# deploy.SetCreateUser – Technical Documentation

Version: 2.0  
Date: 2025-08-01  
Page: 1 of 2

## Summary

This function handles creation of PostgreSQL roles in batch. It supports dry-run simulation, masked logging, role inheritance, and auto-generated fallback passwords.

## Parameters and Defaults

- `p\_usernames` TEXT[]: Required, list of new role names.  
- `p\_user\_password` TEXT: Password string or NULL for auto-password.  
- `p\_user\_inherit\_roles` TEXT[]: Optional list of roles to grant.  
- `p\_execute\_flag` BOOLEAN: FALSE = dry-run, TRUE = real execution.

## Flow Logic

1. Loop through each user in `p\_usernames`.  
2. Check if user already exists via `pg\_roles`.  
3. If not, generate a default or supplied password.  
4. Build SQL for CREATE ROLE ... WITH LOGIN PASSWORD.  
5. Mask password in logs (xxxxxxxxxxxxxx).  
6. Execute real SQL only if `p\_execute\_flag = TRUE`.  
7. Log action into `info.account\_log\_history`.  
8. Grant each user to roles in `p\_user\_inherit\_roles` using dynamic GRANT.

## Security & Logging

- Function logs every action to `info.account\_log\_history`.  
- Passwords are masked in the SQL command column.  
- Supports safe audit usage in dry-run mode.  
- Execution should be limited to trusted admin role.

## Limitations & Risks

- No rollback if GRANT fails.  
- No password policy enforcement.  
- No validation that roles in `p\_user\_inherit\_roles` actually exist.  
- Passwords are stored in RAM if passed directly in SQL call.

## References

- https://www.postgresql.org/docs/current/sql-createrole.html  
- https://www.postgresql.org/docs/current/sql-grant.html  
- https://www.postgresql.org/docs/current/plpgsql-control-structures.html

## TL;DR

Dynamic role creation function with masked logging, default password logic, and dry-run safety.