Trafodion Blueprint Design Document:
Privilege and Authorization Updates

Title	Privilege and Authorization Updates
Date	January 2015
Author	Trafodion Security Team
Audience	Open Source community
Abstract	This blueprint describes additional privileges and privilege checks that will be added to Trafodion.

Document History

Document Version	Date	Changes
1.0	10/31/2014	• Initial revision
1.1	11/17/2014	• Incorporated comments from 11/13/2014 review
1.2	12/13/2014	• Minor changes reflecting the final implementation

Legal Notice

Table of Contents

Introduction

This blueprint describes additional privilege checks required in Trafodion to complete the authorization task including:

- Object privilege checks
- DDL privilege checks
- Utility privilege checks

There is a concurrent project called "ANSI schemas" that adds another layer of privilege checking. Please see blueprint <security-ansi-schemas> for details.

Externals

Trafodion SQL supports two types of schemas - private and shared schemas. Any user can create objects in shared schemas. Only the schema owner can create objects in private schemas.

Object privilege checks

The following objects require additional privilege checks. In some cases privilege checks are missing and in other cases, privileges are partially implemented.

- RI constraints
- Sequence Generators
- Libraries
- User defined routines

RI (Referential Integrity) constraints

Referential integrity defines a relationship between two tables to enforce consistent data. That is, data in one table (the referencing table) must exist in another table (the referenced table). This relationship is enforced through a foreign key. A foreign key is a column (or set of columns) in one table that uniquely identifies a row of another table.

For example, you can define an RI constraint to prevent an insert into the referencing table if the column value does not exist in the referenced table.

In order to create an RI constraint you need to:

- Be DB__ROOT or
- Be owner of the referencing and referenced tables or
- Have correct privileges on the referencing and referenced table:
    - For the referencing table:
        - CREATE/CREATE_TABLE (when creating RI constraints during CREATE TABLE) component privilege or
        - ALTER/ALTER_TABLE (when creating RI constraints during ALTER TABLE) component privilege
    - For the referenced table:
        - REFERENCES (or ALL) privilege on the referenced table via a user or role

If you try to REVOKE the REFERENCES privilege from a user or role the REVOKE should fail if an RI constraint was created based on this privilege. The RI constraint needs to be dropped before the REVOKE can succeed.

Example:

User1:

```
CREATE TABLE dept (dept_no int not null primary key,
                   Dept_name varchar (200));
```

User2:

```
CREATE TABLE empl (empl_no int not null primary key,
                   Empl_dept_no int not null,
                   Empl_name varchar (200),
                   constraint fk foreign key (empl_dept_no)
                   references dept);
```

User2 is unable to create empl because of no REFERENCES privilege on the dept table.

User1:

```
GRANT REFERENCES on dept to user2;
```

Now user2 can create the RI constraint

User1:

```
REVOKE REFERENCES on dept from user2;
```

The REVOKE should fail because of User2's dependent object - fk. Constraint fk should be dropped and REVOKE reissued.

Sequence generators

A sequence generator is a database object that allows users to generate unique IDs. It is its own object and not tied other objects such as tables or views.

Trafodion supports internal sequence generators through columns specified as IDENTITY columns. When users create a table with an IDENTITY column, Trafodion creates a unique internal sequence generator for the column. Users are not necessarily aware that an internal sequence generator has been created, users cannot associate an external sequence generator with an IDENTITY column, and there are special no privileges required to use IDENTITY columns in CREATE requests.

In order to create a sequence generator you need to:

- Be DB__ROOT or
- Be in a shared schema or
- Be the private schema owner or

- Have the CREATE/CREATE_SEQUENCE component level privilege

In order to drop a sequence generator you need to:

- Be DB__ROOT or
- Be the owner of the sequence or
- Have the DROP/DROP_SEQUENCE component level privilege

In order to alter a sequence generator you need to:

- Be DB__ROOT or
- Be the owner of the sequence or
- Have the ALTER/ALTER_SEQUENCE component level privilege

In order to use a sequence generator in a SQL statement you need to:

- Be DB__ROOT or
- Be owner of the sequence or
- Have been granted the USAGE (or ALL) privilege - via a user or role

If you REVOKE the USAGE privilege from a user or role, requests that used to work may now fail because of lack of privilege.

