13

Controlling User Access

ORACLE

Schedule:	Timing	Topic
	20 minutes	Lecture
	20 minutes	Practice
	40 minutes	Total

Objectives

After completing this lesson, you should be able to do the following:

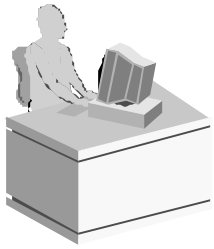
- **Create users**
- **Create roles to ease setup and maintenance of the security model**
- **Use the `GRANT` and `REVOKE` statements to grant and revoke object privileges**
- **Create and access database links**

ORACLE

Lesson Aim

In this lesson, you learn how to control database access to specific objects and add new users with different levels of access privileges.

Controlling User Access



Controlling User Access

In a multiple-user environment, you want to maintain security of the database access and use. With Oracle server database security, you can do the following:

- Control database access
- Give access to specific objects in the database
- Confirm given and received *privileges* with the Oracle data dictionary
- Create synonyms for database objects

Database security can be classified into two categories: system security and data security. System security covers access and use of the database at the system level, such as the username and password, the disk space allocated to users, and the system operations that users can perform. Database security covers access and use of the database objects and the actions that those users can have on the objects.

Privileges

- **Database security:**
  - **System security**
  - **Data security**
- **System privileges: Gaining access to the database**
- **Object privileges: Manipulating the content of the database objects**
- **Schemas: Collections of objects, such as tables, views, and sequences**

ORACLE

Privileges

Privileges are the right to execute particular SQL statements. The database administrator (DBA) is a high-level user with the ability to grant users access to the database and its objects. The users require *system privileges* to gain access to the database and *object privileges* to manipulate the content of the objects in the database. Users can also be given the privilege to grant additional privileges to other users or to *roles*, which are named groups of related privileges.

Schemas

A *schema* is a collection of objects, such as tables, views, and sequences. The schema is owned by a database user and has the same name as that user.

For more information, see *Oracle9i Application Developer's Guide - Fundamentals,* "Establishing a Security Policy" section, and *Oracle9i Concepts*, "Database Security" topic.

System Privileges

- **More than 100 privileges are available.**
- **The database administrator has high-level system privileges for tasks such as:**
  - **Creating new users**
  - **Removing users**
  - **Removing tables**
  - **Backing up tables**

System Privileges

More than 100 distinct system privileges are available for users and roles. System privileges typically are provided by the database administrator.

Typical DBA Privileges

System Privilege	Operations Authorized
CREATE USER	Grantee can create other Oracle users (a privilege required for a DBA role).
DROP USER	Grantee can drop another user.
DROP ANY TABLE	Grantee can drop a table in any schema.
BACKUP ANY TABLE	Grantee can back up any table in any schema with the export utility.
SELECT ANY TABLE	Grantee can query tables, views, or snapshots in any schema.
CREATE ANY TABLE	Grantee can create tables in any schema.

Creating Users

The DBA creates users by using the CREATE USER statement.

```
CREATE USER user
IDENTIFIED BY   password;
```

```
CREATE USER  scott
IDENTIFIED BY   tiger;
User created.
```

ORACLE

Creating a User

The DBA creates the user by executing the CREATE USER statement. The user does not have any privileges at this point. The DBA can then grant privileges to that user. These privileges determine what the user can do at the database level.

The slide gives the abridged syntax for creating a user.

In the syntax:

user	is the name of the user to be created
password	specifies that the user must log in with this password

For more information, see *Oracle9i SQL Reference*, "GRANT" and "CREATE USER."

Instructor Note

For information on DROP USER, refer to *Oracle9i SQL Reference*, "DROP USER."

User System Privileges

- **Once a user is created, the DBA can grant specific system privileges to a user.**

```
GRANT privilege [, privilege...]
TO user [, user| role, PUBLIC...];
```

- **An application developer, for example, may have the following system privileges:**
  - **CREATE SESSION**
  - **CREATE TABLE**
  - **CREATE SEQUENCE**
  - **CREATE VIEW**
  - **CREATE PROCEDURE**

ORACLE

Typical User Privileges

Now that the DBA has created a user, the DBA can assign privileges to that user.

