Security Documentation

Send comments on this topic.

Security Introduction

StrataFrame Role-Based Security

StrataFrame Role Based Security is very comprehensive and provides the end-user with the expected protection afforded by a well designed software security system. Its ultimate goal is to prevent a breach into unauthorized areas of an application from all points of a user interface. The actual implementation for the developer is straightforward and very well documented. This security system is more advanced than a typical out-of-the box framework security module.

Overview

Many application developers are faced with the arduous task of designing and implementing a security system that resides on top of an existing framework. StrataFrame Role Based Security provides a convenient, yet extremely thorough methodology of embedding security at the base level of your application.

Security within StrataFrame is engineered at the most elementary level of the framework. This allows the developer to set security rules with associated business objects and UI controls.

End-User Security Application Included

StrataFrame Role Based Security also provides the developer with a robust Security Application for development and maintenance of global security preferences, permissions, roles, and users. The Security Application may be embedded and distributed with your own application, with a royalty free runtime license.

Technical and User Help

Role Based Security is also distributed with both Technical Help to be used by the developer for implementation within your application, and with User Help that explains the Security Application for the end user. Of course, the User Help may be distributed royalty free runtime.

Security Documentation

Send comments on this topic.

Global Preferences Overview

Purpose: This document describes the use and maintenance of the global preferences of the security system.

Global Preferences Overview

The global preferences are used to enforce universal constraints within the security system. With the exception of password expiration and session lockouts, the values cannot be overridden. The preferences exist in order to control password strength, password policies, logon regulations, and session locking.

The global preferences are stored within their own table within SQL Server, but the table will only have one record per security project.

Security Documentation

Send comments on this topic.

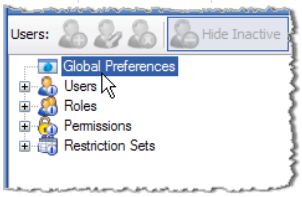
Editing Global Preferences

Purpose: This document describes the method used to edit the global preferences of the security system.

Editing Global Preferences

Follow these steps to edit the global preferences for the security system:

1. Ensure that the Security Dialog is open for the security project for which you want to edit the global preferences.
2. Select the Global Preferences node from the tree view.



3. Click the Edit button at the bottom of the right-hand property sheet.
4. Edit the properties within the global preferences.
5. Click Save to save your changes.

Security Documentation

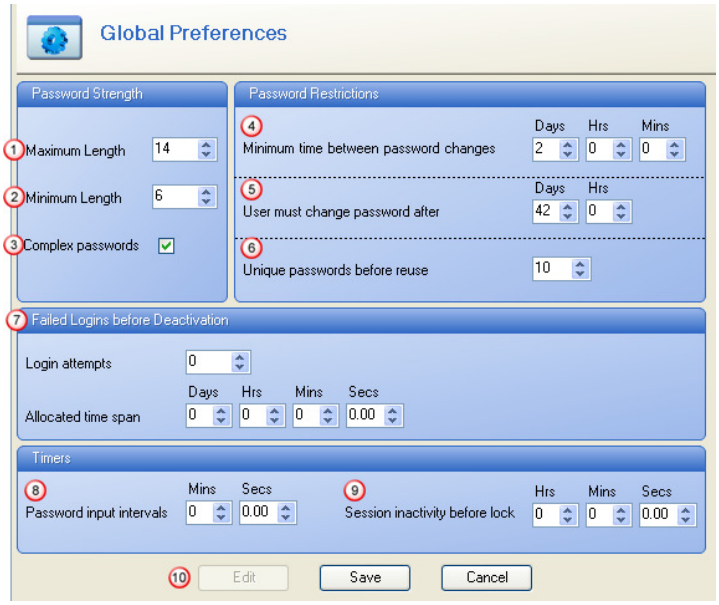
Send comments on this topic.

Global Preferences Properties

Purpose: This document describes the settings within the global preferences that can be edited through the global preferences editor.

Global Preferences Settings

The global preferences settings panel allows you to view/edit the global preferences for a security project.



Global Properties Settings Editor Fields

Password Strength & Restrictions

Typically, most users log on to their computer or application by using a combination of their name and password simply typed on a keyboard. Although alternative technologies exist for authentication, such as biometrics or smart cards, most organizations still rely on traditional passwords; therefore, it is important that password policies can be defined that mandate the use of strong passwords.

These settings exist to enforce password strength within the system.

1	Maximum Length	Specifies the maximum length (number of characters) that a password can be within the system. This value can be between 14 and 28 characters. The default is 14.
2	Minimum Length	Specifies the minimum length (number of characters) that a password can be within the system. This value can be between 0 and 14 characters. The default is 6. If this value is set to 0, users are allows to have blank passwords. The recommended value for this setting is 6-8 characters. **Note:** If Complex Passwords is enabled, and this value is set to less than 6 characters, then the minimum password length is enforced to be 6 characters.
3	Complex Passwords	Determines whether users are required to supply passwords that meet the minimum complexity requirements. Click here for a definition of what the system considers a complex password. If this box is checked, then the minimum password complexity requirements will be enforced any time a user's password is set or changed.
4	Minimum Time Between Password Changes	Determines the amount of time (in days, hours, and minutes) that a user must wait between password changes. For example, if the setting is 3 hours, then when a user changes his/her password he/she must wait 3 hours before his/her password can be changed again.
5	Maximum Password Age	Determines the amount of time (in days and hours) that a user can keep a password before it expires and must be changed. If this value is set to 0, then passwords will never expire and the system will never require users to change their passwords. **Note:** This value can be overriden at the user level by specifying that the user's password will never expire.
6	Enforce Password History	Prevents users from using passwords they have used in the past, up to the number of passwords that specified. This value can be between 0 and 24. The default is 10. If the password history is configured to 0, then the password history feature is disabled.

Logon and Sessions

7	Deactivate Account	Indicates how many invalid logon attempts the system will allow on the account before the account is deactivated. The number of invalid logon attempts specified must occur within the amount of time specified for the account to be disabled. The default number of invalid logon attempts is 3, and the default amount of time is 1 minute.
8	Password Input Intervals	Specifies the amount of time a user must wait after an invalid logon attempt before he/she is allows to logon again. This forces a user to wait the specified amount of time after they enter an invalid password. The default amount if 5 seconds.
9	Session Inactivity Lock	The amount of inactivity the application will allow before the session is locked and the user must re-authenticate before continuing to use the application. The system will resume to is previous state after the session is unlocked. The default for this value is 20 minutes. If this value is set to 0, the application will never timeout. **Note:** This value can be overriden at a user level.

Buttons

10	Edit, Save, and Cancel Buttons	• Edit - This button enables the global preferences for editing. • Save - Commits the changes to the global preferences. • Cancel - Aborts any changes made to the global preference record since the last time the Edit button was clicked.

Security Documentation

Send comments on this topic.

Password Complexity

Purpose: This document describes the requirements for a complex password within the security system.

Password Complexity

The requirements for a complex password follow closely to the requirements defined by Microsoft® for complex password within a Windows® 2003 system.

The exact specifications for a complex password are:

- The password must be at least six characters long.
- The password must contain characters from at least three of the following five categories:
  - English uppercase characters (A – Z)
  - English lowercase characters (a – z)
  - Base 10 digits (0 – 9)
  - Non-alphanumeric or symbols (for example: !,$,#, or %)
  - Unicode characters
- The password cannot contain three or more consecutive characters from a word in the user's account name. For example, if the account name is "John L. Doe", a password would not meet the minimum complexity requirements if any of the following combinations was contained within the password: "joh", "ohn", "doe".

Security Documentation

Send comments on this topic.

Permissions Overview

Purpose: This document provides an overview of permissions and how they are used within the security system.

Permissions Overview

Resources within the application are assigned a permission. This permission is required to access or manipulate that resource within the application. The permission is then granted or denied to specific roles and users to indicate whether the users have the right to access or manipulate that resource.

The permissions available depend on the application and are assigned by the software developer and cannot be created or edited outside of the development environment. Typically, a permission is binary: either you have a particular permission or you don't; however, any permission can be configured to also allow read-only access.

How Permissions are Used

Typically, permissions are divided into one of the following categories:

1. Form-Level Permission - Access to the form is either granted or denied.
2. Table-Level Permission
   - Table-Level Add - A user can add a record to a table.
   - Table-Level Edit - A user can edit a record within a table.
   - Table-Level Delete - A user can delete a record via the form.
3. Field-Level Permission - Access to a field can be specified as either editable, read-only, or denied.

Not all levels of permissions are required to exist for all forms, therefore, it is not necessary or even practical for a permission to exist for every type of event that exists within an application.

For example, for certain maintenance forms within the application, it is only necessary for a single permission to be assigned to deny access to the maintenance form itself; the more granular levels are not required.

On the other hand if a form contains sensitive information that not all users should be allowed to view or edit, the developer can force the security for that part of the application to be more detailed, assigning a separate permission to several different fields within a single table.

The permissions node is exposed at both design-time and run-time; however, at run-time, permissions cannot be modified in any way. Permission cannot be added, edited, or deleted.

Security Documentation

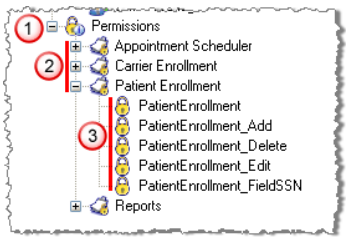
Send comments on this topic.

Viewing Permissions

Purpose: This document provides an overview of viewing permissions within the security system.

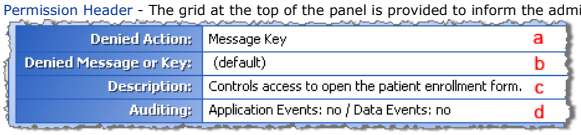
Viewing Permissions

From the left panel, you may either select the base permission node(1), a category node (2), or one of the individual permission nodes (3).



1. Base Permission Node - When the base permission node is selected all permissions within the application will be displayed in the panel on the right.

2. Category Nodes - When a base category node is selected all permission for a particular category will be displayed in panel on the right. Its list is identical to the list displayed when the base permission node is selected, the only difference being that the data is filtered to match the selected category.
3. Individual Permission Nodes - When selected, Roles and Users assigned to that particular permission will be displayed in the panel on the right.
   - Permission Header - The grid at the top of the panel is provided to inform the administrator of the permission's purpose as well as the role's auditing settings.

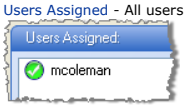


     a. Denied Action - Indicates the action for a denied permission. This is for reference and support and cannot be changed by the application user.
     b. Denied Message or Key - Indicates the actual message or message key (used for localization) which will be used to inform the user. This is for reference and support and cannot be changed by the application user.
     c. Description - Used to provide additional information to the application administrator when trying to identify the purpose of a particular permission.
     d. *Auditing -
        - *Application Events - When checked a log will be updated each time the permission is accessed. For example, if the patient enrollment permission has the application event selected, every time the form is accessed a log entry would be created.
        - *Data Events - This event works in conjunction with the actual form properties or business object settings. It is used to track data changes: creating records, editing records, or deleting records. It is controlled by the application developer and is placed on the header for reference.

