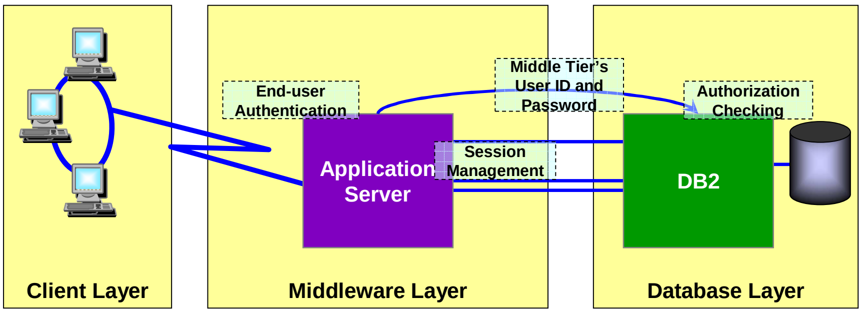
# Trusted Contexts and Database Roles: Should Application Developers Care About Them?



## Agenda

- Trusted Contexts and Database Roles
  - What are the problems being addressed?
    - Current three-tier authorization
    - Auditing, accountability and control
    - Privilege management
  - Trusted contexts
  - Trusted contexts and authorization ID switching
  - Performing actions on behalf of other users
  - Roles and context-specific privileges
  - Enterprise Identity Mapping and Trusted Contexts
  - Trusted Contexts And Object Ownership

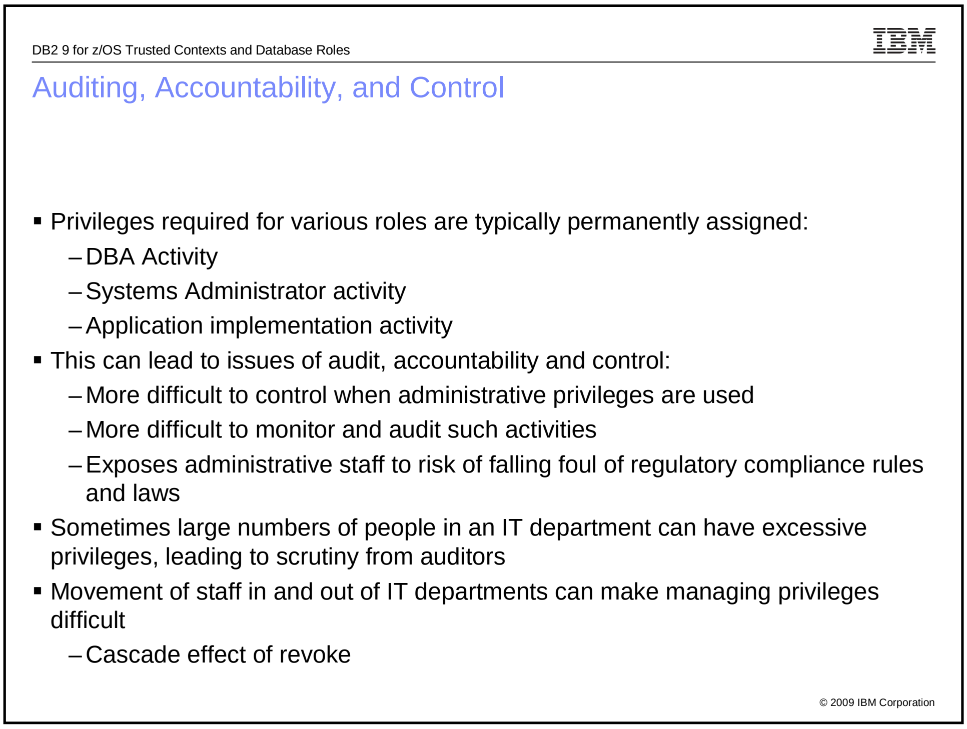
## Current Authentication in a Three-Tier Architecture



- A three-tiered application model with DB2 as the database server:
  - The middle layer authenticates users running client applications.
  - It also manages interactions with the database server.
  - The middle layer's user ID and password are used for database authentication.
  - The privileges of the associated authorization id are checked when accessing the database, including all access on behalf of all end-users.

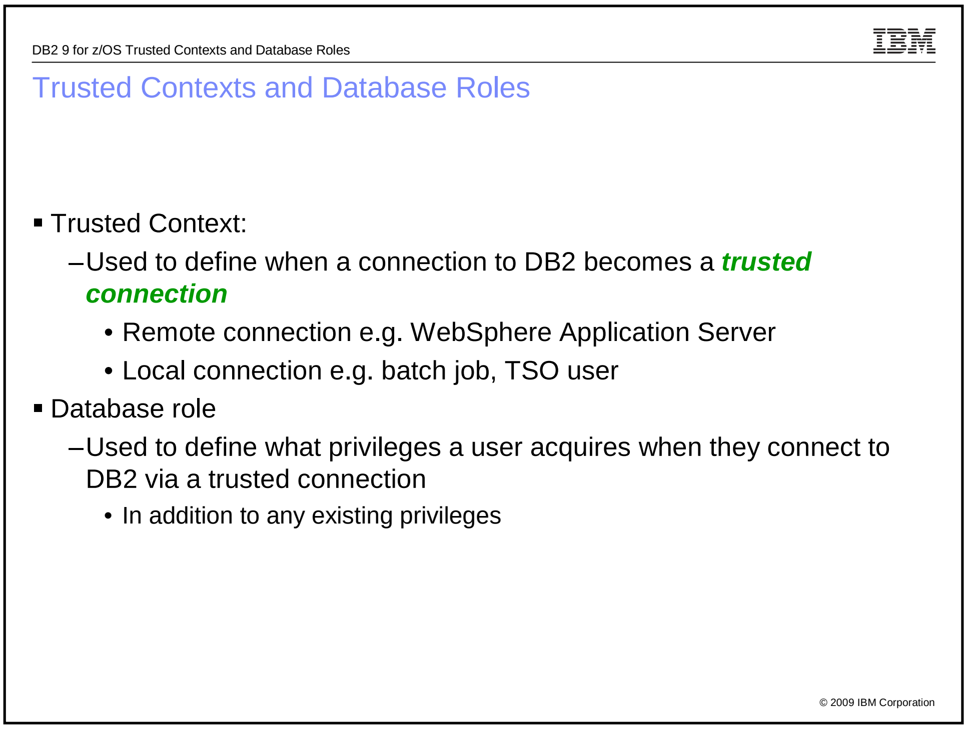
## Three-tier Authentication – The Issues

