Framework Deployment Guide

Minimum Required Permissions and Privileges

5/3/2025

Contents

Minimum Required Permissions and Privileges

This section describes the minimum permissions required to install and run Management Framework components. For information about minimum permissions required for other Genesys components, refer to product- or component-specific documentation.

Minimum System Permissions

The following table provides the minimum system permissions required to install and run Framework components.

Component	Minimum Permissions (UNIX)	Minimum Permissions (Windows)
Configuration Server	Users group	Administrators group [a]
Solution Control Server	Users group	Administrators group
Message Server	Users group	Administrators group
SNMP Master Agent	Users group	Administrators group
Local Control Agent [b]	`root`	Administrators group

a. The user account for the running process is usually determined by the user or object that started the process. For example, if a process is started by LCA, then the process inherits its permissions from LCA.

b. `root` or Administrators permission is required to install the component because, during installation, it updates the startup file and registry.

After a component is installed, you can update the component to start under a different user account with lower privileges. However, before doing so, make sure that you updated the working directories with the correct read and write permissions.

Example

To run LCA as a non-root user, do one of the following, depending on your operating system:

On UNIX

Create startup scripts for LCA that set up LCA to run under the non-root user. For these scripts, it is assumed that LCA is installed in **/home/genesys/GCTI**, and the name of the non-root user is `genesys`. See LCA Startup Script-gctilca for examples of the script. To install the startup script, put it in the directory **/etc/rc.d/init.d/** and run one or both of the following command:

```
chkconfig -add gctilca
```

On Windows

Change the account associated with the LCA service. One way to do this is through Windows Administrative Services, as follows:

1. Go to **Start** > **Settings** > **Control Panel** > **Administrative Services** > **Services**, right-click **LCA**, and select **Properties**.
2. Open the **Log On** tab and in the **Log on as** section, select **This account**, and change the account associated with the LCA service.

Minimum Database Privileges

This section describes the minimum required database privileges required to deploy and access the Configuration Database and Log Database.

Important

PostgreSQL 10.x contains an upgraded SCRAM (Salted Challenge Response Authentication Mechanism) authentication. If you are going to use PostgreSQL 10.x and the new SCRAM, set the password encryption algorithm before running any of the PostgreSQL queries mentioned on this page, as follows:

```
SET password_encryption TO 'scram-sha-256'
```

Configuration Database

This section describes the minimum required database privileges required to deploy and access the Configuration Database.

Deploying the Configuration Database

A database user that accesses the Configuration Database on behalf of Configuration Server, that is, the user identified in the Configuration Server configuration file, requires basic database privileges, as defined in this section.

Important

> When a new database is created, the following instructions and examples assume that the database is accessed by the same user that created it, and that any initialization scripts are run by that same user. If another user runs the scripts, they might not work as described herein.

To deploy the Configuration Database, the user must possess the following minimum required database privileges:

MS SQL

For MS SQL 2000, grant the `public` role to the new database user on the **Database Access** tab of the **SQL Server Login Properties** dialog box for the new user. Grant the following privileges to the new user:

```
GRANT CREATE TABLE TO <DB user>
GRANT CREATE PROCEDURE TO <DB user>
```

For MS SQL 2005 and later, grant the `public` and `db_owner` roles to the new database user.

Oracle

After the new database user is created, grant the following privileges:

```
GRANT CONNECT TO <DB user>
GRANT CREATE TABLE TO <DB user>
GRANT UNLIMITED TABLESPACE TO <DB user>
GRANT CREATE PROCEDURE TO <DB user>
```

PostgreSQL

From `pgAdmin`, create a User and grant the following privileges:

- Can log in
- Can create database object

Or, you can execute the following query:

```
CREATE ROLE <DB user> LOGIN ENCRYPTED PASSWORD <encrypted password> NOINHERIT CREATEDB VALID UNTIL 'infinity';
```

To configure client authentication, update the **pg_hba.conf** file, located in the data directory under the PostgreSQL installation folder. For example:

```
host GCTI_Test gctitest <IP address1>/32 trust
host GCTI_Test gctitest <IP address2>/32 trust
```

This enables the DB user `gctitest` to connect to the `GCTI_Test` database from the hosts `IPaddress1` and `IPaddress2`.

Accessing the Configuration Database

A database user that accesses the Configuration Database on behalf of Configuration Server, that is, the user identified in the Configuration Server configuration file, requires basic database privileges, as defined in this section.