System Privilege	Operations Authorized
CREATE SESSION	Connect to the database
CREATE TABLE	Create tables in the user's schema
CREATE SEQUENCE	Create a sequence in the user's schema
CREATE VIEW	Create a view in the user's schema
CREATE PROCEDURE	Create a stored procedure, function, or package in the user's schema

In the syntax:

privilege	is the system privilege to be granted
user |role|PUBLIC	is the name of the user, the name of the role, or PUBLIC designates that every user is granted the privilege

Note: Current system privileges can be found in the dictionary view SESSION_PRIVS.

Instructor Note

The syntax displayed for the GRANT command is not the full syntax for the statement.

Granting System Privileges

The DBA can grant a user specific system privileges.

```
GRANT  create session, create table,
       create sequence, create view
TO     scott;
Grant succeeded.
```

ORACLE

Granting System Privileges

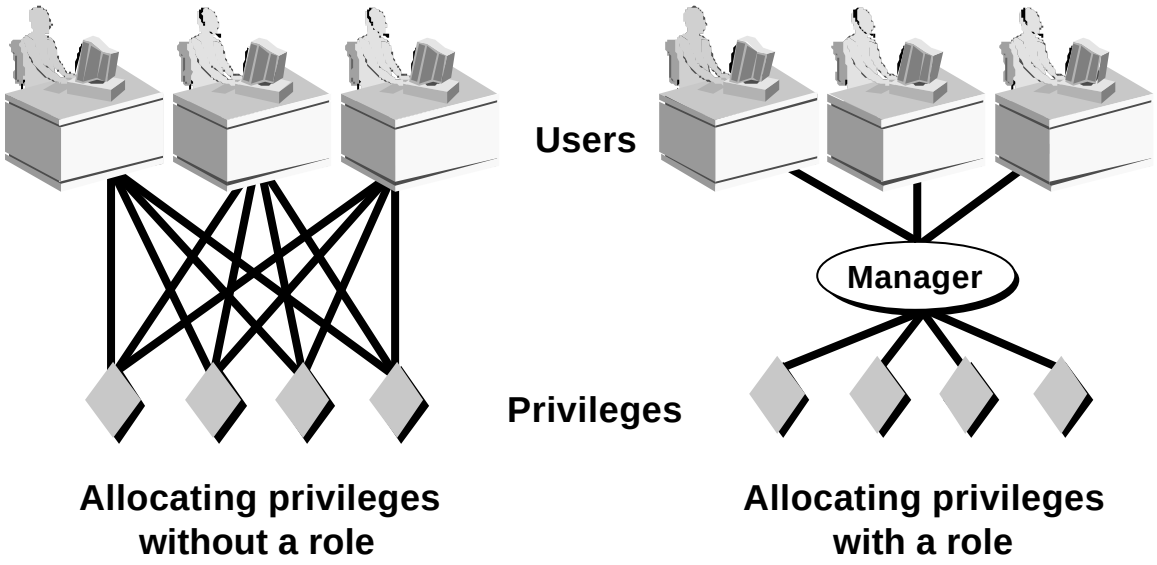
The DBA uses the GRANT statement to allocate system privileges to the user. Once the user has been granted the privileges, the user can immediately use those privileges.

In the example on the slide, user Scott has been assigned the privileges to create sessions, tables, sequences, and views.

Instructor Note

A user needs to have the required space quota to create tables.

What is a Role?



What is a Role?

A role is a named group of related privileges that can be granted to the user. This method makes it easier to revoke and maintain privileges.

A user can have access to several roles, and several users can be assigned the same role. Roles are typically created for a database application.

Creating and Assigning a Role

First, the DBA must create the role. Then the DBA can assign privileges to the role and users to the role.

Syntax

```
CREATE   ROLE  role;
```

In the syntax:

role	is the name of the role to be created

Now that the role is created, the DBA can use the GRANT statement to assign users to the role as well as assign privileges to the role.