- Problems with the current implementation:
  - Loss of end-user identity.
  - Loss of control over end-user access of the database.
  - Diminished accountability.
  - The middleware server's AUTHID needs the privileges to perform *all* requests from *all* end-users.
  - If the middleware server's security is compromised, so is that of the database server.
- Problems with establishing a new connection using the end user's ID and password:
  - Performance overhead:
    - Creating a new connection to the database server;
    - Re-authenticating the end-user at the database server
  - Not possible for servers without access to end-user credentials.



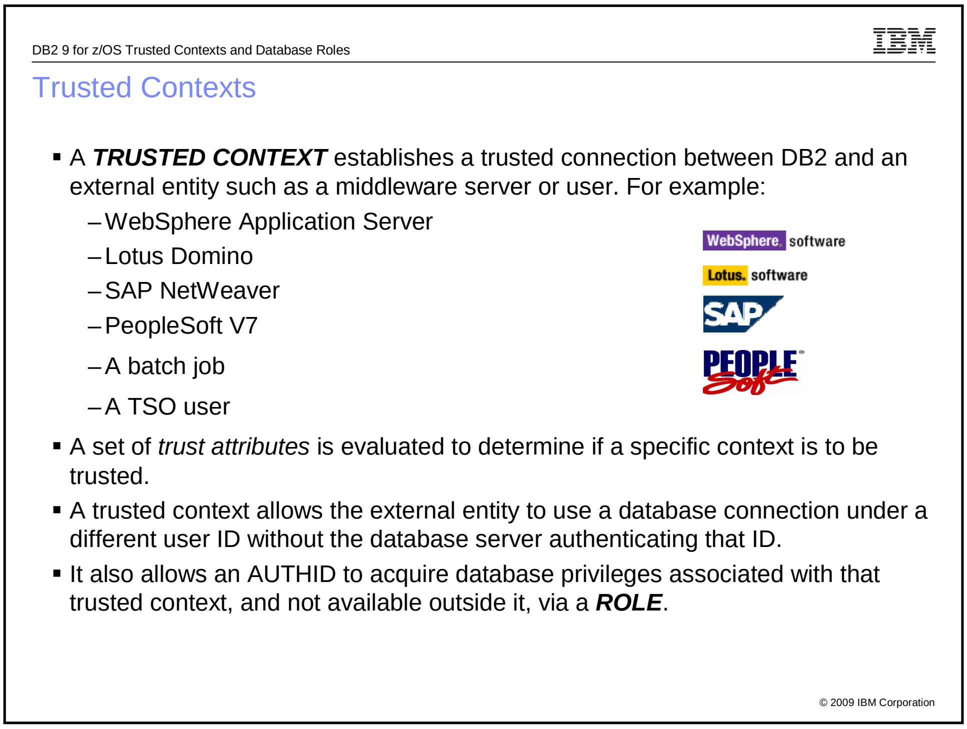
## Auditing, Accountability, and Control

- Privileges required for various roles are typically permanently assigned:
  - DBA Activity
  - Systems Administrator activity
  - Application implementation activity
- This can lead to issues of audit, accountability and control:
  - More difficult to control when administrative privileges are used
  - More difficult to monitor and audit such activities
  - Exposes administrative staff to risk of falling foul of regulatory compliance rules and laws
- Sometimes large numbers of people in an IT department can have excessive privileges, leading to scrutiny from auditors
- Movement of staff in and out of IT departments can make managing privileges difficult
  - Cascade effect of revoke