For a user to access the Configuration Database through Configuration Server, the following database privileges are required:

MS SQL

Grant the `public` role to the new database user and grant the following privileges:

```
GRANT SELECT TO <DB User>
GRANT INSERT TO <DB User>
GRANT UPDATE TO <DB User>
GRANT DELETE TO <DB User>
```

Oracle

Create the new user, as follows:

```
CREATE USER <DB User> IDENTIFIED BY <Password>
```

Grant the following permissions to the new user, as follows:

```
GRANT CONNECT TO <DB user>
BEGIN
  FOR x IN (SELECT owner, table_name FROM all_tables WHERE owner='<Table Owner>')
  LOOP
    EXECUTE IMMEDIATE 'GRANT SELECT ON ' || x.owner || '.' || x.table_name || ' TO <DB user>';
    EXECUTE IMMEDIATE 'GRANT INSERT ON ' || x.owner || '.' || x.table_name || ' TO <DB user>';
    EXECUTE IMMEDIATE 'GRANT DELETE ON ' || x.owner || '.' || x.table_name || ' TO <DB user>';
    EXECUTE IMMEDIATE 'GRANT UPDATE ON ' || x.owner || '.' || x.table_name || ' TO <DB user>';
    EXECUTE IMMEDIATE 'CREATE SYNONYM <DB user>.' || x.table_name || ' FOR ' || x.owner || '.' || x.table_name;
 END LOOP;
END;
/
```

PostgreSQL

From pgAdmin, create a new role with the following privilege:

- Can Login

Or, you can execute the following query:

```
CREATE ROLE <DB user> LOGIN ENCRYPTED PASSWORD <encrypted password> NOINHERIT VALID UNTIL 'infinity';
```

Log into the database and grant permissions to the user, as follows:

```
GRANT SELECT ON ALL TABLES IN SCHEMA public TO <DB User>;
GRANT INSERT ON ALL TABLES IN SCHEMA public TO <DB User>;
GRANT UPDATE ON ALL TABLES IN SCHEMA public TO <DB User>;
GRANT DELETE ON ALL TABLES IN SCHEMA public TO <DB User>;
```

To configure client authentication, update the **pg_hba.conf** file located in the data directory under the PostgreSQL installation folder. For example:

```
host GCTI_Test gctitest <IP address1>/32 trust
host GCTI_Test gctitest <IP address2>/32 trust
```

Log Database

This section describes the minimum required database privileges required to deploy and access the Log Database.

Deploying the Log Database

When a new database is created, the following instructions and examples assume that the database is created under the same user that created it, and that any initialization scripts are run by that same user. If another user runs the scripts, they might not work as described herein.

MS SQL

For MS SQL 2000, grant the `public` role to the new database user on the **Database Access** tab of the **SQL Server Login Properties** dialog box for the new user. Grant the following privileges:

```
GRANT CREATE TABLE TO <DB user>
GRANT CREATE PROCEDURE TO <DB user>
```

For MS SQL 2005 and later, grant the `public` and `db_owner` roles to the new database user.

Oracle

After the new database user is created, grant the necessary privileges as follows:

```
GRANT CONNECT TO <DB user>
GRANT CREATE TABLE TO <DB user>
GRANT UNLIMITED TABLESPACE TO <DB user>
GRANT CREATE PROCEDURE TO <DB user>
GRANT CREATE SEQUENCE TO <DB user>
```

PostgreSQL

From pgAdmin, grant the following privileges:

- Can Login
- Can create database object

Or, you can execute the following query:

```
CREATE ROLE <DB user> LOGIN ENCRYPTED PASSWORD '<encrypted password>' NOINHERIT CREATEDB VALID UNTIL 'infinity';
```

To configure client authentication, update the **pg_hba.conf** file, located in the data directory under the PostgreSQL installation folder. For example:

```
host GCTI_Test gctitest <IP address1>/32 trust
 host GCTI_Test gctitest <IP address2>/32 trust
```

This enables the DB user gctitest to connect to the GCTI_Test database from the hosts `<IPaddress1>` and `<IPaddress2>`.

Accessing the Log Database

A database user that accesses the Log Database on behalf of Message Server, that is, the user identified in the Message Server Database Access Point, requires basic database privileges, as defined in this section.