Instructor Note

Discuss the following four points about roles:

- Are named groups of related privileges
- Can be granted to users
- Simplify the process of granting and revoking privileges
- Are created by a DBA

Creating and Granting Privileges to a Role

- Create a role

```
CREATE ROLE manager;
Role created.
```

- Grant privileges to a role

```
GRANT create table, create view
TO manager;
Grant succeeded.
```

- Grant a role to users

```
GRANT manager TO DEHAAN, KOCHHAR;
Grant succeeded.
```


Creating a Role

The example on the slide creates a manager role and then allows managers to create tables and views. It then grants DeHaan and Kochhar the role of managers. Now DeHaan and Kochhar can create tables and views.

If users have multiple roles granted to them, they receive all of the privileges associated with all of the roles.

Changing Your Password

- **The DBA creates your user account and initializes your password.**
- **You can change your password by using the ALTER USER statement.**

```
ALTER USER scott
IDENTIFIED BY lion;
User altered.
```

ORACLE

Changing Your Password

The DBA creates an account and initializes a password for every user. You can change your password by using the ALTER USER statement.

Syntax

```
ALTER USER user IDENTIFIED BY password;
```

In the syntax:

user	is the name of the user
password	specifies the new password

Although this statement can be used to change your password, there are many other options. You must have the ALTER USER privilege to change any other option.

For more information, see *Oracle9i SQL Reference*, "ALTER USER."

Object Privileges

Object Privilege	Table	View	Sequence	Procedure
ALTER	Ö		Ö	
DELETE	Ö	Ö		
EXECUTE				Ö
INDEX	Ö			
INSERT	Ö	Ö		
REFERENCES	Ö	Ö		
SELECT	Ö	Ö	Ö	
UPDATE	Ö	Ö		

ORACLE

Object Privileges

An *object privilege* is a privilege or right to perform a particular action on a specific table, view, sequence, or procedure. Each object has a particular set of grantable privileges. The table on the slide lists the privileges for various objects. Note that the only privileges that apply to a sequence are `SELECT` and `ALTER`. `UPDATE`, `REFERENCES`, and `INSERT` can be restricted by specifying a subset of updateable columns. A `SELECT` privilege can be restricted by creating a view with a subset of columns and granting the `SELECT` privilege only on the view. A privilege granted on a synonym is converted to a privilege on the base table referenced by the synonym.

Instructor Note

You can use the `ALTER VIEW` and `ALTER PROCEDURE` commands to recompile views and PL/SQL procedures, functions, and packages.

Object Privileges

- **Object privileges vary from object to object.**
- **An owner has all the privileges on the object.**
- **An owner can give specific privileges on that owner's object.**

```
GRANT       object_priv [(columns)]
ON          object
TO          {user|role|PUBLIC}
[WITH GRANT OPTION];
```

ORACLE

Granting Object Privileges

Different object privileges are available for different types of schema objects. A user automatically has all object privileges for schema objects contained in the user's schema. A user can grant any object privilege on any schema object that the user owns to any other user or role. If the grant includes `WITH GRANT OPTION`, then the grantee can further grant the object privilege to other users; otherwise, the grantee can use the privilege but cannot grant it to other users.

In the syntax:

object_priv	is an object privilege to be granted
ALL	specifies all object privileges
columns	specifies the column from a table or view on which privileges are granted
ON *object*	is the object on which the privileges are granted
TO	identifies to whom the privilege is granted
PUBLIC	grants object privileges to all users
WITH GRANT OPTION	allows the grantee to grant the object privileges to other users and roles

Granting Object Privileges

- **Grant query privileges on the EMPLOYEES table.**

```
GRANT  select
ON     employees
TO     sue, rich;
Grant succeeded.
```

- **Grant privileges to update specific columns to users and roles.**

```
GRANT  update (department_name, location_id)
ON     departments
TO     scott, manager;
Grant succeeded.
```

ORACLE

Guidelines

- To grant privileges on an object, the object must be in your own schema, or you must have been granted the object privileges WITH GRANT OPTION.
- An object owner can grant any object privilege on the object to any other user or role of the database.
- The owner of an object automatically acquires all object privileges on that object.

The first example on the slide grants users Sue and Rich the privilege to query your EMPLOYEES table. The second example grants UPDATE privileges on specific columns in the DEPARTMENTS table to Scott and to the manager role.