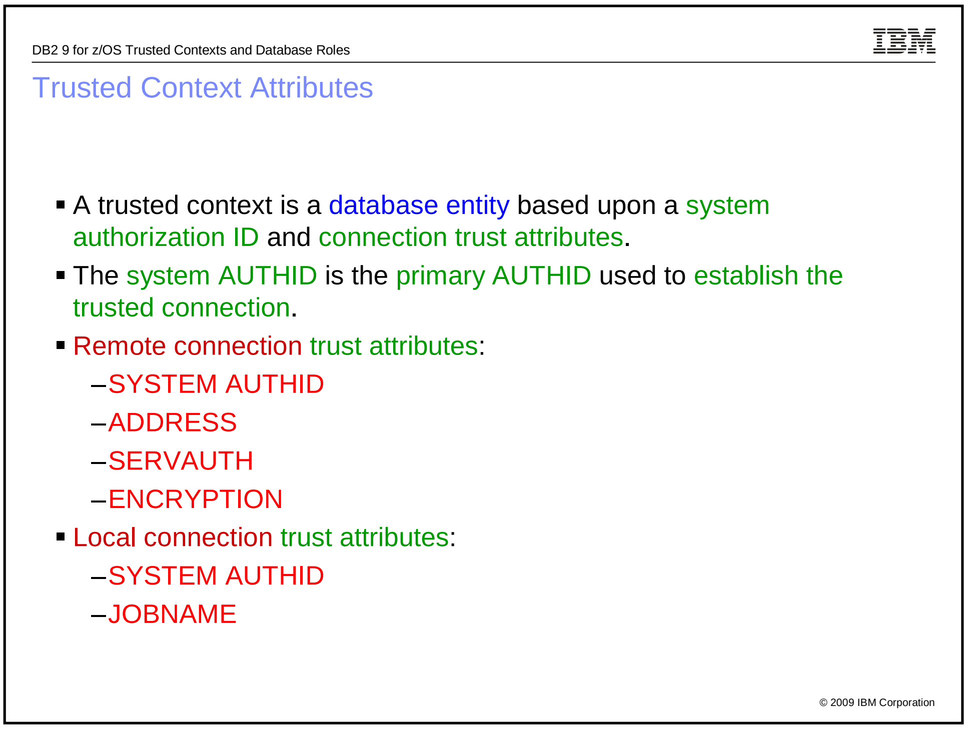
## Trusted Contexts and Database Roles

- Trusted Context:
  - Used to define when a connection to DB2 becomes a ***trusted connection***
    - Remote connection e.g. WebSphere Application Server
    - Local connection e.g. batch job, TSO user
- Database role
  - Used to define what privileges a user acquires when they connect to DB2 via a trusted connection
    - In addition to any existing privileges



## Trusted Contexts

- A ***TRUSTED CONTEXT*** establishes a trusted connection between DB2 and an external entity such as a middleware server or user. For example:
  - WebSphere Application Server
  - Lotus Domino
  - SAP NetWeaver
  - PeopleSoft V7
  - A batch job
  - A TSO user
- A set of *trust attributes* is evaluated to determine if a specific context is to be trusted.
- A trusted context allows the external entity to use a database connection under a different user ID without the database server authenticating that ID.
- It also allows an AUTHID to acquire database privileges associated with that trusted context, and not available outside it, via a ***ROLE***.



## Trusted Context Attributes

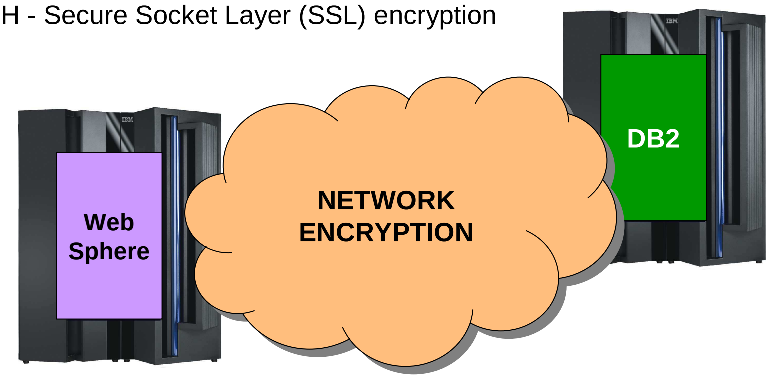
- A trusted context is a database entity based upon a system authorization ID and connection trust attributes.
- The system AUTHID is the primary AUTHID used to establish the trusted connection.
- Remote connection trust attributes:
  - SYSTEM AUTHID
  - ADDRESS
  - SERVAUTH
  - ENCRYPTION
- Local connection trust attributes:
  - SYSTEM AUTHID
  - JOBNAME

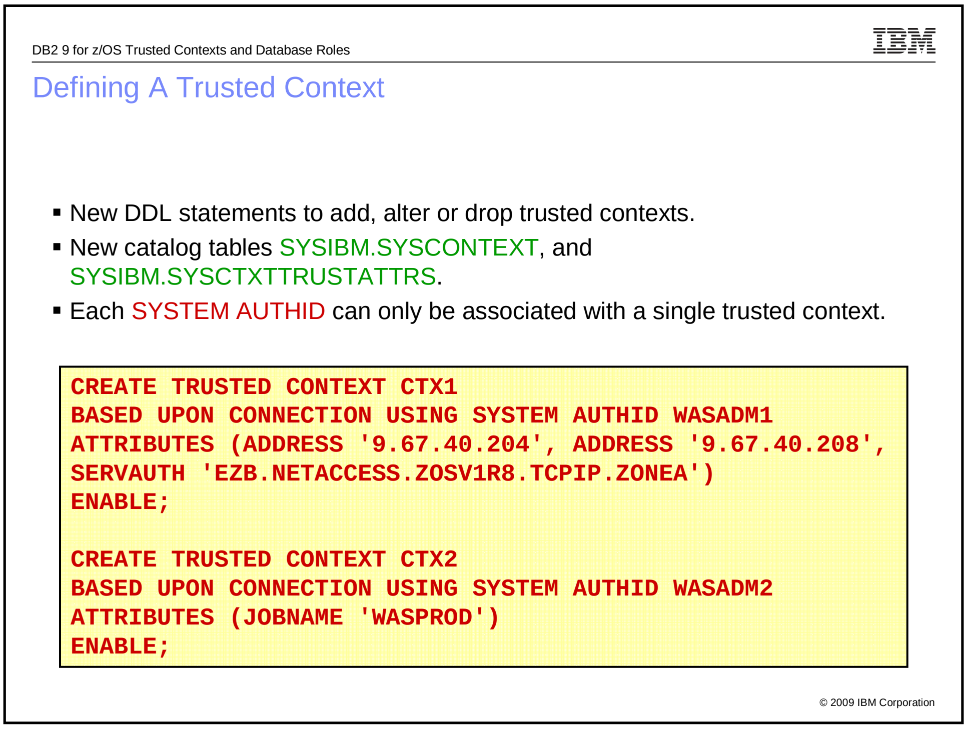
## Local And Remote Trusted Context Attributes