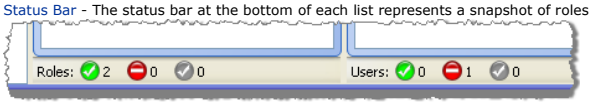
        ***Note:** The auditing functionality described above is planned but has not yet been implemented. This functionality will become available with a future release.

   - Roles Assigned - All roles that have been attached to the permission will be displayed.

   - Users Assigned - All users that have been assigned the permission will be displayed.



   - Status Bar - The status bar at the bottom of each list represents a snapshot of roles and users that have the following access: granted, blocked, or read only.



Security Documentation

Send comments on this topic.

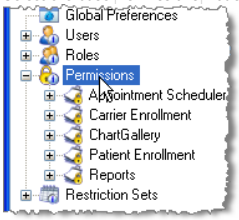
Adding a New Permission

Purpose: This document describes the process of adding a new permission to the security system.

Adding a New Permission

Follow these steps to add a new permission to the security system:

1. Select the base Permissions node.



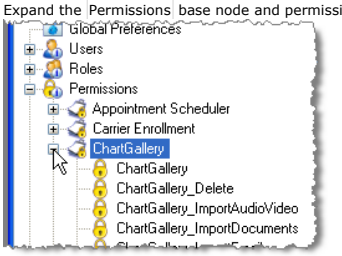
2. Click the Add a New Permission button on the Security Editor toolbar.
   OR
   Right-click the selected Permissions node within the left pane of the Security Editor and select New Permission... from the context menu.
   OR
   Right-click anywhere within the right pane and select New Permission... from the context menu.
3. Enter the information for the new permission within the Permission Editor.
4. Click Save to save the new permission and return to the Security Editor.

Purpose: This document describes the process of editing a permission within the security system.

Editing a Permission

Follow these steps to edit a permission within the security system:

1. Expand the Permissions base node and permission category to display the permission you want to edit.



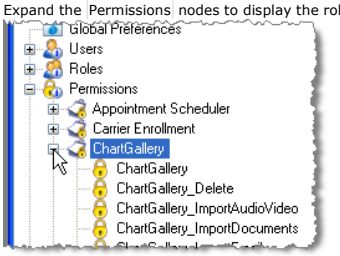
2. Right-click the node representing the permission to edit and select Edit Permission... from the context menu.
   OR
   Select the node representing the permission to edit and click the Edit Selected Permission button on the Security Editor toolbar.
3. Enter the information for the permission within the Permission Editor.
4. Click Save to save the permission and return to the Security Editor.

Purpose: This document describes the process of deleting a permission within the security system.

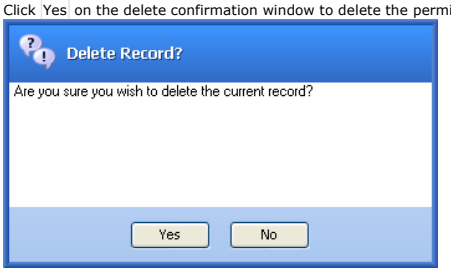
Deleting a Permission

To delete an existing permission:

1. Expand the Permissions nodes to display the role you want to delete.



2. Right-click the node representing the permission to delete and select Delete Permission... from the context menu.
   OR
   Select the node representing the permission to delete and click the Delete Selected Permission button on the Security Editor toolbar.
3. Click Yes on the delete confirmation window to delete the permission.

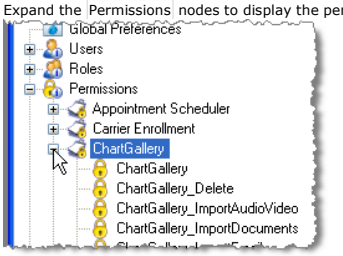


Purpose: This document describes the process of copying an existing permission within the security system.

Copying an Existing Permission

Follow these steps to copy an existing permission within the security system:

1. Expand the Permissions nodes to display the permission you want to copy.



2. Right-click the node representing the permission to copy and select Copy Permission... from the context menu.
3. Review and finalize the information for the permission within the Permission Editor.
4. Click Save to save the Permission and return to the Security Editor.

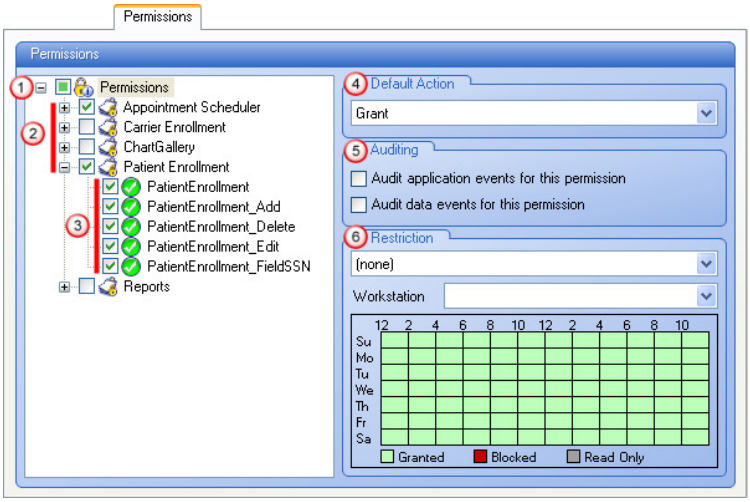
Note: Since the Key defaults to the exact Key name of the source permission, and no duplicate Keys may exist, the Key for the copied permission must be changed before a Save is allowed.

Purpose: This document provides an overview of assigning permissions to users and roles within the security system.

Assigning Permissions

Permissions are selected via the list row check box; a check mark indicates the permission is enabled. Permission selection and property management (such as actions, auditing, and restriction sets) can be cascaded from the parent node.

For example, if you want to select all permissions that comprise the "Patient Enrollment" category, simply select the Patient Enrollment category node and all of its children will be selected. This is a very fast way assign entire categories of permissions to a given user or role, or to apply properties to all permissions of a given category.



Permissions may be assigned or applied at three levels:

1. All Permissions - All available permissions will be selected for the role.
2. By Category – all available permissions for the category will be selected for the role.
3. By Permission – an individual permission is selected when checked.

Note: If a parent nodes is not selected yet one or more of its children are selected, the parent node will be marked with a square icon. On the preceding image notice the base permission node is showing a partial selection, while the selection for the Patient Enrollment node is complete since it includes all of its children.

Each assigned permission may have the following properties:

4. Default Action – This combo box allows up to three choices.
   a. Deny - This selection will deny access to the permission and the user will be informed typically with a warning message. This flag is used in conjunction with the "Denied Action" selection on the actual permission.
   b. Grant – This selection grants the user access to the permission.

c. Read-Only – This selection allows a user to view the permission but the data cannot be added, edited, or deleted.

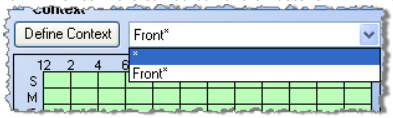
> **Note:** Read-only status will typically be disabled during permission creation for those permissions where it does not make sense. As such, if the Read-only option is not displayed in the combo box, it has been disabled at the permission level.

5. *Auditing - Selection of either auditing event overrides its corresponding lower level selection. For example, if the selection is made at the role level it will override any selection made at the permission level. If the selection is made at the user level it will override any selection made at the role or permission levels. Please refer to the "Permission Properties" help topic for further information.

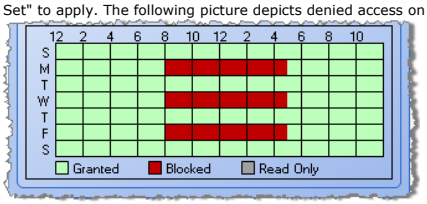
> ***Note:** The auditing functionality described above is planned but has not yet been implemented. This functionality will become available with a future release.

6. Restriction Set - A restriction set must first be created before it may be applied to a permission. Please refer to the "Restriction Sets Overview" and "Adding a New Restriction Set" help topics for further information.

a. Workstation - This combo box is used to select the workstation to be represented in the visual representation. If the restriction set has not been defined by workstation, only one row will exist in the combo box and an asterisk "*" will be displayed. Otherwise, the desired workstation may be chosen from the list in order to display the associated grid.



b. Visual Representation - This is a visual image, by workstation, for the selected restriction set. It is used to aid the administrator when trying to determine what actual "Restriction Set" to apply. The following picture depicts denied access on Monday, Wednesday, and Friday from 8:00am to 5:00pm.



Security Documentation

Send comments on this topic.

Permission Hierarchy

Purpose: To explain the permission hierarchy used when two or more permissions of the same level conflict.

Permission Hierarchy

A unique permission can be assigned to many roles and many roles can be assigned to a user. In addition, the same unique permission can be overridden at the user level. In other words, a single permission could be assigned to a user from multiple roles or at the user level.

When conflicting permissions are applied to a user, the following two rules are used to select the final effective permission:

- The assignment at the user level always takes precedence over any assignment at the role level.
- If a permission is assigned to multiple roles and subsequently those roles are assigned to a user, the permission assignment with the highest action takes precedence. The possible actions, from highest to lowest, are:
  1. Granted
  2. Granted with Restriction Set
  3. Read-only with Restriction Set
  4. Read-only
  5. Blocked with Restriction Set
  6. Blocked

> **Note:** If two permissions with restrictions sets of the same level (i.e. two permissions set as "Blocked with Restriction Set") are applied to a single user, the two different permission sets will be combined using the above hierarchy as a guide.

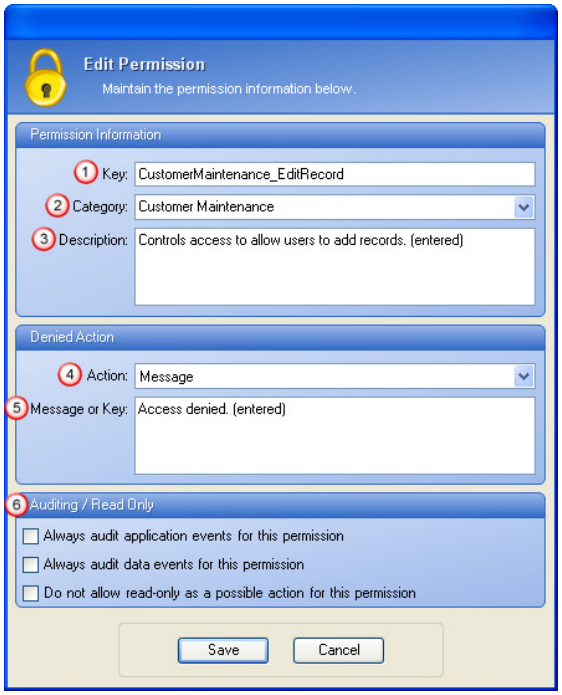
Security Documentation

Send comments on this topic.

Permission Properties

Purpose: This document describes the properties associated with a permission.