If Sue or Rich now want to SELECT data from the employees table, the syntax they must use is:

```
SELECT  *
FROM    scott.employees;
```

Alternatively, they can create a synonym for the table and SELECT from the synonym:

```
CREATE SYNONYM emp FOR scott.employees;
SELECT * FROM emp;
```

Note: DBAs generally allocate system privileges; any user who owns an object can grant object privileges.

Instructor Note

Please read the Instructor Note at the end of this lesson.

Using the WITH GRANT OPTION and PUBLIC Keywords

- Give a user authority to pass along privileges.

```
GRANT  select, insert
ON     departments
TO     scott
WITH   GRANT OPTION;
Grant succeeded.
```

- Allow all users on the system to query data from Alice's DEPARTMENTS table.

```
GRANT  select
ON     alice.departments
TO     PUBLIC;
Grant succeeded.
```

ORACLE

The WITH GRANT OPTION Keyword

A privilege that is granted with the WITH GRANT OPTION clause can be passed on to other users and roles by the grantee. Object privileges granted with the WITH GRANT OPTION clause are revoked when the grantor's privilege is revoked.

The example on the slide gives user Scott access to your DEPARTMENTS table with the privileges to query the table and add rows to the table. The example also allows Scott to give others these privileges.

The PUBLIC Keyword

An owner of a table can grant access to all users by using the PUBLIC keyword.

The second example allows all users on the system to query data from Alice's DEPARTMENTS table.

Instructor Note

If a statement does not use the full name of an object, the Oracle server implicitly prefixes the object name with the current user's name (or schema). If user Scott queries the DEPARTMENTS table, for example, the system selects from the SCOTT.DEPARTMENTS table.

If a statement does not use the full name of an object, and the current user does not own an object of that name, the system prefixes the object name with PUBLIC. For example, if user Scott queries the USER_OBJECTS table, and Scott does not own such a table, the system selects from the data dictionary view by way of the PUBLIC.USER_OBJECTS public synonym.

Confirming Privileges Granted

Data Dictionary View	Description
ROLE_SYS_PRIVS	System privileges granted to roles
ROLE_TAB_PRIVS	Table privileges granted to roles
USER_ROLE_PRIVS	Roles accessible by the user
USER_TAB_PRIVS_MADE	Object privileges granted on the user's objects
USER_TAB_PRIVS_RECD	Object privileges granted to the user
USER_COL_PRIVS_MADE	Object privileges granted on the columns of the user's objects
USER_COL_PRIVS_RECD	Object privileges granted to the user on specific columns
USER_SYS_PRIVS	Lists system privileges granted to the user

ORACLE

Confirming Granted Privileges

If you attempt to perform an unauthorized operation, such as deleting a row from a table for which you do not have the DELETE privilege, the Oracle server does not permit the operation to take place.

If you receive the Oracle server error message "table or view does not exist," you have done either of the following:

- Named a table or view that does not exist
- Attempted to perform an operation on a table or view for which you do not have the appropriate privilege

You can access the data dictionary to view the privileges that you have. The chart on the slide describes various data dictionary views.

How to Revoke Object Privileges

- **You use the `REVOKE` statement to revoke privileges granted to other users.**
- **Privileges granted to others through the `WITH GRANT OPTION` clause are also revoked.**

```
REVOKE {privilege [, privilege...]|ALL}
ON     object
FROM   {user[, user...]|role|PUBLIC}
[CASCADE CONSTRAINTS];
```

ORACLE

Revoking Object Privileges

You can remove privileges granted to other users by using the REVOKE statement. When you use the REVOKE statement, the privileges that you specify are revoked from the users you name and from any other users to whom those privileges were granted through the WITH GRANT OPTION clause.

In the syntax:

CASCADE CONSTRAINTS	is required to remove any referential integrity constraints made to the object by means of the REFERENCES privilege

For more information, see *Oracle9i SQL Reference*, "REVOKE."