- Remote connection trust attributes:
  - SYSTEM AUTHID – the system user ID provided by e.g. a middleware server.
  - ADDRESS – IP address or domain name (restricted to TCP/IP only).
  - SERVAUTH – a resource in the RACF SERVAUTH class.
  - ENCRYPTION – minimum level of encryption for the connection.
- Local connection trust attributes:
  - SYSTEM AUTHID is typically derived from:
    - Started task (RRSAF) – JOB statement USER or RACF USER
    - TSO – TSO logon ID
    - BATCH – JOB statement USER
  - JOBNAME is derived from:
    - Started task (RRSAF) – JOB or started class name
    - TSO – TSO logon ID
    - BATCH – JOB name

## The ENCRYPTION Attribute

- Defines the minimum encryption level required for the data stream for the connection. Supported values are:
  - NONE - No encryption. The default.
  - LOW - DRDA data stream encryption
  - HIGH - Secure Socket Layer (SSL) encryption

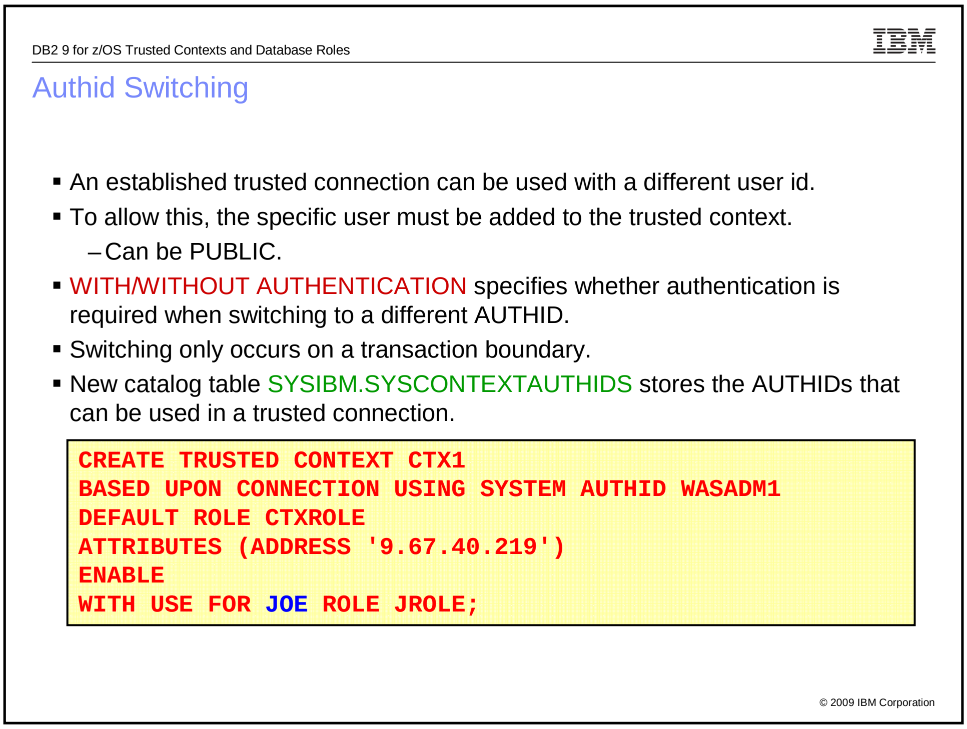




## Defining A Trusted Context

- New DDL statements to add, alter or drop trusted contexts.
- New catalog tables SYSIBM.SYSCONTEXT, and SYSIBM.SYSCTXTTRUSTATTRS.
- Each SYSTEM AUTHID can only be associated with a single trusted context.

```
CREATE TRUSTED CONTEXT CTX1
BASED UPON CONNECTION USING SYSTEM AUTHID WASADM1
ATTRIBUTES (ADDRESS '9.67.40.204', ADDRESS '9.67.40.208',
SERVAUTH 'EZB.NETACCESS.ZOSV1R8.TCPIP.ZONEA')
ENABLE;

CREATE TRUSTED CONTEXT CTX2
BASED UPON CONNECTION USING SYSTEM AUTHID WASADM2
ATTRIBUTES (JOBNAME 'WASPROD')
ENABLE;
```



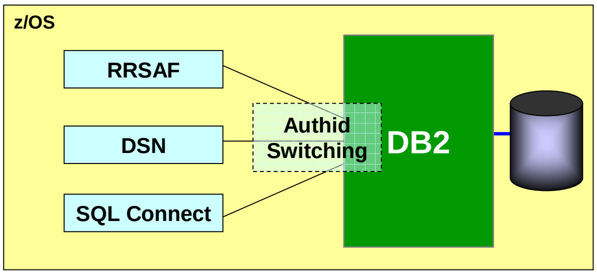
## Authid Switching

- An established trusted connection can be used with a different user id.
- To allow this, the specific user must be added to the trusted context.
  - Can be PUBLIC.
- WITH/WITHOUT AUTHENTICATION specifies whether authentication is required when switching to a different AUTHID.
- Switching only occurs on a transaction boundary.
- New catalog table SYSIBM.SYSCONTEXTAUTHIDS stores the AUTHIDs that can be used in a trusted connection.

```
CREATE TRUSTED CONTEXT CTX1
BASED UPON CONNECTION USING SYSTEM AUTHID WASADM1
DEFAULT ROLE CTXROLE
ATTRIBUTES (ADDRESS '9.67.40.219')
ENABLE
WITH USE FOR JOE ROLE JROLE;
```