Examples:

User 1:

```
CREATE TABLE t1 (a int not null primary key, b int not null);
INSERT INTO test024t1 values (1,1), (2,2);
CREATE TABLE t2 (z int not null primary key, a int not null, b int not null);

CREATE SEQUENCE seq1;
CREATE SEQUENCE seq4 maxvalue 3 no cache;
```

User 2:

```
SELECT * from t1 where a < seqnum(seq4);
INSERT into t2 select seqnum(seq1), a, b from t1;
```

User2 is unable to perform these statements because of no privilege

User1:

```
GRANT USAGE on sequence seq1 to user2;
GRANT USAGE on sequence seq4 to user2;
```

User2 is now able to perform these operations.

Libraries

A library is a repository of files that contains code required for user defined routines. User defined routines consist of functions and stored procedures. In order to perform any library commands you must be granted the MANAGE_LIBRARY component privileges. In addition:

In order to create a library you need to:

- Be DB__ROOT or
- Have the MANAGE_LIBRARY privilege and
    - Be in a shared schema or
    - Be the schema owner or
    - Have the CREATE/CREATE_LIBRARY component level privilege

In order to drop a library you need to:

- Be DB__ROOT or
- Be the owner of the library or
- Have the DROP/DROP_LIBRARY component level privilege

In order to use a library you need to:

- Be DB__ROOT or
- Be the owner of the library or
- Have been granted the USAGE (or ALL) privilege via a user or role

If you try to REVOKE the USAGE privilege from a user or role, the REVOKE fails if a user defined function was created based on this privilege. The user defined function needs to be dropped before the REVOKE can succeed.

Examples:

User 1:

```
CREATE LIBRARY library1 file 'udrinfo.dll';
```

User2:

```
CREATE TABLE_MAPPING FUNCTION sessionize(colname char(10), timeintval int)
      returns (userid char(32), ts largeint, session_id largeint)
      external name 'SESSIONIZE'
      library library1;
```

User2 is unable to create the function because of no USAGE privilege

User1:

```
GRANT USAGE on library library1 to user2;
```

User2 is now able to create the function.

User defined routines

User defined routines consist of stored procedures, functions, and table_mapping functions.

In order to create a user defined routine you need to:

- Be DB__ROOT or
- Have the USAGE (or ALL) privilege on the library and
    - Be in a shared schema or
    - Be the private schema owner or
    - Have the CREATE/CREATE_ROUTINE component level privilege

In order to drop a routine you need to:

- Be DB__ROOT or
- Be the owner of the routine or
- Have the DROP/DROP_ROUTINE component level privilege

In order to alter a routine you need to:

- Be DB__ROOT or
- Be the owner of the routine or
- Have the ALTER/ALTER_ROUTINE component level privilege

In order to execute a routine you need to:

- Be DB__ROOT or
- Be the owner of the routine or
- Have been granted the EXECUTE (or ALL) privilege - directly on the user or through a granted role

When the routine executes, it executes as the Trafodion ID. Once invoker and definer rights are implemented, this changes.

If you REVOKE the EXECUTE privilege from a user or role, requests that used to work may now fail because of lack of privilege.

Examples:

User 1:

```
CREATE LIBRARY library1 file 'udrinfo.dll';
CREATE TABLE_MAPPING FUNCTION sessionize(colname char(10), timeintval int)
      returns (userid char(32), ts largeint, session_id largeint)
      external name 'SESSIONIZE'
      library library1;
```

User2:

```
SELECT ts, userid, sessionid FROM UDF(sessionize("name", 60));
```

User2 is unable to perform this statement because of no EXECUTE privilege

User1:

```
GRANT EXECUTE on function sessionize to user2;
```

User2 is now able to execute the SELECT statement.

Privilege checks for DDL operations

To support additional privilege checks for DDL operations, the following new component privilege will be added:

```
CREATE COMPONENT PRIVILEGE MANAGE_COMPONENTS AS 'MC' ON SQL_OPERATIONS SYSTEM
```

```
  DETAIL 'Allow grantee to register and unregister components and add
  component operations';
GRANT COMPONENT PRIVILEGE MANAGE_COMPONENTS
 ON SQL_OPERATIONS TO DB__ROOTROLE WITH GRANT OPTION;
```

In order to perform DDL operations you must be authorized.