Revoking Object Privileges

As user Alice, revoke the SELECT and INSERT privileges given to user Scott on the DEPARTMENTS table.

```
REVOKE  select, insert
ON      departments
FROM    scott;
Revoke succeeded.
```

ORACLE

Revoking Object Privileges (continued)

The example on the slide revokes SELECT and INSERT privileges given to user Scott on the DEPARTMENTS table.

Note: If a user is granted a privilege with the WITH GRANT OPTION clause, that user can also grant the privilege with the WITH GRANT OPTION clause, so that a long chain of grantees is possible, but no circular grants are permitted. If the owner revokes a privilege from a user who granted the privilege to other users, the revoking cascades to all privileges granted.

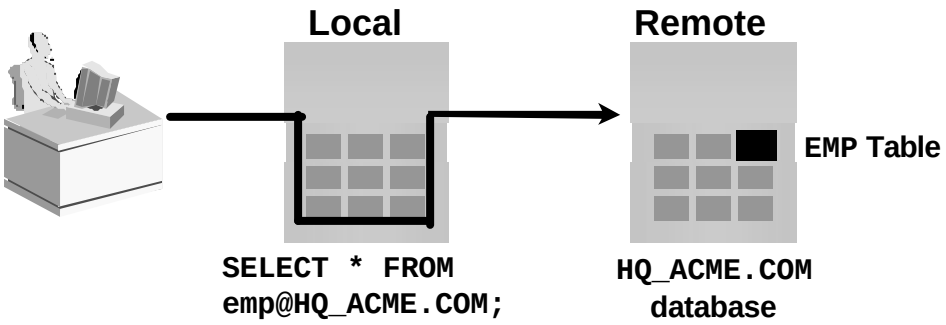
For example, if user A grants SELECT privilege on a table to user B including the WITH GRANT OPTION clause, user B can grant to user C the SELECT privilege with the WITH GRANT OPTION clause as well, and user C can then grant to user D the SELECT privilege. If user A revokes privilege from user B, then the privileges granted to users C and D are also revoked.

Instructor Note

Revoking system privileges is not within the scope of this lesson. For information on this topic refer to: *Oracle9i SQL Reference*, "REVOKE *system_privileges_and_roles*."

Database Links

A database link connection allows local users to access data on a remote database.



ORACLE

Database Links

A database link is a pointer that defines a one-way communication path from an Oracle database server to another database server. The link pointer is actually defined as an entry in a data dictionary table. To access the link, you must be connected to the local database that contains the data dictionary entry.

A database link connection is one-way in the sense that a client connected to local database A can use a link stored in database A to access information in remote database B, but users connected to database B cannot use the same link to access data in database A. If local users on database B want to access data on database A, they must define a link that is stored in the data dictionary of database B.

A database link connection gives local users access to data on a remote database. For this connection to occur, each database in the distributed system must have a unique global database name. The global database name uniquely identifies a database server in a distributed system.

The great advantage of database links is that they allow users to access another user's objects in a remote database so that they are bounded by the privilege set of the object's owner. In other words, a local user can access a remote database without having to be a user on the remote database.

The example shows a user `SCOTT` accessing the `EMP` table on the remote database with the global name `HQ.ACME.COM`.

Note: Typically, the DBA is responsible for creating the database link. The dictionary view `USER_DB_LINKS` contains information on links to which a user has access.

Database Links

- **Create the database link.**

```
CREATE PUBLIC DATABASE LINK hq.acme.com
USING 'sales';
Database link created.
```

- **Write SQL statements that use the database link.**

```
SELECT *
FROM emp@HQ.ACME.COM;
```

ORACLE

Using Database Links

The example shown creates a database link. The `USING` clause identifies the service name of a remote database.

Once the database link is created, you can write SQL statements against the data in the remote site. If a synonym is set up, you can write SQL statements using the synonym.

For example:

```
CREATE PUBLIC SYNONYM HQ_EMP FOR emp@HQ.ACME.COM;
```

Then write a SQL statement that uses the synonym:

```
SELECT * FROM HQ_EMP;
```

You cannot grant privileges on remote objects.

Instructor Note

Let the students know that using distributed databases encompasses much more than what is shown here. If the students want more information, refer them to the *Oracle9i Concepts*, "Distributed Database Concepts."