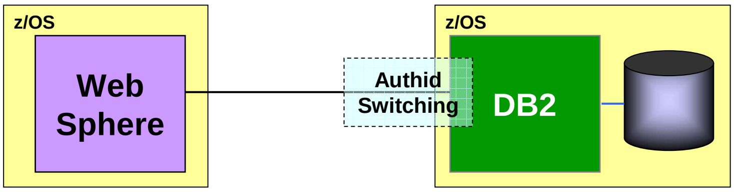
## Authid Switching – Local Processing

- Allowing a trusted connection to be used by a different user at a local DB2:
  - RRSAF: the SIGNON function in CALL DSNRLI.
  - The DSN Command processor: the new ASUSER option.
  - SQL CONNECT, via the USER and USING clauses (only locally).
- In all cases, if the primary AUTHID does not have access to the trusted context, then the connection request fails and returns to an unconnected state.



## Authid Switching, Remote Processing (DB2 Server)

- When DB2, as a server, receives a request to switch users it:
  - Calls the connection exit, which associates AUTHID set and an SQL ID with the remote request, replacing the previous ones.
  - Determines if the primary AUTHID is allowed to use the trusted connection: if WITH AUTHENTICATION is specified, an authentication token is required.
  - Performs SECURITY LABEL verification for the new user ID.
  - Initializes the connection, creating a ‘clean’ environment, e.g. open cursors are closed, temporary table information is dropped.
  - If the primary AUTHID is not allowed to use the trusted connection or SECURITY LABEL verification fails, then the connection state is *unconnected*.



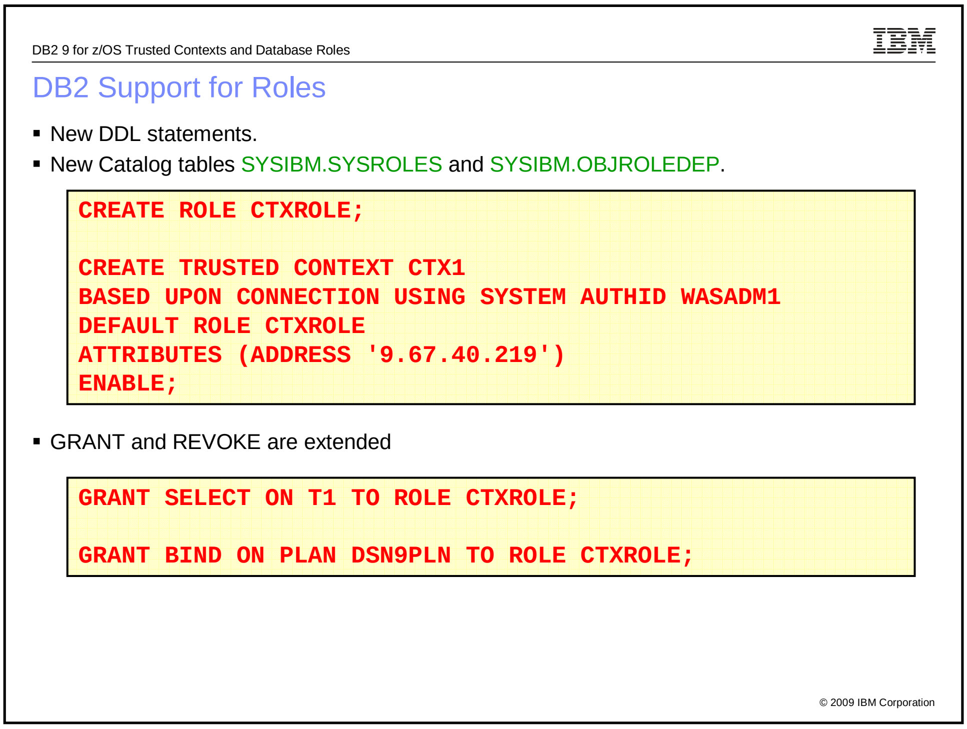
## Performing Actions On Behalf Of Other Users

```
  CREATE TRUSTED CONTEXT CTX1
  BASED UPON CONNECTION
  USING SYSTEM AUTHID PRODDBA
  ATTRIBUTES (JOBNAME ‘PRODDBA1’)
  ENABLE
  WITH USE FOR PRODOWNR;
//PRODDBA1 JOB  USER=‘PRODDBA’
//IKJEFT1B EXEC PGM=IKJEFT1B
//SYSTSPRT DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSTSIN  DD   *
  DSN SYSTEM(DB1P) ASUSER(PRODOWNR)
  END
//SYSIN    DD   *
  CREATE VIEW PRODVIEW AS SELECT ... ;
  COMMIT ;
  GRANT SELECT ON PRODVIEW TO PUBLIC;
  COMMIT;
//
```

## Roles and Context-specific Privileges

- **Roles** provide the flexibility to grant privileges to an AUTHID only when the user is connected via a trusted connection.
- They greatly simplify management of authorization.
- An individual **role** can be defined for any AUTHID using the trusted connection, in which case the user inherits the privileges granted to the individual **role**.
- Where there is no individual **role**, any AUTHID using a trusted context inherits the privileges of the trusted context’s default **role,** if defined.

```
CREATE TRUSTED CONTEXT CTX1
BASED UPON CONNECTION USING SYSTEM AUTHID WASADM1
DEFAULT ROLE CTXROLE
ATTRIBUTES (ADDRESS '9.67.40.219')
ENABLE
WITH USE FOR JOE ROLE JROLE;
```