Permission Properties



Permission Properties

1	Key	Enter a unique identifier that best describes the permission.
2	Category	Enter a category for the permission or select one that has been previously entered. Typically, an application function that is easily recognized by an end-user is used for the text of a category. For example, financial reports, appointment scheduling, order entry, etc.
3	Description	This property is used to further describe the permission.
4	Action	Select the action to be invoked when this permission is denied. • No Message - This option is used when no message needs to be displayed to the end user. For example: 1. Programmatic Access - This option may be used for programmatic access when user intervention is not necessary. 2. Field Level - If this is set for a field and its access is denied, the user will not be able to view the contents of the field (it will be blank). The actual data that is bound to the field is not changed, only its viewing state. • Message - The message entered in the Message or Key text box will be displayed to the end-user. • Message Key - The message key entered in the Message or Key text box will be looked-up and its corresponding text will be displayed to the user. • Replace each Character - This selection is the same as the "No Message" selection, the only difference being that instead of blanks being displayed, the character "X" will be used to replace the field contents. Again, only the viewing state is changed with this selection, the actual data is unharmed.
5	Message or Key	Based on the selection made via the Action combo box, enter the text of the denied message or a localized key for display to the end-user.
6	Auditing* / Read Only	• Application Events - When checked a log will be updated each time the permission is accessed. For example, if a Logon permission has this checkbox selected, a log entry will be created every time the event is accessed during the logon process. • Data Events - This event works in conjunction with the actual form properties or business object settings. It is used to track data changes: creating records, editing records, or deleting records. Each CRUD function can be controlled via business properties settings providing developer control. **Note:** These event exists at the permission level for flexibility. They are typically assigned to a permission at the role or user level. It is important to remember that a user cannot change a permission. Hence, if you select auditing at the root permission it cannot be disabled via the runtime application. • Do Not Allow Read-only - This check box excludes the read-only choice from the action combo box at the role or user level. It simply does not make sense for some permissions to be allowed a read-only status. For example, form access should be either granted or denied, not read-only. For this reason, it is recommended all form permissions have this option checked.

* - Auditing functionality was not included with the 1.5 release of Strataframe. This functionality will become available with a future release.

Security Documentation

Send comments on this topic.

Roles Overview

Purpose: This document provides an overview of roles and how they are used within the security system.

Roles Overview

In role-based security, permissions are associated with roles. Users are then made members of roles, thereby acquiring the associated permissions. The purpose of the role is to group like tasks such as nurses, doctors, or insurance clerks together, thereby helping manage users and control access to application functions.

The assignment of a permission to a role is a simple binary function: permissions are checked to assign access and are unchecked to deny access. Furthermore, a role can be controlled at a more granular level with actions, auditing, and its restrictions of use.

Security Documentation

Send comments on this topic.

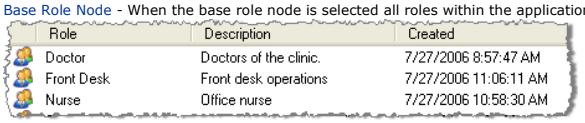
Viewing Roles

Purpose: This document provides an overview of viewing roles within the security system.

Viewing Roles

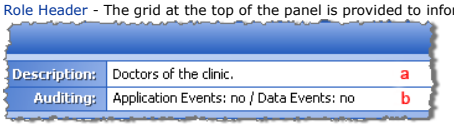
From the left panel, you may either select the base role node (1), or one of the individual role nodes (2).

1. Base Role Node - When the base role node is selected all roles within the application will be displayed in the panel on the right.



Role	Description	Created
Doctor	Doctors of the clinic.	7/27/2006 8:57:47 AM
Front Desk	Front desk operations	7/27/2006 11:06:11 AM
Nurse	Office nurse	7/27/2006 10:58:30 AM

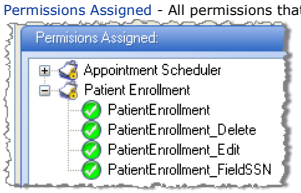
2. Individual Role Nodes - When an individual role node is selected both Permissions Assigned and Users Assigned to a particular role will be displayed in the panel to the right.
   - Role Header - The grid at the top of the panel is provided to inform the administrator of the role's purpose as well as the role's auditing settings.



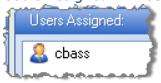
     a. Description - Used to identify the purpose and function of the role.
     b. *Auditing - Selection of either auditing event overrides its corresponding lower level selection. For example, if the selection is made at the role level it will override any selection made at the permission level. If the selection is made at the user level it will override any selection made at the role or permission levels. Please refer to the "Permission Properties" help topic for further information.
        - *Application Events - When checked a log will be updated each time the permission is accessed. For example, if the patient enrollment permission has the application event selected, every time the form is accessed a log entry would be created.
        - *Data Events - This event works in conjunction with the actual form properties or business object settings. It is used to track data changes: creating records, editing records, or deleting records. It is controlled by the application developer and is placed on the header for reference.

     Note: The auditing functionality described above is planned but has not yet been implemented. This functionality will become available with a future release.

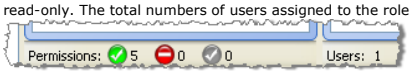
   - Permissions Assigned - All permissions that have been attached to the role will be displayed.



   - Users Assigned - All users that have been attached to the role will be displayed.



   - Status Bar - The status bar at the bottom of each list represents a snapshot of permissions and users. The permissions are summarized by the access status: granted, blocked, or read-only. The total numbers of users assigned to the role are displayed on the right panel.



Security Documentation

Send comments on this topic.

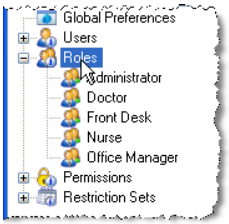
Adding a New Role

Purpose: This document describes the process of adding a new role to the security system.

Adding a New Role

Follow these steps to add a new role to the security system:

1. Select the base Roles node.



2. Click the Add a New Role button on the Security Editor toolbar.
   OR
   Right-click the selected Roles node within the left pane of the Security Editor and select New Role... from the context menu.
   OR
   Right-click anywhere within the right pane and select New Role... from the context menu.
3. Enter the information for the new role within the Role Editor.
4. Click Save to save the new role and return to the Security Editor.

Security Documentation

Send comments on this topic.

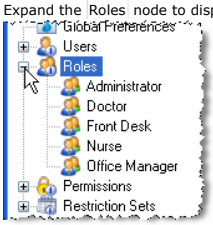
Editing a Role

Purpose: This document describes the process of editing a role within the security system.

Editing a Role

Follow these steps to edit a role within the security system:

1. Expand the Roles node to display the role you want to edit.



2. Right-click the node representing the role to edit and select Edit Role... from the context menu.
   OR
   Select the node representing the role to edit and click the Edit Selected Role button on the Security Editor toolbar.
3. Enter the information for the role within the Role Editor.
4. Click Save to save the role and return to the Security Editor.

Security Documentation

Send comments on this topic.

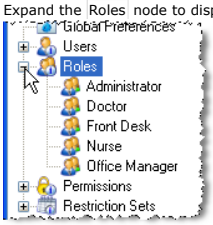
Deleting a Role

Purpose: This document describes the process of deleting a role within the security system.

Deleting a Role

To delete an existing role:

1. Expand the Roles node to display the role you want to delete.



2. Right-click the node representing the role to delete and select Delete Role... from the context menu.
   OR
   Select the node representing the role to delete and click the Delete Selected Role button on the Security Editor toolbar.
3. Click Yes on the delete confirmation window to delete the role.

Security Documentation

Send comments on this topic.

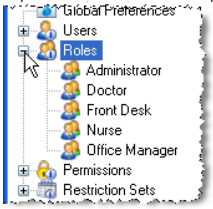
Copying a Role

Purpose: This document describes the process of copying an existing role within the security system.

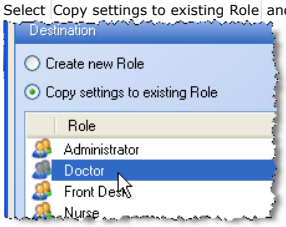
Copying an Existing Role

Follow these steps to copy an existing role within the security system:

1. Expand the Roles node to display the role you want to copy.



2. Right-click the node representing the role to copy and select Copy Role... from the context menu.
3. Select Create new Role from the Role Copy window.
   OR
   Select Copy settings to existing Role and select the target existing Role from the Role Copy window.



4. Click OK to accept the role copy and display the Role Editor.
5. Review and finalize the information for the role within the Role Editor.
6. Click Save to save the role and return to the Security Editor.

Note: When copying settings to an existing role, all auditing and permission settings for the existing role will be overwritten with the source role's settings. The role name and description will be left untouched.

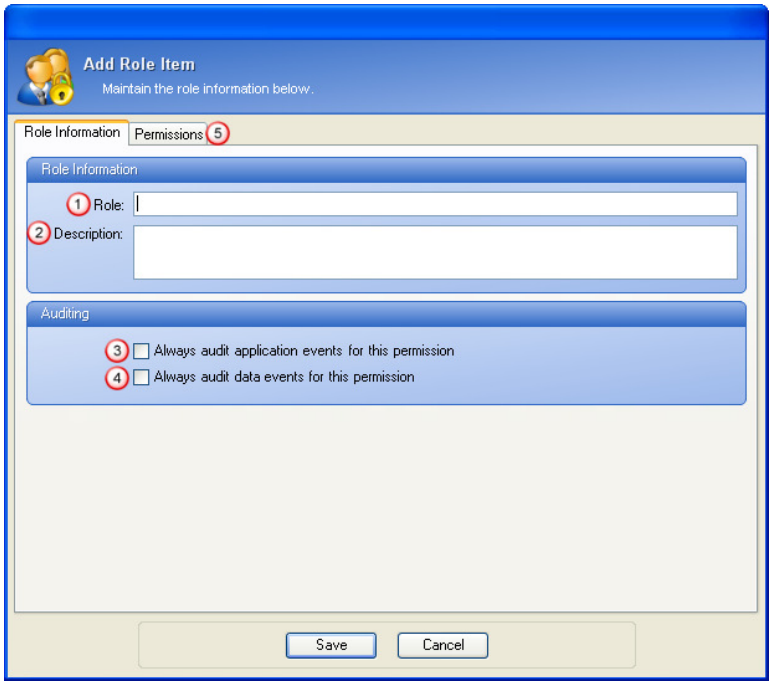
Security Documentation

Send comments on this topic.

Role Properties

Purpose: This document describes the properties associated with a role.

Role Properties



Role Properties

1	Role	The unique description of the role to be used as the primary identifier of the role. It is recommended that the role name describe typical employee functions (doctor, nurse, etc.)
2	Description	Can be used to further describe the role. The Description is typically used by an administrator for remarks or comments.
3	Audit Application Events	If selected, all application events regardless of function will be audited for this role. Application events are defined by the application's functionality (i.e. logon/logoff.)
4	Audit Data Events	A data event exists at the actual business object or table level. A business object containing patient demographics information or transactions could be an auditable event. Hence, if selected all data processing for the business object such as adding, editing, or deleting records will be captured. The records that are edited will contain the field's original value and its new value for comparison.
5	Permissions	Please refer to the Assigning Permissions help topic for further information.

Security Documentation

Send comments on this topic.

Users Overview

Purpose: This document provides an overview of users and how they are used within the security system.

Users Overview

The purpose of any security system is to control user access. A well designed system aids the application administrator when actually configuring a user's access rights. The objective of a role-based security system is to manage the predominance of the access rights at the actual role level and only override permission at the user level when necessary.