- For the following operations, you must be DB__ROOT, be the table owner, or have the ALTER_TABLE or ALTER component privilege:

  ALTER TABLE ADD COLUMN
  ALTER TABLE ADD CONSTRAINT CHECK
  ALTER TABLE ADD CONSTRAINT PRIMARY KEY
  ALTER TABLE ADD CONSTRAINT REFERENTIAL INTEGRITY
  ALTER TABLE ADD CONSTRAINT UNIQUE
  ALTER TABLE DISABLE INDEX
  ALTER TABLE DROP COLUMN
  ALTER TABLE DROP CONSTRAINT
  ALTER TABLE ENABLE INDEX
  ALTER TABLE RENAME
  CREATE_INDEX - you can also create an index with the CREATE_INDEX privilege

- For the following operation, you must be DB__ROOT or have the MANAGE_USER component privilege:

  ALTER USER
  REGISTER USER
  UNREGISTER USER

- For the following operations, you must be DB__ROOT or have the MANAGE_ROLE component privilege:

  CREATE ROLE
  DROP ROLE
  GRANT and REVOKE role on behalf of the owner

- For the following operations, you must be DB__ROOT or have the MANAGE_COMPONENT privilege:

  CREATE COMPONENT PRIVILEGE
  DROP COMPONENT PRIVILEGE
  REGISTER COMPONENT
  UNREGISTER COMPONENT

- For the following operations, you must be DB__ROOT, be in a shared schema, be the private schema owner, or have the CREATE_<type> or CREATE component privilege:

  CREATE INDEX - you can also create an index with the ALTER_TABLE privilege

CREATE_LIBRARY - also need the MANAGE_LIBRARY component privilege
CREATE_ROUTINE - also need the USAGE privilege on the library
CREATE_SCHEMA
CREATE_SEQUENCE
CREATE_TABLE
CREATE_VIEW

- For the following operations, you must be DB__ROOT or have the DROP_<type> or DROP component privilege:

  DROP INDEX
  DROP LIBRARY
  DROP ROUTINE
  DROP SCHEMA
  DROP SEQUENCE
  DROP TABLE
  DROP VIEW

- For the following operations, you must be DB__ROOT or have the ALTER_<type> or ALTER component privilege:

  ALTER LIBRARY
  ALTER ROUTINE
  ALTER SEQUENCE
  ALTER VIEW

Privilege checks for utility operations

The following utility operations are supported:

- CONTROL statements
- Library operations
- LOAD/UNLOAD
- POPULATE INDEX
- PURGEDATA
- SET statements
- SHOW statements
- UPDATE STATISTICS

To support these utilities, the following new component privileges will be added:

```
CREATE COMPONENT PRIVILEGE MANAGE_LIBRARY AS 'ML' ON SQL_OPERATIONS SYSTEM
  DETAIL 'Allow grantee to load user defined routines in Trafodion';
GRANT COMPONENT PRIVILEGE MANAGE_LIBRARY
 ON SQL_OPERATIONS TO DB__ROOTROLE WITH GRANT OPTION;

CREATE COMPONENT PRIVILEGE MANAGE_LOAD AS 'MT' ON SQL_OPERATIONS SYSTEM
  DETAIL 'Allow grantee to perform load and unload requests';
GRANT COMPONENT PRIVILEGE MANAGE_LOAD
```

```
 ON SQL_OPERATIONS TO DB__ROOTROLE WITH GRANT OPTION;

CREATE COMPONENT PRIVILEGE MANAGE_STATISTICS AS 'MS' ON SQL_OPERATIONS SYSTEM
  DETAIL 'Allow grantee to perform statistic requests';
GRANT COMPONENT PRIVILEGE MANAGE_STATISTICS ON SQL_OPERATIONS TO DB__ROOTROLE
 WITH GRANT OPTION;

CREATE COMPONENT PRIVILEGE SHOW AS 'SW' ON SQL_OPERATIONS SYSTEM
  DETAIL 'Allow grantee to perform explain, get, invoke and show requests';
GRANT COMPONENT PRIVILEGE SHOW ON SQL_OPERATIONS TO DB__ROOTROLE
 WITH GRANT OPTION;
GRANT COMPONENT PRIVILEGE SHOW ON SQL_OPERATIONS TO "PUBLIC";

The SHOW privilege has been granted to PUBLIC by default.  If this is not
desired, the REVOKE COMPONENT PRIVILEGE ON SQL_OPERATIONS FROM "PUBLIC"; can
be performed.
```

CONTROL statements

CONTROL statements consist of:

- CONTROL QUERY DEFAULT
- CONTROL QUERY SHAPE
- CONTROL SESSION
- CONTROL TABLE

Any authenticated user can perform these commands; no special privilege checks are performed.