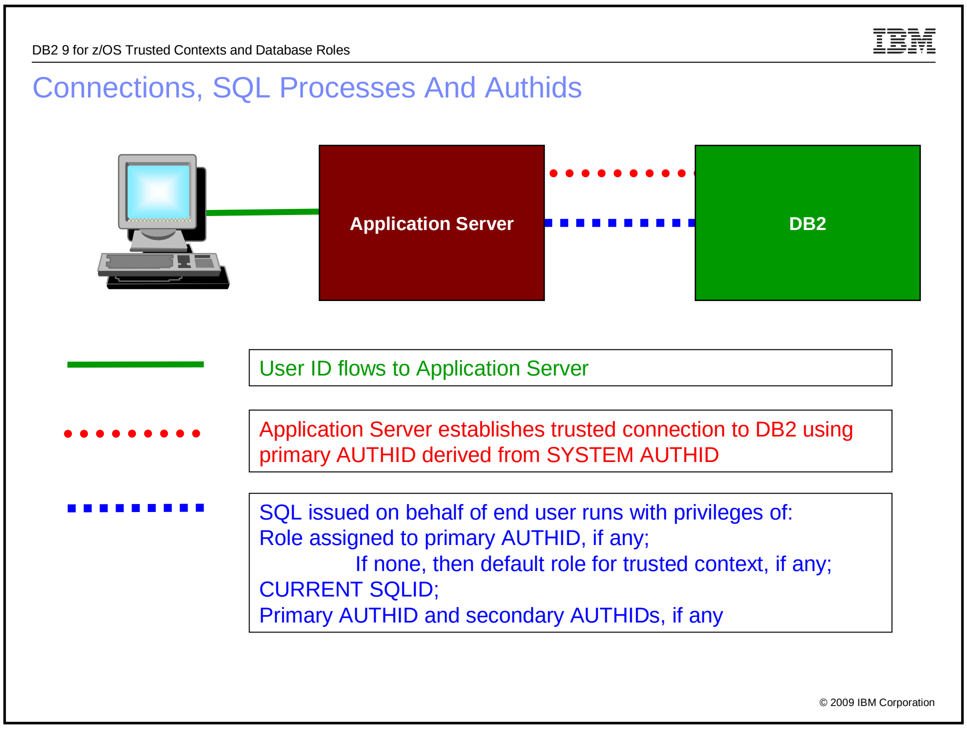
## DB2 Support for Roles

- New DDL statements.
- New Catalog tables SYSIBM.SYSROLES and SYSIBM.OBJROLEDEP.

```
CREATE ROLE CTXROLE;

CREATE TRUSTED CONTEXT CTX1
BASED UPON CONNECTION USING SYSTEM AUTHID WASADM1
DEFAULT ROLE CTXROLE
ATTRIBUTES (ADDRESS '9.67.40.219')
ENABLE;
```

- GRANT and REVOKE are extended

```
GRANT SELECT ON T1 TO ROLE CTXROLE;

GRANT BIND ON PLAN DSN9PLN TO ROLE CTXROLE;
```



## Connections, SQL Processes And Authids

━━━━━ User ID flows to Application Server

• • • • • • • • • Application Server establishes trusted connection to DB2 using primary AUTHID derived from SYSTEM AUTHID

■ ■ ■ ■ ■ ■ ■ ■ ■ SQL issued on behalf of end user runs with privileges of:
Role assigned to primary AUTHID, if any;
If none, then default role for trusted context, if any;
CURRENT SQLID;
Primary AUTHID and secondary AUTHIDs, if any

## Usage Scenarios

- Securing An Application Server
  - Most existing application servers connect to DB2 using userid/password pairs:
    - Significant exposure if someone steals the userid/password
  - A trusted context and roles can be used to limit exposure by ensuring that role privileges are only valid when used by a valid application server IP address.
- Dynamic SQL auditing
  - GRANT dynamic SQL privileges to a ROLE
  - End user identity can be delegated directly to DB2 without granting dynamic SQL privileges directly to the end user
  - End user passwords can be optional.
  - No added complexity for administration of GRANTs, while retaining the ability to audit the end user’s identity
- Trusted Contexts and already-verified DRDA
  - Improves ability to replace SNA connections with TCP/IP
  - Communication Database is used to identify trusted connections and specify “system userid” for the Trusted Context
  - End user identity is automatically propagated from one DB2 system to the other.

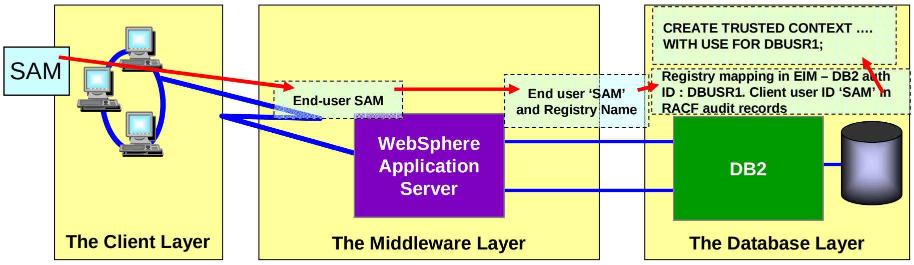
## Auditing DBA Activities

- Many sites need to be able to audit DBA access to sensitive customer data. DB2 9 can help by enabling an auditable DBA process:
  1. Grant DBA privileges to a ROLE
  2. Start audit trace for that ROLE
  3. When a DBA needs to perform a system change:
     - Use Trusted Context to assign DBA ROLE to person
     - DBA is given request and performs activity
     - Remove role association from trusted context
  4. Have another person review the audit trace