This technique allows changes to be made on a global level, the actual role, and ultimately inherited at the individual user level. Nevertheless, it is important to be able to granularly control any user without effecting roles or other users. This system is designed to incorporate the necessary control and access for the end-user.

Security Documentation

Send comments on this topic.

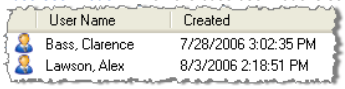
Viewing Users

Purpose: This document provides an overview of viewing users within the security system.

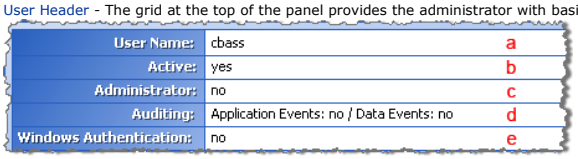
Viewing Users

From the left panel, you may either select the base user node (1), or one of the individual user nodes (2).

1. Base User Node - When the base user node is selected all roles within the application will be displayed in the panel on the right.



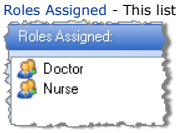
2. Individual User Nodes - When an individual user node is selected both Permissions Assigned and Roles Assigned to a particular user will be displayed in the panel to the right.
   - User Header - The grid at the top of the panel provides the administrator with basic reference information for a particular user.



     a. User Name - Name used to identify the user during logon.
     b. Active - Indicate if the user account is active for use.
     c. Administrator - Indicates if the user has administrator rights.
     d. *Auditing - Selection of either auditing event overrides its corresponding lower level selection. For example, if the selection is made at the role level it will override any selection made at the permission level. If the selection is made at the user level it will override any selection made at the role or permission levels. Please refer to the "Permission Properties" help topic for further information.
        - *Application Events - When checked a log will be updated each time the permission is accessed. For example, if the patient enrollment permission has the application event selected, every time the form is accessed a log entry would be created.
        - *Data Events - This event works in conjunction with the actual form properties or business object settings. It is used to track data changes: creating records, editing records, or deleting records. It is controlled by the application developer and is placed on the header for reference.

          ***Note:** The auditing functionality described above is planned but has not yet been implemented. This functionality will become available with a future release.

     e. Windows Authentication - This selection indicates that windows authentication will be used during the logon process.
   - Roles Assigned - This list provides an accounting of all roles assigned to the user.



   - Permissions Assigned - This treeview contains three nodes: Inherited, Overridden, and Combined.
     a. Inherited - This node lists permissions that are assigned to the user because they compose a role. For example, if the permission "Patient Enrollment" is assigned to the role "Front Desk" and this role is assigned to this user, then "Patient Enrollment" will be listed under this node.
     b. Overridden - This node lists permission that have been explicitly assigned or overridden at the user level.
     c. Combined - This node is a combination of the previous two nodes. Its contents are used to create the permissions collection that is used during the logon process. If two or more inherited permissions are conflicting, the final effective permission shown in the Combined node is determined using the Permission Hierarchy.
   - Status Bar - The status bar at the bottom of each list represents a snapshot of roles and permissions. The total numbers of roles assigned to the user are displayed on the left panel. The permissions are summarized by the access status: granted, blocked, or read-only.

Security Documentation

Send comments on this topic.

Adding a New User

Purpose: This document describes the process of adding a new user to the security system.

Adding a New User

Follow these steps to add a new User to the security system:

1. Select the base Users node.

2. Click the Add a New User button on the Security Editor toolbar.
   OR
   Right-click the selected Users node within the left pane of the Security Editor and select New User... from the context menu.
   OR
   Right-click anywhere within the right pane and select New User... from the context menu.
3. Enter the information for the new user within the User Editor.
4. Click Save to save the new user and return to the Security Editor.

Purpose: This document describes the process of editing a user within the security system.

Editing a User

Follow these steps to edit a user within the security system:

1. Expand the Users node to display the role you want to edit.

2. Right-click the node representing the user to edit and select Edit User... from the context menu.
   OR
   Select the node representing the user to edit and click the Edit Selected User button on the Security Editor toolbar.
3. Enter the information for the user within the User Editor.
4. Click Save to save the user and return to the Security Editor.

Purpose: This document describes the process of deleting a user within the security system.

Deleting a User

To delete an existing user:

1. Expand the Users node to display the role you want to delete.

2. Right-click the node representing the user to delete and select Delete User... from the context menu.
   OR
   Select the node representing the user to delete and click the Delete Selected User button on the Security Editor toolbar.
3. Click Yes on the delete confirmation window to delete the user.

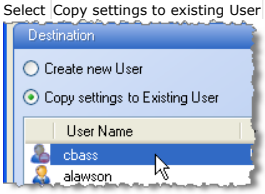
Purpose: This document describes the process of copying an existing user within the security system.

Copying an Existing User

Follow these steps to copy an existing user within the security system:

1. Expand the Users node to display the user you want to copy.

2. Right-click the node representing the user to copy and select Copy User... from the context menu.
3. Select Create new User from the User Copy window.
   OR
   Select Copy settings to existing User and select the target existing User from the User Copy window.



4. Click OK to accept the user copy and display the User Editor.
5. Review and finalize the information for the user within the User Editor.
6. Click Save to save the user and return to the Security Editor.

Note: When copying settings to an existing user, all options, including role and permission settings, for the existing user will be overwritten with the source user's settings. The user name and logon information will be left untouched.

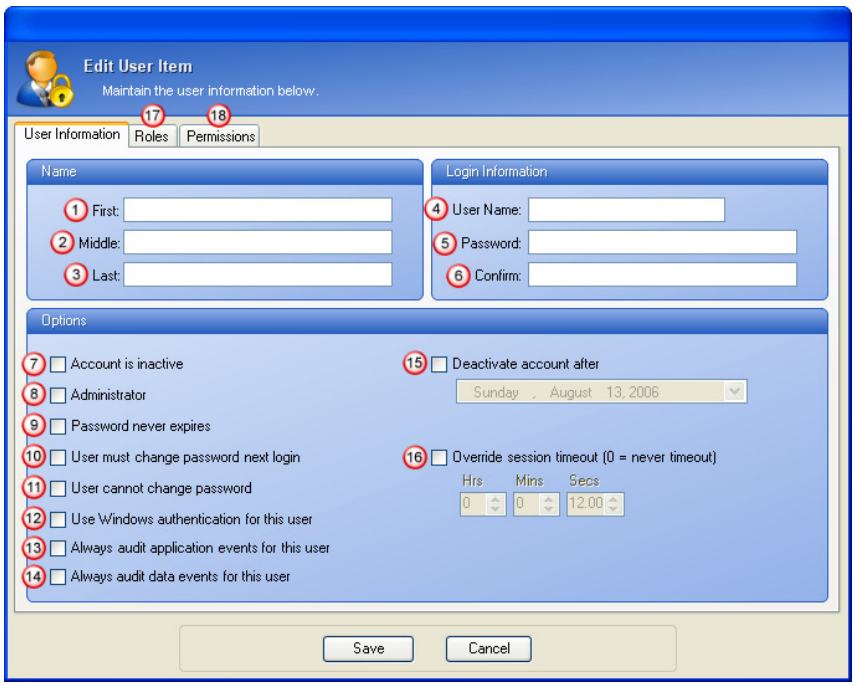
Security Documentation

Send comments on this topic.

User Properties

Purpose: This document describes the properties associated with a user.

User Properties



Name		
1	First	Enter the first name of the user.
2	Middle	If desired, enter the middle name or initial.
3	Last	Enter last name of the user.
Logon Information		
4	Username	Enter unique username that is to be used during the logon process. The username is not case-sensitive and must be at least 3 characters long.
	Password	Enter the unique, case-sensitive password. If the global preference Complex Passwords is selected then complex password rules will be enforced.

5		
6	Confirm	Re-enter the password in order to ensure it was entered correctly.
Options		
7	Account is inactive	Marks user account as inactive denying all access. Additionally, inactive accounts can be filtered from the user list on the Security Editor.
8	Administrator	Marks the user account as an administrator granting a higher level of application access, not requiring permission management. An administrator is also granted access to all aspects of the "Security Editor" for maintenance. Any user marked as administrator will have full access, regardless of any specific roles or permissions applied. **Tip:** Administrator users are identified by a police officer user icon in security maintenance.
9	Password never expires	This selection overrides any global password refreshment rules and grants the password an eternal existence.
10	Change password next logon	The user will be required to create a new password during the next logon process.
11	User cannot change password	This selection precludes the user from changing their password.
12	Windows authentication	If selected, the entered password on this form will be circumvented during the logon process. The user will instead be authenticated via the windows operating system. This type of authentication is convenient for the administrator during the setup process since a single password is used for the machine and the application but is less secure when using session locks within the application.
13	Audit application events*	This will override any previous role or permission settings and will force all application events to be audited for this user.
14	Audit data events*	This will override any previous role or permission settings and will force all data events to be audited for this user.
15	Deactivate account	This option will deactivate a user account after the given date has expired. The calendar control will be enabled and an actual date must be selected to determine the date of deactivation. This feature is typically used for temporary employees when a their access to the application should cease after a given date.
16	Override timeout session	Overrides the global preference for session timeouts. If selected the time box to the right will be enabled and the new overridden timeout must be entered.
Roles and Permissions		
17	Roles	Select the row check box to assign a role. All of the role's permissions will be attached to the user per the permission hierarchy scheme.
18	Permissions	Please refer to the Assigning Permissions help topic for further information.

* - Auditing functionality is planned but has not yet been implmented. This functionality will become available with a future release.

Security Documentation

Send comments on this topic.

Restriction Sets Overview

Purpose: This document provides an overview of restriction sets and how they are used within the security system.

Restriction Set Overview

A Restriction Set is used to create a global user access rule that is comprised of days-of-week, time-of-day, workstation, and action. A Restriction Set is used to enable or inhibit a user's access depending on the action (grant, deny, or read-only).

Ultimately, a restriction set is enforced at the role or user level during permission application. Any restriction set that is assigned to a role will be inherited by its corresponding users. Conversely, additional granularity is provided at the user level if the restrictions imposed at the role level need to be overridden.

Security Documentation

Send comments on this topic.

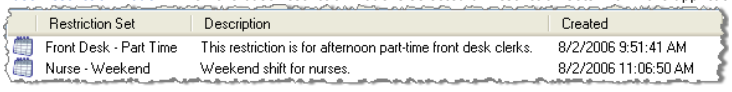
Viewing Restriction Sets

Purpose: This document provides an overview of viewing restriction sets within the security system.

Viewing Restriction Sets

From the left panel, you may either select the base restriction node (1), or one of the individual restriction nodes (2).

1. Base Restriction Node - When the base restriction node is selected all restriction sets within the application will be displayed in the panel on the right.