MS SQL

For MS SQL 2000, grant the `public` role to the new database user on the **Database Access** tab of the **SQL Server Login Properties** dialog box for the new user.
For MS SQL 2005 and later, grant the `public` role to the new database user.
Grant the following privileges to the user:

```
GRANT SELECT TO <DB user>
GRANT INSERT TO <DB user>
GRANT EXECUTE TO <DB user>
```

Oracle

Create the new user, as follows:

```
CREATE USER <DB User> IDENTIFIED BY <Password>
```

Grant the following permissions to the new user:

```
GRANT CONNECT TO <DB user>
GRANT ALL PRIVILEGES ON <Table Owner>.SQ_ATTR_ID TO <DB User>;
CREATE SYNONYM <DB user>.SQ_ATTR_ID FOR <Table Owner>.SQ_ATTR_ID;
GRANT EXECUTE on <Table Owner>.G_LOG_GET_RANGE to <DB User>;
CREATE SYNONYM <DB user>.G_LOG_GET_RANGE FOR <Table Owner>.G_LOG_GET_RANGE;
BEGIN
  FOR x IN (SELECT owner, table_name FROM all_tables WHERE owner='<Table Owner>')
  LOOP
    EXECUTE IMMEDIATE 'GRANT SELECT ON ' || x.owner || '.' || x.table_name || ' TO <DB user>';
    EXECUTE IMMEDIATE 'GRANT INSERT ON ' || x.owner || '.' || x.table_name || ' TO <DB user>';
    EXECUTE IMMEDIATE 'CREATE SYNONYM <DB user>.' || x.table_name || ' FOR ' || x.owner || '.' || x.table_name;
  END LOOP;
END;
/
```

PostgreSQL

From pgAdmin, create a new role:

- Can Login

Or, you can execute the following query:

```
CREATE ROLE <DB user> LOGIN ENCRYPTED PASSWORD <encrypted password> NOINHERIT VALID UNTIL 'infinity';
```

After logging in to the database, grant the following permissions to the user:

```
GRANT SELECT ON ALL TABLES IN SCHEMA public TO <DB User>;
GRANT INSERT ON ALL TABLES IN SCHEMA public TO <DB User>;
GRANT UPDATE ON ALL TABLES IN SCHEMA public TO <DB User>;
GRANT ALL ON SEQUENCE public.sq_attr_id TO <DB User>;
```

To configure client authentication, update the **pg_hba.conf** file, located in the data directory under the PostgreSQL installation folder. For example:

```
host GCTI_Test gctitest <IP address1>/32 trust
host GCTI_Test gctitest <IP address2>/32 trust
```

Sample Scripts

This section contains sample script required to run LCA on UNIX under a non-root user.

LCA Startup Script-`gctilca`

The following is an example of a script to allow LCA to run under a non-root user.

```
#!/bin/bash
#
# chkconfig: 345 80 20
# description: run lca
#
# You should put this script to /etc/rc.d/init.d and run command:
# chkconfig --add gctilca
#GCTI home dir
GCTI=/home/genesys/GCTI
DIRNAME=LCA
HOMEDIR=$GCTI/$DIRNAME
USER=genesys
SCRIPTNAME=gctilca
HOME_USER=/home/genesys
PATH=/sbin:/bin:/usr/bin:/usr/sbin
prog=lca
RETVAL=0
if [ ! -x $HOMEDIR/$prog ]; then
exit 1
fi
# Source function library.
. /etc/rc.d/init.d/functions
start () {
echo -n $"Starting $SCRIPTNAME: "
if [ -e /var/lock/subsys/$prog ]; then
echo -n $"$SCRIPTNAME is already running.";
failure $"cannot start $SCRIPTNAME: $SCRIPTNAME already running.";
echo
return 1
fi
daemon --user=$USER ". $HOME_USER/.bash_profile ; cd $HOMEDIR ;
   ./run.sh >/dev/null 2>/dev/null &"
sleep 1
CHECK=`ps -e | grep $prog | grep -v $SCRIPTNAME | awk '{print $4}'`
if [ "$CHECK" = "$prog" ]; then
RETVAL=0
else
RETVAL=1
fi
[ $RETVAL -eq "0" ] && touch /var/lock/subsys/$prog
echo
return $RETVAL
}
stop () {
echo -n $"Stopping $SCRIPTNAME: "
if [ ! -e /var/lock/subsys/$prog ]; then
echo -n $"$SCRIPTNAME is not running."
failure $"cannot stop $SCRIPTNAME: $SCRIPTNAME is not running."
echo
return 1;
fi
killproc $prog
RETVAL=$?
echo
[ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/$prog;
return $RETVAL
}
usage ()
{
echo "Usage: service $PROG {start|stop|restart}"
}
case $1 in
start)
start
;;
stop)
stop
;;
restart)
stop
start
;;
*)
usage ; RETVAL=2
;;
esac
exit $RETVAL
```