## Enterprise Identity Mapping

- Users often have multiple user identities in multiple registries (e.g. RACF, LDAP, Kerberos) on multiple platforms
- Users often authenticate on one platform and run applications on another
  - Often means re-authentication on the application platform or running under a system user identity on the application platform; this creates a potential security exposure
- z/OS feature EIM allows EIM administrators to define relationships between those user identities
- Applications can use the EIM API to allow users to authenticate once without having to re-authenticate when switching platforms and identities

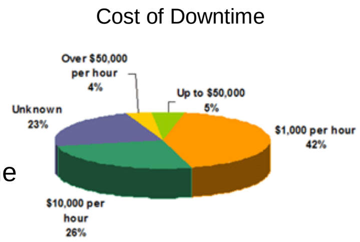
## Trusted Contexts and Enterprise Identity Mapping


SAM
End-user SAM
End user 'SAM' and Registry Name
CREATE TRUSTED CONTEXT .... WITH USE FOR DBUSR1;
Registry mapping in EIM – DB2 auth ID : DBUSR1. Client user ID 'SAM' in RACF audit records
WebSphere Application Server
DB2
The Client Layer
The Middleware Layer
The Database Layer

- Configure WebSphere Application Server to create trusted connection and send the registry name to the server
  - Database property 'propagateClientIdentityUsingTrustedContext' is set to 'true'
  - Application parameter 'TargetRealmName' is set to the registry name at the server
- DB2 passes the registry name and client user ID to EIM Domain controller to obtain DB2 auth ID
- DB2 checks if the DB2 auth ID is allowed to use the trusted connection
- Client user ID, WAS security token & DB2 auth ID recorded in DB2 audit logs
- Client user ID, DB2 auth ID, & registry name are recorded in RACF audit logs

## Trusted Contexts And Object Ownership

- Outside trusted contexts and roles, object ownership is tied to a user.
- When a user creates an object, they become its owner.
- To remove the privileges of that user on the object, it has to be dropped; all grants associated with it are revoked.
- The object then has to be recreated and the privileges re-granted.
- If the object owner is a role, removing privileges from the end-user will not require the object to be dropped and recreated.



## Trusted Contexts And Object Ownership (cont.)

- Role ownership allows tighter security controls: e.g. DBAs only exercise privileges when performing approved activities via a trusted context and role.
- When a trusted context has a default role, the role becomes the owner of  created objects, if ROLE AS OBJECT OWNER is specified.
- When a role is defined as the object owner, then it must have all the privileges necessary to create the object.
- If ROLE AS OBJECT OWNER is not specified, there is no change in determining object ownership.
- If a role owns a created object, then the user inheriting the privileges of the role through a trusted context requires a GRANT to access it outside the trusted context.

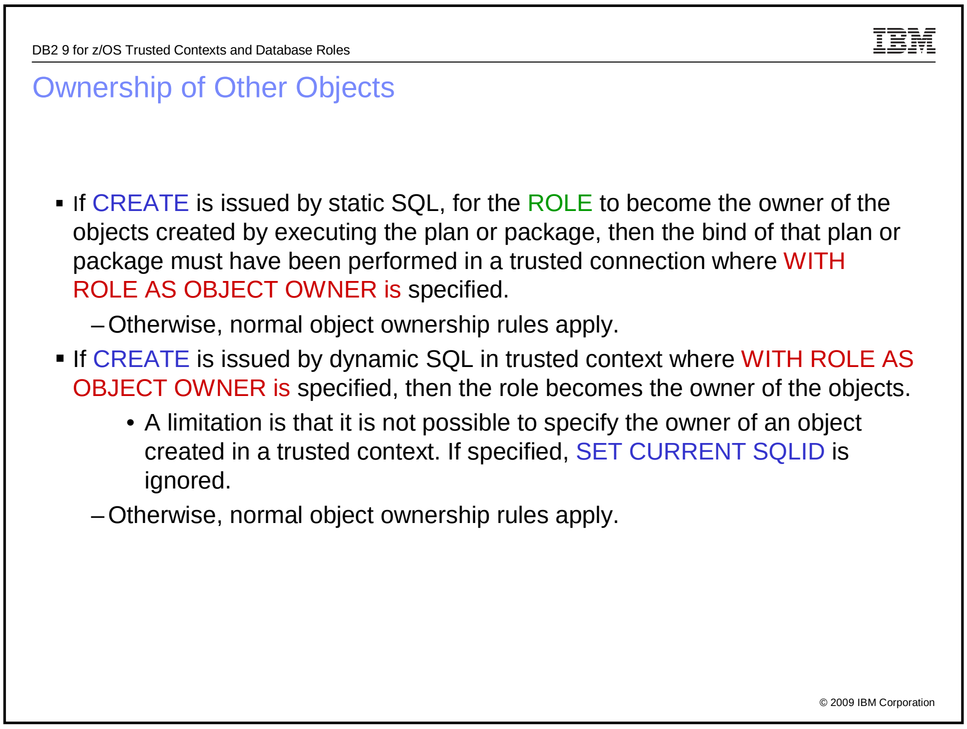
```
CREATE ROLE CTXROLE;

CREATE TRUSTED CONTEXT CTX1 …
DEFAULT ROLE CTXROLE WITH ROLE AS OBJECT OWNER … ;
```

## Plan And Package Ownership:

- Determining ownership when BIND or REBIND are issued in a trusted context, and WITH ROLE AS OBJECT OWNER is specified:
  - If the OWNER BIND option is not specified, the role associated with the binder becomes the owner.
  - If the OWNER BIND option is specified, the ROLE specified for OWNER becomes the owner (the OWNER specified must be a ROLE).
    - The binder needs to be granted BINDAGENT from that ROLE.
    - The binder also receives BINDAGENT, if the ROLE associated with the binder has BINDAGENT.
- If WITHOUT ROLE AS OBJECT OWNER is specified (or defaulted) for the trusted context, then the current rules for BIND and REBIND ownership apply.
  - If a role is associated in a trusted context, then the role privileges are included in the binder's privilege set to determine if the binder is allowed to perform the bind.