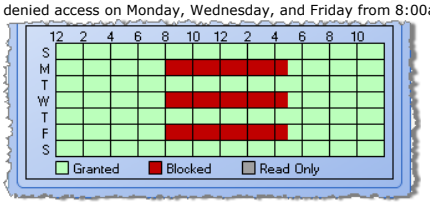


Restriction Set	Description	Created
Front Desk - Part Time	This restriction is for afternoon part-time front desk clerks.	8/2/2006 9:51:41 AM
Nurse - Weekend	Weekend shift for nurses.	8/2/2006 11:06:50 AM

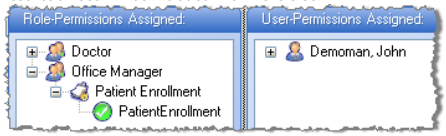
2. Individual Restriction Nodes - When an individual restriction node is selected both Role-Permissions Assigned and User-Permissions Assigned to that particular restriction set will be displayed in the panel to the right.

- Workstation Combo Box - This combo box is used to select the workstation to represent via the visual grid. Initially the default workstation * will be selected. Drop down the box in order to select a workstation.

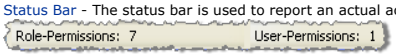
- Visual Representation - The grid is used to get a quick reference by workstation the day-of-wee, time-of-day, and action status. The following picture depicts that the restriction is denied access on Monday, Wednesday, and Friday from 8:00am to 5:00pm.



- Assigned Lists - The following two lists display all roles and users that have been assigned to the particular restriction set. This makes it quick and easy to determine to what roles or individual users the current restriction set has been applied. If a node is expanded, i.e. patient enrollment, each independent permission within the category that is attached to the restriction set will be included within the tree.



- Status Bar - The status bar is used to report an actual accounting of permissions that have been assigned the Restriction Set at both the Role and User Level.



Security Documentation

Send comments on this topic.

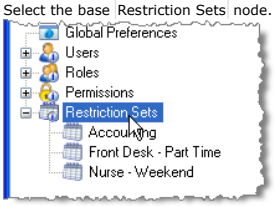
Adding a New Restriction Set

Purpose: This document describes the process of adding a new restriction set to the security system.

Adding a New Restriction Set

Follow these steps to add a new restriction set to the security system:

1. Select the base Restriction Sets node.



2. Click the Add a New Restriction button on the Security Editor toolbar.
   OR
   Right-click the selected Restriction node within the left pane of the Security Editor and select New Restriction... from the context menu.
   OR
   Right-click anywhere within the right pane and select New Restriction... from the context menu.
3. Enter the information for the new restriction within the Restriction Set Editor.
4. Click Save to save the new restriction and return to the Security Editor.

Security Documentation

Send comments on this topic.

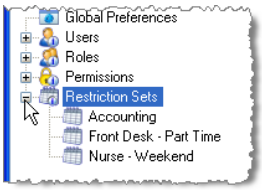
Editing a Restriction Set

Purpose: This document describes the process of editing a restriction set within the security system.

Editing a Restriction Set

Follow these steps to edit a restriction set within the security system:

1. Expand the Restriction Sets node to display the restriction you want to edit.



2. Right-click the node representing the restriction to edit and select Edit Restriction... from the context menu.
   OR
   Select the node representing the restriction to edit and click the Edit Selected Restriction button on the Security Editor toolbar.
3. Enter the information for the role within the Restriction Set Editor.
4. Click Save to save the restriction and return to the Security Editor.

Security Documentation

Send comments on this topic.

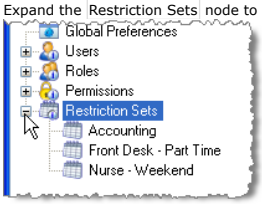
Deleting a Restriction Set

Purpose: This document describes the process of deleting a restriction set within the security system.

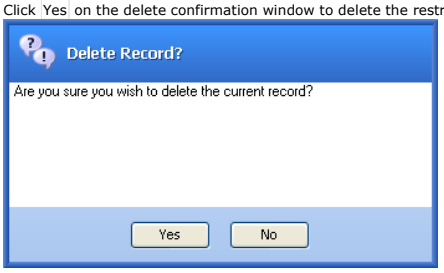
Deleting a Restriction Set

To delete an existing restriction set:

1. Expand the Restriction Sets node to display the restriction you want to delete.



2. Right-click the node representing the restriction to delete and select Delete Restriction... from the context menu.
   OR
   Select the node representing the restriction to delete and click the Delete Selected Restriction button on the Security Editor toolbar.
3. Click Yes on the delete confirmation window to delete the restriction.



Security Documentation

Send comments on this topic.

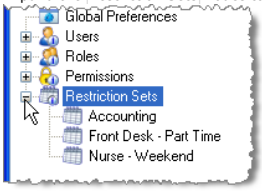
Copying a Restriction Set

Purpose: This document describes the process of copying an existing restriction set within the security system.

Copying an Existing Restriction Set

Follow these steps to copy an existing restriction set within the security system:

1. Expand the Restriction Sets nodes to display the restriction you want to copy.



2. Right-click the node representing the restriction to copy and select Copy Restriction... from the context menu.
3. Review and finalize the information for the restriction within the Restriction Set Editor.

4. Click Save to save the restriction and return to the Security Editor.

Note: Since the Name defaults to the exact Name of the source permission, and no duplicate Names may exist, the Name for the copied permission must be changed before a Save is allowed.

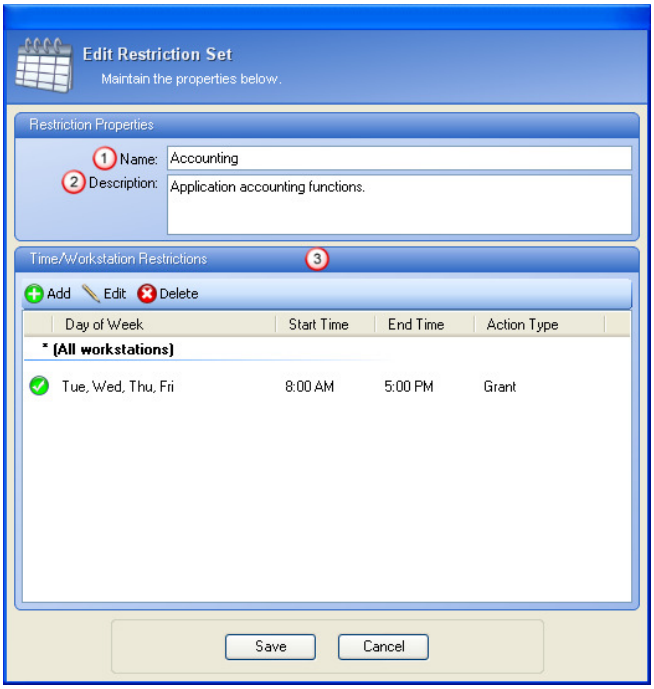
Security Documentation

Send comments on this topic.

Restriction Set Properties

Purpose: This document describes the properties associated with a restriction set.

Restriction Set Properties

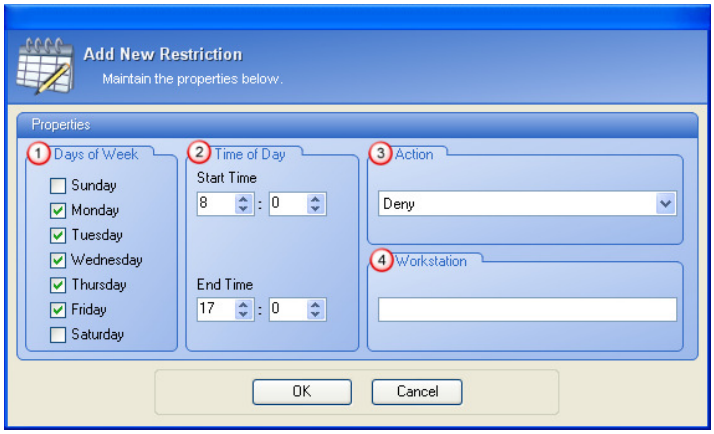


Restriction Properties

1	Name	Enter a unique name identifier that best describes the function of the restriction set.
2	Description	This property is used to further describe a restriction set.
3	Time/Workstation Restrictions	This list is the portal used to a maintain each restriction. The maintenance functions are invoked via the toolstrip buttons, add, edit, or delete. For reference the list is grouped by workstation. • Add - Displays Restriction Entry dialog window permitting a new record to be created. • Edit - Highlight the desired context item, in the list, and its contents will be displayed via the Restriction Entry dialog for editing. • Delete - Highlight the desired context item, in the list, for deletion. A warning message will be displayed requiring confirmation before the row will be deleted.

Restriction Entry

This dialog is called via an Add or Edit from the Time/Workstation Restrictions section of the Edit Restriction Set window. It is organized into four categories: days-of-week, time, action, and workstation.



Restriction Entry

1	Days of Week	Select the days of week for the context.
2	Time	Select the time range the context is to be enforced. The time range cannot overlap with an existing context's time range.
3	Action	Select the action for this context: deny, grant, or read-only. The action will override the default action for the permission based on its contexts: time or workstation.
4	Workstation	Enter the name of the workstation on which the context is to be enforced, wildcards are permitted. If the network is configured where all task specific machines have a common naming convention a wildcard can be used. The normal windows wildcard characters "?" and "*" apply. For example, a group of workstations named FrontDesk1, FrontDesk2, and FrontDesk3 could be handled by entering "Front?" or "Front*" into the Workstation field.

Security Documentation

Send comments on this topic.

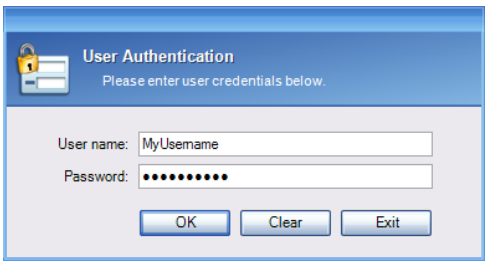
Logging into the Application

Purpose: This document discusses how to log into a security-enabled application.

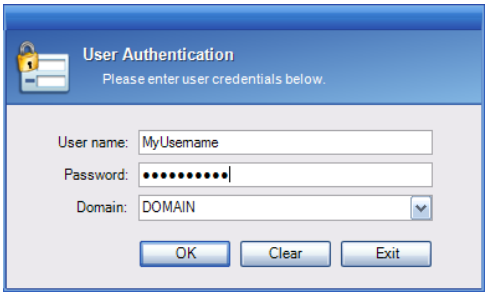
Logging In

A security-enabled StrataFrame application will generally require a login to be entered before the application can be accessed. The login form requires a username and password and can also require a domain name if Windows authentication is enabled for the application. The following are two examples of typical login forms:

- A typical StrataFrame Application Login form.



- A typical StrataFrame application login form with Windows authentication enabled.



Purpose: This document discusses the error messages that you could receive when trying to log in.