Library operations

A library is a database object that maps to a physical file on the Trafodion platform. The library's physical file is either a JAR file for running stored procedures in Java (SPJs) or a DLL (or .so file) for user-defined functions in C and C++. Libraries exist to provide greater security for JAR and DLL files. Libraries are database objects whose access is controlled using standard SQL security. JAR and DLL files are controlled by Linux security. Deploying a JAR or DLL file to a Trafodion instance requires creating a library, and users must have the required privileges for creating libraries.

The process of creating a library involves copying the JAR or DLL file to the Trafodion platform by the creator of the library. During the create process, Trafodion copies the library into a secure location on every Trafodion node and creates a library that points to the code file in a schema of the database. To create a library in a schema, users must have the MANAGE_LIBRARY and the ability to create library objects in the schema. To create a UDR that references the library, a user must have the USAGE privilege on the library and the ability to create objects in the schema. Users who have the necessary privileges to create libraries are prevented from uploading a code file that is already in use by another library in the catalog. This restriction prevents users from accidentally overwriting a code file that has the same name.

LOAD/UNLOAD (bulk)

The LOAD statement uses the Trafodion Bulk Loader to load data from a source table, either a Trafodion table or a Hive table, into a target Trafodion table. The Trafodion Bulk Loader prepares and loads HFiles directly in the region servers and bypasses the write path and the cost associated with it. The write path begins at a client, moves to a region server, and ends when data eventually is written to an HBase data file called an HFile.

- To perform the LOAD you must:
  - Be DB__ROOT or
  - Be the target table owner or
  - Have correct privileges including
    - SELECT and INSERT privileges on the Target table.
    - DELETE privilege on the target table if TRUNCATE is specified or
  - Have the MANAGE_LOAD component privilege

The UNLOAD statement unloads data from Trafodion tables into an HDFS location that you specify. Extracted data can be either compressed or uncompressed based on what you choose.

- To perform the UNLOAD operation you must:
  - Be DB__ROOT or
  - Be the target table owner or
  - Have the SELECT privilege on the target table or
  - Have the MANAGE_LOAD component privilege

POPULATE INDEX

The POPULATE INDEX statement performs a fast INSERT of data into an index from the parent table.

To perform the POPULATE INDEX operation, you must

- Be DB__ROOT or
- Be the table owner or
- Have the SELECT and INSERT (or ALL) privileges on the associated table or

PURGEDATA

The PURGEDATA statement performs a fast delete of data from a table.

To perform the PURGEDATA operation, you must

- Be DB__ROOT or
- Be the table owner or
- Have the SELECT and DELETE (or ALL) privileges on the associated table

SET statements

SET statements consist of:

- SET CATALOG
- SET SCHEMA
- SET (RESET) PARSERFLAGS
- SET (RESET) ENVVARS
- SET TABLE
- SET SESSION

You must be DB__ROOT to set or reset parserflags and envvars. For all other commands, any authenticated user can perform these commands; no special privilege checking is performed.

SHOW statements

SHOW statements consist of commands that fall in a number of categories.

Statements against an object include:

- EXPLAIN
- INVOKE
- SHOW TrafCi command
- SHOWDDL
- SHOWPLAN
- SHOWSHAPE
- SHOWSTATS - you can also perform SHOWSTATS if you have the MANAGE_STATISTICS privilege

To perform SHOW statements for objects you must:

- Be DB__ROOT or
- Be the object owner or
- Have been granted the SHOW component privilege or
- Have select privilege on the target object

This task was only partially completed for release 1.0. Only table variants of SHOWDDL and INVOKE have been implemented.

The GET command displays a wide variety of information. The aspiration goal is to only display items in GET commands where the current user as some privilege. For example, "get tables in schema <schema>" only returns tables that the current user has some privilege. This is independent on whether you have the SHOW component privilege. However, this is a complex task so initially, to perform a GET command you must:

- Be DB__ROOT or
- Have been granted the SHOW component privilege

Session commands include:

- SHOWCONTROL
- SHOWLEAKS
- SHOW SET
- SHOW TRANSACTION

Any authenticated user can perform these commands, no privilege checks are performed.

UPDATE STATISTICS

The statistics statements display and update histogram statistics for one or more groups of columns for a table. These statistics are used to devise optimized access plans.

To perform UPDATE STATISTICS you must:

- Be DB__ROOT or
- Be the target table owner or
- Have been granted the MANAGE_STATISTICS component privilege