Summary

In this lesson, you should have learned about DCL statements that control access to the database and database objects:

Statement	Action
CREATE USER	Creates a user (usually performed by a DBA)
GRANT	Gives other users privileges to access the your objects
CREATE ROLE	Creates a collection of privileges (usually performed by a DBA)
ALTER USER	Changes a user's password
REVOKE	Removes privileges on an object from users

ORACLE

Summary

DBAs establish initial database security for users by assigning privileges to the users.

- The DBA creates users who must have a password. The DBA is also responsible for establishing the initial system privileges for a user.
- Once the user has created an object, the user can pass along any of the available object privileges to other users or to all users by using the GRANT statement.
- A DBA can create roles by using the CREATE ROLE statement to pass along a collection of system or object privileges to multiple users. Roles make granting and revoking privileges easier to maintain.
- Users can change their password by using the ALTER USER statement.
- You can remove privileges from users by using the REVOKE statement.
- With data dictionary views, users can view the privileges granted to them and those that are granted on their objects.
- With database links, you can access data on remote databases. Privileges cannot be granted on remote objects.

Practice 13 Overview

This practice covers the following topics:

- **Granting other users privileges to your table**
- **Modifying another user's table through the privileges granted to you**
- **Creating a synonym**
- **Querying the data dictionary views related to privileges**

ORACLE

Practice 13 Overview

Team up with other students for this exercise about controlling access to database objects.

Instructor Note

For this practice, divide the students into teams, and then pair off the teams so that half are Team 1s and the other half are Team 2s.

1. What privilege should a user be given to log on to the Oracle Server? Is this a system or an object privilege?

   _____

   _

2. What privilege should a user be given to create tables?

   _____

   _

3. If you create a table, who can pass along privileges to other users on your table?

   _____

   _

4. You are the DBA. You are creating many users who require the same system privileges. What should you use to make your job easier?

   _____

   _

5. What command do you use to change your password?

   _____

   _

6. Grant another user access to your DEPARTMENTS table. Have the user grant you query

7.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

8. Add a new row to your DEPARTMENTS table. Team 1 should add Education as department number 500. Team 2 should add Human Resources department number 510. Query the other team's table.
9. Create a synonym for the other team's DEPARTMENTS table.

Practice 13 (continued)

10. Query all the rows in the other team's DEPARTMENTS table by using your synonym.

Team 1 SELECT statement results:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
500	Education		

9 rows selected.

Team 2 SELECT statement results:

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700
510	Human Resources		

9 rows selected.

11. Query the USER_TABLES data dictionary to see information about the tables that you own.

TABLE_NAME
COUNTRIES
DEPARTMENTS
DEPT
EMP
EMPLOYEES
JOBS
JOB_GRADES
JOB_HISTORY
LOCATIONS
REGIONS

10 rows selected.

12. Query the ALL_TABLES data dictionary view to see information about all the tables that you can access. Exclude tables that you own.

    Note: Your list may not exactly match the list shown below.

13

TABLE_NAME	OWNER
DEPARTMENTS	owner

Instructor Note (for pages 13-14)

Let students know that the *privilege(col,col)* syntax can be used only with `UPDATE`. Most students try to use this syntax with `SELECT` as shown below:

```
GRANT SELECT(salary,last_name)
ON employees TO scott;
```

The above syntax returns the error `ERROR at line 1:ORA-00969: missing ON keyword.`

Instructor Note

Let students know about fine-grained access control. Using fine-grained access control, you can implement security policies with functions and then associate those security policies with tables or views. The database server automatically enforces those security policies, no matter how the data is accessed (for example, by ad hoc queries).

You can:

- Use different policies for `SELECT`, `INSERT`, `UPDATE`, and `DELETE` commands
- Use security policies only where you need them (for example, on salary information)
- Use more than one policy for each table, including building on top of base policies in packaged applications

For the implementation of fine-grained access control, you may need to use functions or packages in PL/SQL. The PL/SQL `DBMS_RLS` package enables you to administer your security policies. Using this package, you can add, drop, enable, disable, and refresh the policies you create. For more information on implementing fine-grained access control, refer to: *Oracle9i Concepts*, "Fine-Grained Access Control."