Failed Logins

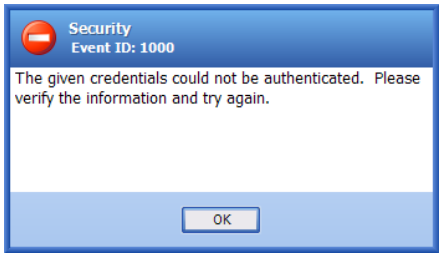
There are several reasons why a user might not be able to login to the application:

- Invalid user name/password/domain combination
- Deactivated account
- Denied login permission

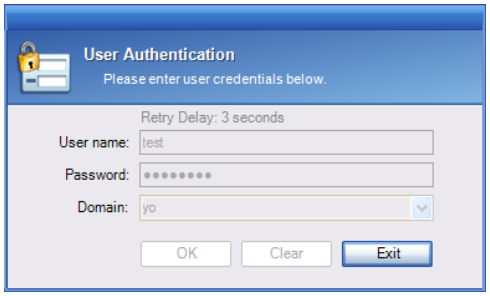
Invalid UserName/Password/Domain Combination

The following notifications and effects can occur as a result of an invalid entry of the Username, Password, or Domain:

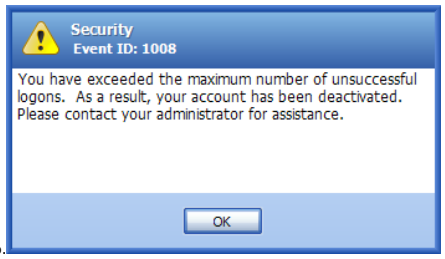
- Invalid Credentials - When an invalid user name/password/domain combination is supplied, the user will receive a message similar to the one shown below indicating that the credentials they provided could not be authenticated.



- Login Form Retry Delay - In addition to the above message, the login form will also be disabled for a period of time specified by the password input interval in the global preferences. A label on the login form will display the amount of time that the user must wait before they will be allowed to attempt another login. This security measure is in place to help prevent "brute force" password attacks where the attacker tries to guess the password by repeatedly entering passwords until the guess one that allows him/her to log into the application.



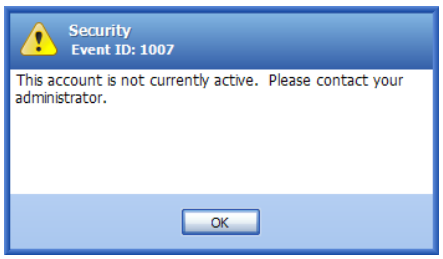
- Maximum Logins - When a password for a username has been entered improperly too many times (specified by the max invalid login attempts global preference), the account will be disabled, and the user will receive a message indicating so.



Deactivated Account

The following message will occur as a result of a deactivated account:

- Deactivated Account - When an account has been deactivated, the account is no longer permitted access to login to the application. If a login for the deactivated account is attempted, the user will be shown a message indicating that they cannot login to the application as a result of account deactivation.

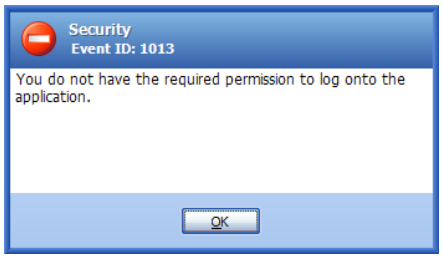


Denied Login Permission

The following message will occur as a result of denied login permissions:

- Denied Login Permission - The permission required to login to the application can be set through the MicroFour.StrataFrame.Security.Login.LoginSecurityKey property. If this permission

is denied to a user (generally through a restriction set), then user will be notified that he/she is not allowed to login to the application at this time and workstation.



Security Documentation

Send comments on this topic.

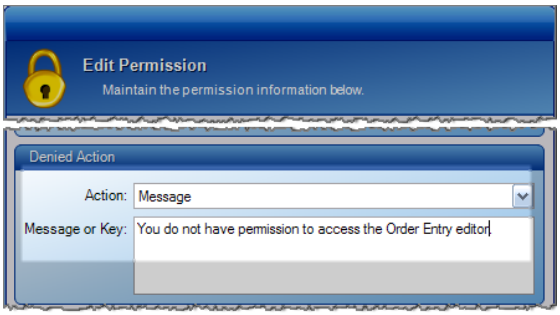
Security Keys on Forms and Business Objects

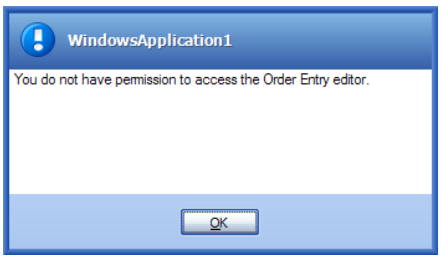
Purpose: This document provides a description of how security keys assigned to forms and business objects affect the function of the objects at runtime. For information on how to attach security keys to objects, visit Assigning Security Keys.

Security Keys on Forms and Business Objects

When a security key is attached to a form or a business object, the framework verifies that the current user has the requested permission before permitting access to that permission. Permissions that attach to business objects and fields should not allow read-only access (read-only access can be turned off within the Permission Editor for each permission).

The framework will show an access denied message, specified within the permission itself, when the permission is denied to a requested form or business object action. The framework will not disable or hide menu options that lead to these options, but the menu options can be disabled programmatically by testing the permissions on the CurrentUser object.





Security Documentation

Send comments on this topic.

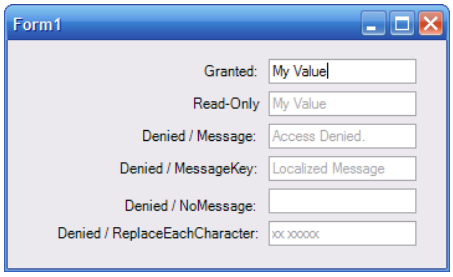
Security Keys on Business Object Fields

Purpose: This document provides a description of how security keys assigned to business object fields affect bound controls at runtime. For information on how to attach security keys to business object fields, visit Assigning Security Keys.

Security Keys on Business Object Fields

Security keys that are assigned to business object fields can be configured to allow read-only as a possible permission action for the field as bound controls can easily be made read-only. Like allowing read-only selecting "Replace Each Character" is only a valid choice for field-level security keys since the characters can only be replaced when the data is bound to a field.

The following screen shot displays examples of how each permission action and/or denied action affects controls bound to a business object.



Security Documentation

Send comments on this topic.

Session Locking

Purpose: The purpose of this document is to discuss the use of session locking and how it affects the application when the current user is changed when the session locks.

Session Locking

Session locking allows the application to be "locked" to require the user to re-authenticate to continue use of the application. Session locking also allows the logged in user to be quickly switched, without closing and restarting the application. When a user re-authenticates, his/her permissions are re-evaluated from the database, causing objects to which the user does not have access to be hidden or restricted, this includes forms, fields, etc.

Locked Forms

When an application is locked and a different user than the previous one logs back into the system, the new user might not have permission to view all of the forms that are currently open within the application. For each form that the new user does not have permission to view, a "form lock panel" is superimposed over the form to prevent the new user from having access to the already opened forms.

The locked forms and the entire application are prevented from being closed so that information is not lost if the previous user had unsaved changes on a locked. A user with sufficient privileges to view the form must first log into the system to close the forms before the application can be closed.

Security Documentation

Send comments on this topic.

Adding Security to an Application

Purpose: The purpose of this document is to provide step-by-step instructions on how to implement StrataFrame's role-based security within an application.

Adding Security to an Application

To add security to an application:

1. Create your application from the StrataFrame Windows Application w/ Security template – If you do not create your application from the Windows application template that includes security, you will need to complete the following steps manually:
   a. Add a reference to MicroFour StrataFrame Security.dll – The MicroFour StrataFrame Security.dll contains the role-based security implementation and must be referenced by your application.
   b. Show a Login Form to set the CurrentUser – You can either show the default login form, or create a login form from the Login Form Template. This login form should be show from within the ShowLoginAndInitMainForm() method within the AppMain.vb (Program.cs) file. For more information, refer to Showing the Initial Login Form.
   c. Initialize Session Locking - (optional) – The code required to initialize session locking should be added to the InitApplication() method within the AppMain.vb (Program.cs) file. For more information, refer to Initializing Session Locking.
   d. Specify the Security Key for encrypted user data – User data stored within the database is encrypted using 3DES encryption. A key must be specified to seed the encryption algorithm. This code should be added to the InitApplication() method within the AppMain.vb (Program.cs) file. For more information, refer to Specifying the Encryption Key for User Data.
   e. Specify the SecurityDataSourceKey - (optional) – If the security tables (SFS* tables) are not located within the database referenced by the default DataSource, then you must add a DbDataSourceItem with access to the database containing the tables, and specify the SecurityDataSourceKey to inform the role-based security module where the security tables

are located. For more information, refer to Setting the SecurityDataSourceKey.

f. Retrieve global preferences from the database – The global preferences should be retrieved from the database and stored within the SecurityBasics class properties. The code to accomplish this should be placed within the InitApplication() method within the AppMain.vb (Program.cs) file. For more information, refer to Retrieving Global Preferences from the Database.

g. Specify default values – There are additional properties on the SecurityBasics class that contain default values for your application and should be configured within the InitApplication() method of the AppMain.vb (Program.cs) file. These default values specify everything from the usernames and passwords for built in accounts to the default denied action and denied message.

2. Add a custom login form - (optional) – Adding a custom login form to your application is not required, but is recommended because the base login form does not display an application logo or company logo of any kind. For more information, refer to Creating a Custom Login Form.
3. Create permissions for the application within the Role-Based Security Editor – The permissions for an application must be created at design time through the Role-Based Security Editor. Once the permissions are created, they can them be attached to objects within the application to permission required to access that object. For more information, refer to the Adding a New Permission and Assigning Permissions topics.
4. Create Roles and Users for the application within the Role-Based Security Editor - (optional) – You can optionally create pre-defined roles and users that can be deployed with your application. These roles and users must be defined within the Role-Based Security Editor for your application. For more information, refer to the Adding a New Role and Adding a New User topics.
5. Assign security keys to objects within your application – Once the security keys have been defined through the Role-Based security editor, they must be attached to objects within your application to define the permission required to access that object. You can assign permissions to Business Object Fields, Business Object Actions, and Forms.
6. Programmatically test permissions within your application - (optional) – You can enable/disable or show/hide objects within your application by programmatically testing the CurrentUser's permissions and adjusting object properties appropriately.
7. Add the ability to show the SecurityDialog within your application – The SecurityDialog is used within your application to provide a maintenance form to your end-users that allows them to maintain users and roles. For more information, refer to Calling the Security Maintenance Dialog.
8. Deploy the security data with your database – The SFS* tables within the StrataFrame database are required for the StrataFrame security module to operate at runtime. The SFS* tables must exist in a location that is reachable by your application through one of your DbDataSourceItems. The Database Deployment Toolkit has the ability to deploy both the SFS* tables and the data for the tables containing the users, roles, and permissions you specify. Without the DDT, you must manually add the SFS* tables to your database script for deployment and devise a method to deploy the contents of the SFS* tables in the StrataFrame (design-time) database to your application's database at runtime. You only need to gather the records that match your project, identified by the appropriate record within the SFSProjects table. For more information, refer to Deploying Security Data.

Purpose: This document provides information on using the Security Key Type Editor to search for security keys and retrieve them from the database at design-time.

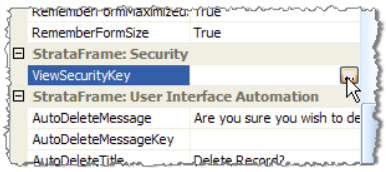
Invoking the Security Key Type Editor

The Security Key Type Editor is used to set the `*SecurityKey` properties on business objects, forms, and fields on business objects through the Business Object Mapper.

