Security Runtime Users Reference

Control runtime user accounts.

Security RuntimeUsers (Reference): Dynamic user management during runtime, allowing user creation and authentication without modifying solution configuration.

RuntimeUsers provide:

- Dynamic user creation during runtime
- External database integration
- Active Directory/LDAP authentication
- Temporary or permanent user accounts
- Script-based user management

RuntimeUsers combined with SecurityUsers form the complete Solution Users.

RuntimeUsers vs SecurityUsers
Configuration Sources
RuntimeUsers Table Properties
Database Configuration
Script Management
Windows AD / LDAP Integration
Runtime Behavior
Security Configuration
Best Practices Checklist
Common Configurations
Troubleshooting
See Also

RuntimeUsers vs SecurityUsers

Aspect	SecurityUsers	RuntimeUsers
Creation	Design-time only	Runtime only
Storage	Solution file	External database
Engineering Access	Yes	No
Modify Solution	Yes	No
Runtime Access	Yes	Yes
Source	Internal	External/Scripts

Note: RuntimeUsers cannot access Engineering mode or modify solution configuration. They are application users only.

Configuration Sources

1. Script Creation

Create User

```
string error = await @Security.AddRuntimeUserAsync(
    "username",  // name
    "Operator",     // permissionsStr
    "password",    // password
    "",             // passwordHint
    "Enhanced",     // policyStr
    "",             // contactInfoEmail
    "",             // contactInfoPhone
    "",             // contactInfoCompleteName
    false           // oneTimePassword
);
```

2. External SQL Database

Configuration: Datasets DBs RuntimeUsers

- Default: SQLite database
- Supports: SQL Server, MySQL, PostgreSQL
- Auto-created table structure

3. AD/LDAP Integration

- Users validated against directory
- Created in memory only
- No database storage

RuntimeUsers Table Properties

Access: Security RuntimeUsers (read-only view)

Property	Description	Modifiable
Name	Unique username	Via script/DB
Password	Encrypted credential	Via script/DB
Permissions	Group assignments	Via script/DB
Policy	Security policy	Via script/DB
Blocked	Access denied flag	Via script/DB
Deleted	Soft delete marker	Via script/DB
InvalidAttempts	Failed login count	Auto-updated
ChangePasswordRequired	Force password change	Via script/DB
LastChangePasswordUTC_Ticks	Password change timestamp	Auto-updated
LastBlockedUserUTC_Ticks	Block timestamp	Auto-updated
Level	Hierarchical access	Via script/DB
Category	User classification	Via script/DB
ContactInfo	Email/phone	Via script/DB

Database Configuration

Default SQLite Structure

Location: `<SolutionPath>.dbRuntimeUsers`

Table automatically created with:

- User authentication fields
- Permission assignments
- Policy enforcement
- Audit tracking

Protect the Runtime Users database. By default, the Runtime Users database is an unencrypted SQLite file in the solution folder, or in the Path set under Execution Environment in Runtime Startup. Anyone who can write to that folder can replace the file, and the users in the replacement file can log on. Move the Runtime Users database to a server database with its own logon, such as SQL Server with encryption turned on, in **Datasets DBs RuntimeUsers**. Restrict write access to the solution folder and that Path folder.

Custom Database

1. Configure **Datasets DBs RuntimeUsers**
2. Set connection string
3. System creates table if missing
4. Maintain schema compatibility

Script Management

Creating Users

Creating Users

```
string error = await @Security.AddRuntimeUserAsync(
    "username",  // name
    "Operator",     // permissionsStr
    "password",    // password
    "",             // passwordHint
    "Enhanced",     // policyStr
    "",             // contactInfoEmail
    "",             // contactInfoPhone
    "",             // contactInfoCompleteName
    false           // oneTimePassword
);
```

Modifying Users

csharp

```
// Change password
@Security.ChangePassword("john.doe", "newPassword");

// Update permissions
@Security.SetUserPermissions("john.doe", "Operator,Supervisor");

// Block user
@Security.BlockUser("john.doe");
```

Deleting Users

csharp

```
// Soft delete (mark as deleted)
@Security.DeleteUser("john.doe", softDelete: true);

// Hard delete (remove from database)
@Security.DeleteUser("john.doe", softDelete: false);
```

Windows AD / LDAP Integration

Windows AD Authentication

Automatic Availability when:

- Solution runs on Windows
- Domain connectivity exists
- Port 3102 (default) accessible

Configuration:

1. Navigate to **Runtime Startup**
2. Enable **Use WA** checkbox
3. Set **PortWA**: 3102 (or custom)
4. Configure redundancy ports if needed

Connection Methods:

Rich Client:

```
Server: ServerName
Port: 3102 (AD port)
Windows Authentication: Enabled
```

Web Client URL:

```
http://server/fs-2024/TSmartClient.application?port1=3102&wa=true
```

LDAP Server Configuration

Setup:

1. Navigate to **Security RuntimeUsers**
2. Enter LDAP server in **AD/LDAP Server** field:

```
ldap://company.local:389
ldaps://secure.company.local:636
```

Connection String Examples:

Standard LDAP:

```
ldap://dc1.company.local:389
```

Secure LDAP:

```
ldaps://dc1.company.local:636
```

With Base DN:

```
ldap://dc1.company.local:389/DC=company,DC=local
```

Permission Mapping

User Resolution Order:

1. Check for exact username match in Security Users
2. Map Windows groups to permission groups
3. Apply Guest permissions if no match

Group Mapping Example:

```
Windows Group: Domain\Engineers
Permission Group: Engineering
Result: User gets Engineering permissions
```

LDAP Attributes Mapping:

LDAP Attribute	Solution Property
sAMAccountName	UserName
memberOf	Permissions (via groups)
displayName	Display name
mail	ContactInfo
department	UserGroup

Runtime Behavior

User Validation Order

1. Check SecurityUsers (internal)
2. Query RuntimeUsers database
3. Validate against AD/LDAP
4. Apply Guest if no match

Authentication Flow

```
Client.LogOn(username, password)

1. Check Engineering Users (SecurityUsers)
2. Check Runtime Users (Database)
3. Check LDAP Server (if configured)
4. First valid match logs in
```

Session Management

csharp

```csharp
// Get all active users
var users = @Security.GetActiveUsers();

// Check if RuntimeUser
bool isRuntimeUser = @Security.IsRuntimeUser(username);

// Get user source
string source = @Security.GetUserSource(username);
// Returns: "Internal", "Database", "AD"
```

Security Configuration

Forcing AD-Only Access

1. Enable **Use WA** in Runtime Startup
2. Disable standard authentication port
3. Configure AD-only port

csharp

```csharp
// Check if using Windows Authentication
if (@Client.IsWindowsAuthenticated)
{
    string domain = @Client.WindowsDomain;
    string user = @Client.WindowsUserName;
    @Info.Trace($"AD User: {domain}\\{user}");
}
```

Service Account Configuration

csharp

```csharp
// Configure service account for LDAP queries
@Security.LDAPServiceAccount = "svc_scada";
@Security.LDAPServicePassword = GetSecurePassword();
@Security.LDAPSearchBase = "OU=Users,DC=company,DC=local";
```

Best Practices Checklist

- **Use appropriate source** - AD for enterprise, DB for standalone
- **Set permissions carefully** - RuntimeUsers can't be admins
- **Use secure LDAP** - LDAPS on port 636 when possible
- **Implement audit trail** - Track user creation/modification
- **Regular cleanup** - Remove inactive RuntimeUsers
- **Secure database** - Protect RuntimeUsers table
- **Document user sources** - Clear origin tracking
- **Test authentication paths** - Verify all methods work

Common Configurations

Manufacturing Domain

```
AD Server: mfg.company.local
Port: 3102
Groups:
  - MFG\Operators  Operator
  - MFG\Engineers  Engineering
  - MFG\Managers  Supervisor
```

Enterprise LDAP

```
LDAP: ldaps://enterprise.local:636
Base DN: DC=enterprise,DC=local
Groups:
  - CN=SCADA_Users  User
  - CN=SCADA_Admin  Administrator
```

Troubleshooting

User not found:

- Check database connection
- Verify table exists
- Review AD connectivity
- Confirm username format

Cannot create user:

- Verify CreateUsers permission
- Check database write access
- Review policy restrictions
- Confirm unique username

AD/LDAP users not working:

- Test domain connectivity
- Verify group mappings
- Check authentication mode
- Review domain credentials

Wrong permissions:

- Check group name spelling (exact match required)
- Review permission mapping
- Test with known group

Slow authentication:

- Check domain controller load
- Review network latency
- Optimize LDAP queries
- Consider caching

See Also

- Security Module (Reference) - Complete security documentation
- Security & Compliance (Tutorial) - Setup guide
- Standards Compliance (How-to Guide) - Regulatory requirements

---

In this section...