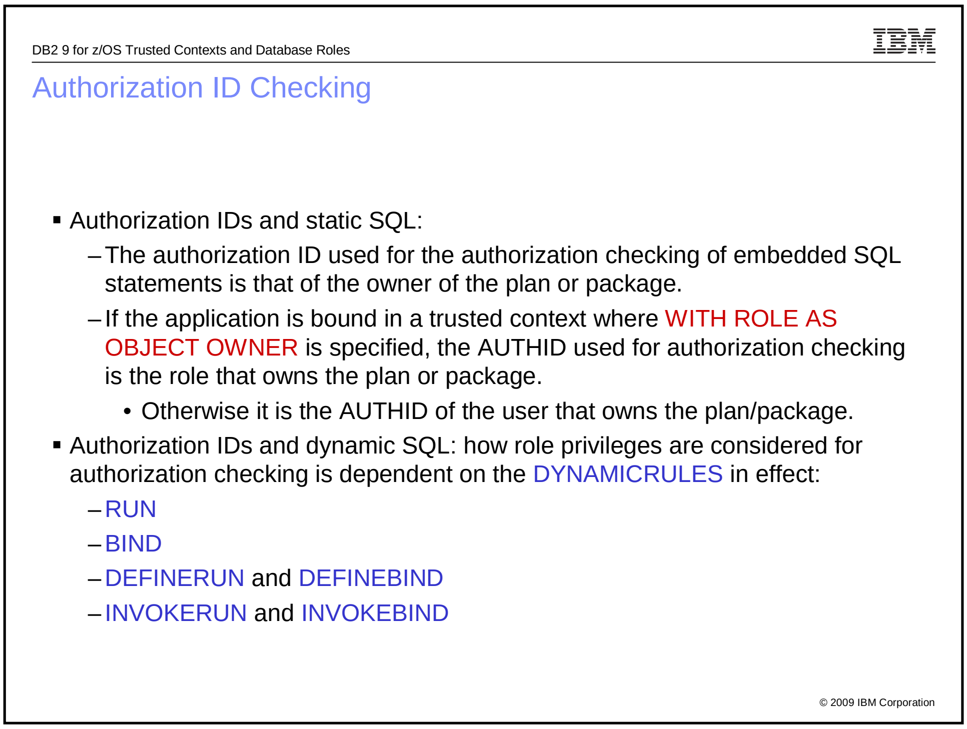
## Plan And Package Ownership Considerations:

- Plan and Package ownership considerations:
  - For a package to be bound remotely with a ROLE as the owner of the package at the remote DB2, then the trusted context at the remote DB2 must be specified as WITH ROLE AS OBJECT OWNER.
  - If OWNER is specified for a remote BIND across a trusted connection, OWNER could be a role or an AUTHID. Outbound AUTHID translation is not performed for the OWNER.
  - If the plan owner is a role and the application uses a package bound at a remote DB2 server, then the plan owner privilege to execute the package is not considered at the remote DB2 server.
    - The package owner/the process runner (as determined by DYNAMICRULES) at the DB2 server must have the EXECUTE privilege on the package at the remote server.



## Ownership of Other Objects

- If CREATE is issued by static SQL, for the ROLE to become the owner of the objects created by executing the plan or package, then the bind of that plan or package must have been performed in a trusted connection where WITH ROLE AS OBJECT OWNER is specified.
  - Otherwise, normal object ownership rules apply.
- If CREATE is issued by dynamic SQL in trusted context where WITH ROLE AS OBJECT OWNER is specified, then the role becomes the owner of the objects.
    - A limitation is that it is not possible to specify the owner of an object created in a trusted context. If specified, SET CURRENT SQLID is ignored.
  - Otherwise, normal object ownership rules apply.



## Authorization ID Checking

- Authorization IDs and static SQL:
  - The authorization ID used for the authorization checking of embedded SQL statements is that of the owner of the plan or package.
  - If the application is bound in a trusted context where WITH ROLE AS OBJECT OWNER is specified, the AUTHID used for authorization checking is the role that owns the plan or package.
    - Otherwise it is the AUTHID of the user that owns the plan/package.
- Authorization IDs and dynamic SQL: how role privileges are considered for authorization checking is dependent on the DYNAMICRULES in effect:
  - RUN
  - BIND
  - DEFINERUN and DEFINEBIND
  - INVOKERUN and INVOKEBIND

## -DISPLAY THREAD Report and Other Changes

```
DSNV401I - DISPLAY THREAD REPORT FOLLOWS -
DSNV402I - ACTIVE THREADS - 133
NAME     ST A    REQ ID           AUTHID     PLAN     ASID TOKEN
BATCH    T  *     10 JOB01        ADMF001    APPL01   0027 12
 V485-TRUSTED CONTEXT=DOMINOCONTEXT, SYSTEM AUTHID=SYSADM,
      ROLE=USRROLE
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I - DSNVDT '-DIS THD' NORMAL COMPLETION
```

### Other changes:

- Trusted context name, role name, original application user, and security token fields are added to the IFCID correlation header.
- IFCIDs  which include extra information about trusted contexts and roles:
  - IFCIDs 62, 140, 141, 142, 169 and 314.
- New IFCIDs (added to audit trace class 10):
  - IFCIDs 269 and 270
- Changes to the Access Control Authorization Exit