The exact properties that support this type editor are:

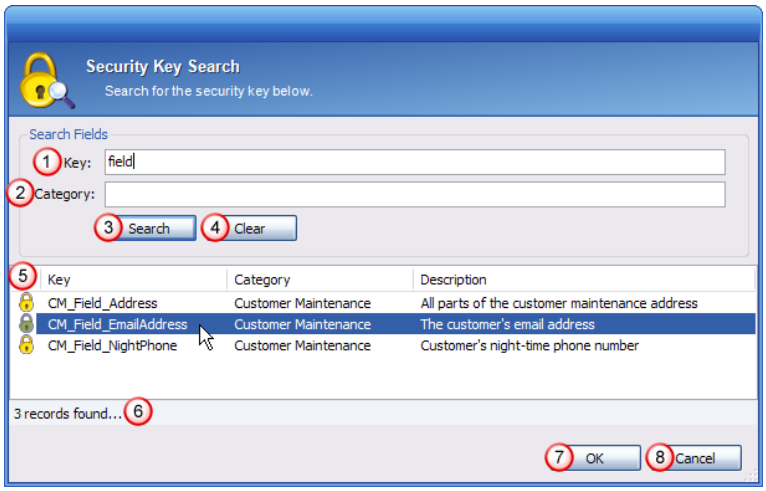
- Business Object Properties:
  - AddSecurityKey
  - EditSecurityKey
  - DeleteSecurityKey
  - QuerySecurityKey
- BaseForm Properties:
  - ViewSecurityKey

In order to invoke the type editor on one of these properties, click the ... button within the box for the property value.



Using the Security Key Type Editor

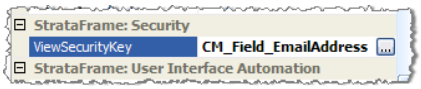
The Seccurity Key type editor is used to search for a specific security key (permission key) that has been created through the Security Maintenance Editor and assign it to a specific property.



File Group Properties Dialog Fields

1	Key	You can search by either the security key name or the category name or both.
2	Category	You can search by either the security key name or the category name or both.
3	Search	Click `Search` or press the enter key to execute the search.
4	Clear	The `Clear` button clears the search results and the `Key:` and `Category:` fields.
5	Results List	The results list contains the results of the search. It displays the Security Key, Category, and Description of the permission.
6	Status Bar	The status bar lists the number of results found by the search.
7	OK	Clicking `OK` closes the form and enters the selected security key in the property being edited.
8	Cancel	Clicking `Cancel` closes the form and does not modify the value of the property being edited.

Once the security key is located, double-click the record within the results list or select the record within the results list and click OK.



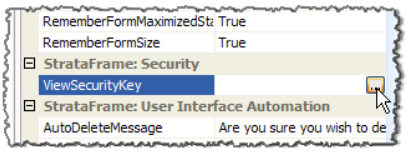
Purpose: The purpose of this document is to describe the process of assigning security keys to forms and describing the results of different permission actions when a security key is assigned to a form.

ViewSecurityKey Property

The `ViewSecurityKey` property is used to define the security key that is required to open a form.

Assigning the Key

Assigning the ViewSecurityKey property can be done through the property sheet of the form designer or programmatically. The Security Key Type Editor (link) is used to search for the appropriate security key to assign to the ViewSecurityKey property.



Sample – Setting the ViewSecurityKey Property [Visual Basic]

```
Imports MicroFour.StrataFrame.UI.Windows.Forms
...
Public Sub TestSecurity()
    '-Create a new form, set the security key and show it
    Dim loForm As New StandardForm()
    loForm.ViewSecurityKey = "MySecurityKey"
    loForm.Show()
End Sub
```

Sample – Setting the ViewSecurityKey Property [C#]

```
using MicroFour.StrataFrame.UI.Windows.Forms;
```

```
...
public void TestSecurity()
{
    //-- Create a new form, set the security key and show it
    StandardForm loForm = new StandardForm();
    loForm.ViewSecurityKey = "MySecurityKey";
    loForm.Show();
}
```

How the ViewSecurityKey Affects a Form at Run-time

Action	Result
ViewSecurityKey left blank	Security is not checked on the form, and the form is always shown.
ViewSecurityKey set, permission granted	The user has permission to view the form, and the form is shown.
ViewSecurityKey set, permission denied, denied action set to NoMessage	The form is not shown to the end-user and a MessageForm is shown to the end user containing the DefaultBlockedMessage.
ViewSecurityKey set, permission denied, denied action set to Message	The form is not shown to the end-user and a MessageForm is show to the end-user containing the message assigned within the SecurityDialog.
ViewSecurityKey set, permission denied, denied action set to MessageKey	The form is not shown to the end-user and a MessageForm is shown to the end-user containing a message that is retrieved from the localization data by the given message key.
ViewSecurityKey set, permission denied, denied action set to ReplaceEachChar	The form is not shown to the end-user and a MessageForm is shown to the end-user containing the DefaultBlockedMessage.
ViewSecurityKey set, permission set to read-only	The form is not shown to the end-user and a MessageForm is shown to the end-user containing the DefaultBlockedMessage.
ViewSecurityKey set, user changes while app is running, permission denied	The form remains shown, but is covered by a "Form Locked" panel that prevents the contents of the form from being displayed. Additionally, the form cannot be closed until a user with sufficient rights to view the form logs back into the application.

Business Object-Level Security Keys

Purpose: The purpose of this document is to describe the process of assigning security keys to business objects and to describe how the framework utilizes these security keys when an action is requested by the end-user.

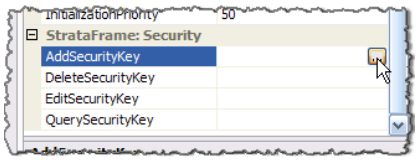
Business Object Security Keys

- AddSecurityKey - Defines the permission that is required to add a new record to a business object.
- DeleteSecurityKey - Defines the permission that is required to delete a record from a business object.
- EditSecurityKey - Defines the permission that is required to edit a record on the business object.
- QuerySecurityKey - Defines the permission that is used to determine whether SQL queries that are processed through this business object should be audited.

Like all business object properties, the business object security key properties can be set within the component designer for the business object to affect all instances of the business object, or set directly on an instance of a business object that has been dropped on a form to set the property for that one instance only.

Assigning the Key

Assigning the business object security key properties can be done through the property sheet of the business object's component designer or programmatically. The Security Key Type Editor is used to search for the appropriate security key to assign to the values for the properties.



Sample - Business Object Security Keys [Visual Basic]

```
Public Sub TestSecurity()
    '-Create a new business object
    Dim loBo As New MyBusinessObject()

    '-Test the Add security
    loBo.AddSecurityKey = "MyAddSecurityKey"
    loBo.Add()

    '-Test the Edit security
    loBo.EditSecurityKey = "MyEditSecurityKey"
    loBo.Edit()

    '-Test the Delete security
    loBo.DeleteSecurityKey = "MyDeleteSecurityKey"
    loBo.DeleteCurrentRow()
End Sub
```

Sample - Business Object Security Keys [C#]

```
public void TestSecurity()
{
    //-- Create a new business object
    MyBusinessObject loBo = new MyBusinessObject();
```

```
    //-- Test the Add security
    loBo.AddSecurityKey = "MyAddSecurityKey";
    loBo.Add();

    //-- Test the Edit security
    loBo.EditSecurityKey = "MyEditSecurityKey";
    loBo.Edit();

    //-- Test the Delete security
    loBo.DeleteSecurityKey = "MyDeleteSecurityKey";
    loBo.DeleteCurrentRow();
}
```

How the Security Keys Affect Business Objects at Run-time

Each security key property has a corresponding method that performs an action on the business object (AddSecurityKey affects calls to Add(), EditSecurityKey affects calls to Edit()). These methods, Add(), Edit(), and DeleteCurrentRow() have overloads that accept Boolean value that indicate whether the business object should check the current user's permissions before performing the action. The default overload checks the users permissions.

When the security is checked, and the permission is denied, the SecurityDenied event will be raised. This event is automatically handled by the StandardForm class to show a message to the end user depending upon the denied action of the requested permission. This automation can be disabled so that you can handle the denying of the requested action manually.

Security Documentation

Send comments on this topic.

Field-Level Security Keys

Purpose: The purpose of this document is to describe the process of assigning security keys to fields on business objects and describing how bound controls are affected fields that have a security key assigned to them.

Field-Level Security Keys

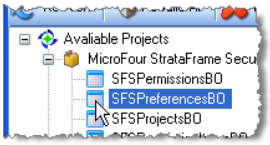
The security keys assigned to fields within a business object are stored within a shared dictionary that is declared within the partial class for each business object type. This allows the collection of keys to be shared across business object instances that are the same type.

Assigning Field-Level Keys using the BOMapper

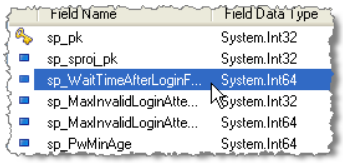
The Business Object Mapper is used to assign security keys to fields within a business object.

To assign a security key to a field:

1. Open the Visual Studio solution for your application.
2. Select `Business Object Mapper` from the `StrataFrame` menu.
3. All created business objects will be available in the left panel under their associated project. Select the desired business object from the available options.



4. The fields for the selected business object will become available in the right panel. Select the desired field from the available options.



5. Edit the Custom Field Properties by right-clicking on the field and selecting `Customize Field...` from the context menu
   OR
   Clicking the `Customize Field` command button.
6. The security key is entered in the `Security Key` text box. Type the security key in the Security Key textboxo
   OR
   Click the browse (...) button to bring up the Security Key Type Editor to browse for the security key.
7. Click `OK` to confirm the changes.

Note: More information on editing custom field properties within the Business Object Mapper can be found in the Business Layer section of the StrataFrame help file.

Assigning Security Keys Programmatically

The recommended method of assigning security keys programmatically is to override the `GetCustomBindablePropertyDescriptors()` method within a business object to return a collection of property descriptors that describe the custom properties that belong to the business object. This method is only called once per business object type, so it is the most logical place to programmatically set the security keys for fields within the business object.

Note: Programmatically setting the security keys for properties is normally only required when you create a custom property and need to set the security key for that custom property. Since you will need to use the GetCustomBindablePropertyDescriptors() method to return the property descriptor for the custom property, it is logical to also set the security key for the custom property within this method.

Sample - Setting the Security Key on a Custom Property Programmatically [Visual Basic]

```
Imports MicroFour.StrataFrame.Business
...
Protected Overrides Function GetCustomBindablePropertyDescriptors() As FieldPropertyDescriptor()
```

```
    '-- Set the security key for the custom field property
    _FieldPermissionKeys.Add("MyCustomField", "MySecurityKey")

    '-- Return the property descriptor for the custom field
    Return New FieldPropertyDescriptor() {New ReflectionPropertyDescriptor( _
        "MyCustomField", Me.GetType())}
End Function
```

Sample - Setting the Security Key on a Custom Property Programmatically [C#]

```
using MicroFour.StrataFrame.Business;
...
protected override FieldPropertyDescriptor[] GetCustomBindablePropertyDescriptors()
{
    //-- Set the security key for the custom field property
    _FieldPermissionKeys.Add("MyCustomField", "MySecurityKey");

    //-- Return the property descriptor for the custom field
    return new FieldPropertyDescriptor[] {new ReflectionPropertyDescriptor(
        "MyCustomField", Me.GetType())};
}
```


Security Documentation

Send comments on this topic.

Retrieving Global Preferences from the Database

Purpose: This document describes the process of retrieving the global preferences from the database for use within the application.

Overview

All global preferences are stored within the `MicroFour.StrataFrame.Security.SecurityBasics` static class. The properties are stored in the database within the `SFSPreferences` table.

Retrieving Global Preferences from the Database

A shared (static) method is provided to retrieve global preferences from the database. Executing `MicroFour.StrataFrame.Security.BusinessObjects.SFSPreferencesBO.RetrieveSecurityPreferences()` retrieves the preferences from the SFSPreferences table in the database and populates the properties in the SecurityBasics class.

Global Preferences Properties

The global preferences are accessed through the following properties on the `SecurityBasics` class:

Property	Database Field	Description
AllowWindowsAuth	sp_AlowWindowsAuth	Determines whether the application as a whole will allow users to be authenticated against active directory. Also determines whether the domain combo box will be shown on the logon form.
InvalidLogonRetryDelay	sp_WaitTimeAfterLogonFailure	The time that the logon form will be disabled after a user enters a bad user name and password.
MaximumInvalidLogonAttempts	sp_MaxInvalidLogonAttempts	The maximum number of times that an end-user can enter an incorrect password for the same user name before the user is deactivated.
MaximumInvalidLogonAttemptsTime	sp_MaxInvalidLogonAttemptsTime	The TimeSpan that specifies the amount of time in which the MaximumInvalidLogonAttempts must occur for the account to be disabled.
PasswordMaximumAge	sp_PwMaxAge	The maximum age of a password before it must be changed (the amount of time after which a password will expire).
PasswordMaximumLength	sp_PwMaxLength	The maximum length of newly created passwords within the application.
PasswordMinimumAge	sp_PwMinAge	The minimum amount of time that an end-user must wait between password changes.
PasswordMinimumLength	sp_PwMinLength	The minimum length of newly created passwords within the application.
PasswordMustBeComplex	sp_PwAreComplex	Determines whether newly created passwords must follow the password complexity rules specified by Windows® Server 2003.
SessionTimeout	sp_SessionTimeout	The default time that a user's session can remain idle before the session will timeout and lock. (Session locking must be enabled for a user's session to lock.)
UniquePasswordsBeforeRepeat	sp_PwBeforeRepeat	The number of unique passwords that an end-user must supply before they can repeat a password within their password history.

Security Documentation

Send comments on this topic.

Showing the Initial Login Form

Purpose: This document describes the steps necessary to show the initial logon form of a StrataFrame application.

Showing the Login Form

Showing the initial login form within a StrataFrame application is accomplished through the MicroFour.StrataFrame.Security.Login class. This class has a LoginFormType property that accepts the System.Type of the form that will be shown as the login form. This form type defaults to MicroFour.StrataFrame.Security.LoginForm and can be set to any custom login form that implements the MicroFour.StrataFrame.Security.ILoginForm interface.

Once the LoginFormType has been set (or left to the default value), you can call the Login.ShowLoginAndAuthUser() method to show the login form and authenticate a user into the system.

The method will return a Boolean value indicating whether the end-user was authenticated or he/she canceled out of the form and the application should exit.

The ShowLoginAndInitForm() Method

The ShowLoginAndInitMainForm() method within the AppMain.vb (Program.cs) file is provided to give a location to show the initial login form for the application. The following sample shows how to show the login form for the application.

Sample - Showing the Initial Login Form [Visual Basic]

```
Imports MicroFour.StrataFrame.Security.BusinessObjects
...
Private Shared Sub ShowLoginAndInitMainForm( _
         ByVal e As ShowLoginAndInitFormEventArgs)

         '-- Set the login form type to the appropriate form type (optional)
         Login.LoginFormType = GetType(MyLoginForm)

         '-- Show the login form and set the return value back to e.ShowMainForm
         '   The true value passed to the method tells the method to allow the
         '   end-user to cancel the login form, which will return False from the
         '   method.
         e.ShowMainForm = Login.ShowLoginAndAuthUser(True)
End Sub
```

Sample - Showing the Initial Login Form [C#]

```
using MicroFour.StrataFrame.Security;
...
private static void ShowLoginAndInitMainForm(ShowLoginAndInitFormEventArgs e)
{

         //-- Set the login form type to the appropriate form type (optional)
         Login.LoginFormType = typeof(MyLoginForm);

         //-- Show the login form and set the return value back to e.ShowMainForm
         //   The true value passed to the method tells the method to allow the
         //   end-user to cancel the login form, which will return False from the
         //   method.
         e.ShowMainForm = Login.ShowLoginAndAuthUser(true);
}
```


Purpose: This document describes the code required to initialize the session locking feature of the StrataFrame Security module.

Initializing Session Locking

The SessionLock class contains the code necessary to enable the session locking feature of the StrataFrame Security module. The SessionLock.StartSessionMonitoring() method is used to insert the low-level API hooks into the Windows process to allow the SessionLock class to track user input and monitor the session timeout. The full details of the SessionLock class can be found here.

In addition to inserting the low-level hooks and starting the session monitoring, the session lock class contains the SessionLockKey property that allows you to specify a key or key combination that can be used to quickly lock the application on a terminal just as if the end-user's session had timed out.

The following sample shows the code that should be added to the InitApplication() method within the AppMain.vb (Program.cs) file to start the session monitoring and enable session locking within the application.

Sample - Starting Session Monitoring [Visual Basic]

```
Imports MicroFour.StrataFrame.Security.BusinessObjects
...
Private Shared Sub InitApplication(ByVal e As InitializingApplicationEventArgs)
         '-- Add the low-level hooks to start the session monitoring
         SessionLock.StartSessionMonitoring()

         '-- Set the quick lock key to F11 (or any other key)
         SessionLock.SessionLockKey = Keys.F11
End Sub
```

Sample - Starting Session Monitoring [C#]

```
using MicroFour.StrataFrame.Security;
...
private static void InitApplication(InitializingApplicationEventArgs e)
{
         //-- Initialize the session locking
         SessionLock.StartSessionMonitoring();
         SessionLock.SessionLockKey = Keys.F11;
}
```

Security Documentation

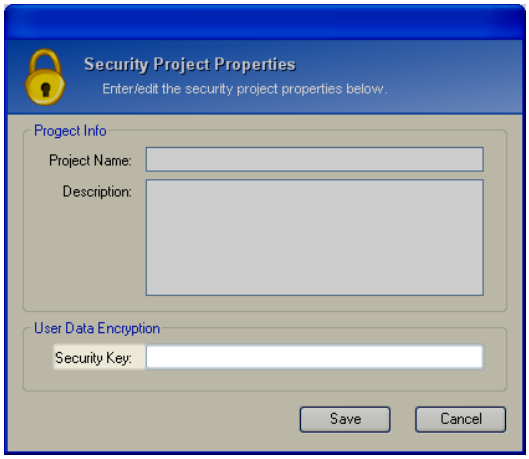
Send comments on this topic.

Specifying the Encryption Key for User Data

Purpose: This document describes the process of specifying the encryption key used to encrypt the user data for users stored in the StrataFrame security tables.

Specifying the Encryption Key for User Data

User data is stored within the SFSUsers database table using 3DES encryption. The key for the 3DES encryption is specified by using the SecurityBasics.SetSecurityKeyAndVectorForUserAuthentication() method. This method creates a hash of the given key and uses it at the System.Byte[] that is used for the encryption key and the initialization vector of the 3DES algorithm.



If you are deploying users as initial data to your application's database (users that were created at design-time using the StrataFrame Role-based security editor), then the string value that is passed to the SetSecurityKeyAndVectorForUserAuthentication method must be the same value as that specified as the security key for the security project.

The SetSecurityKeyAndVectorForUserAuthentication method is generally called from the InitApplication() method within the AppMain.vb (Program.cs) file. The following sample illustrates the use of this method.

Sample - Calling SetSecurityKeyAndVectorForUserAuthentication() [Visual Basic]

```
Imports MicroFour.StrataFrame.Security.BusinessObjects
...
Private Shared Sub InitApplication(ByVal e As InitializingApplicationEventArgs)
        '-- Set the security key for the user data encryption
        SecurityBasics.SetSecurityKeyAndVectorForUserAuthentication("MyKey")
End Sub
```

Sample - Calling SetSecurityKeyAndVectorForUserAuthentication() [C#]

```
using MicroFour.StrataFrame.Security;
...
private static void InitApplication(InitializingApplicationEventArgs e)
{
        '-- Set the security key for the user data encryption
        SecurityBasics.SetSecurityKeyAndVectorForUserAuthentication("MyKey");
}
```


Security Documentation

Send comments on this topic.

Setting the SecurityDataSourceKey

Purpose: This document describes the process needed to set the data source key that is used to access the StrataFrame security tables.

Setting the SecurityDataSourceKey

The StrataFrame security module requires access to the StrataFrame security tables within some SQL Server database. The SecurityBasics.SecurityDataSourceKey property specifies the key of the DbDataSourceItem that will be used to access the StrataFrame security tables. If the StrataFrame security tables are included within your application's database and the data source key for the database is the default of System.String.Empty (""), then the SecurityBasics.SecurityDataSourceKey does not need to be set because it defaults to an empty string (""). However, if you place the StrataFrame security tables in a database that is outside the application's main database or in a database that is accessed using a different data source key, then the SecurityBasics.SecurityDataSourceKey property must be set to the proper value for the application framework to access the security tables.

The SecurityBasics.SecurityDataSourceKey is generally set anywhere within either the SetDataSources() method or the InitApplication() method, both of which can be found within the AppMain.vb (Program.cs) file. The following sample shows how to set the SecurityDataSourceKey property within the SetDataSources() method.

Sample - Setting the SecurityDataSourceKey Property [Visual Basic]

```
Imports MicroFour.StrataFrame.Security.BusinessObjects
...
Private Shared Sub SetDataSources()
        '-- Set the data sources for the application
        DataLayer.DataSources.Add(New SqlDataSourceItem("", "main connstring"))
        DataLayer.DataSources.Add(New SqlDataSourceItem("SECURITY", "security connstring"))
```

```
        '-- Set the SecurityDataSourceKey property to indicate the data source that
        '   will be used to access the StrataFrame security tables
        SecurityBasics.SecurityDataSourceKey = "SECURITY"
End Sub
```

Sample - Setting the SecurityDataSourceKey Property [C#]

```
using MicroFour.StrataFrame.Security;
...
private static void SetDataSources()
{
        //-- Set the data sources for the application
        DataLayer.DataSources.Add(New SqlDataSourceItem("", "main string"));
        DataLayer.DataSources.Add(New SqlDataSourceItem("SECURITY", "security string"));

        //-- Set the SecurityDataSourceKey property to indicate the data source that
        //   will be used to access the StrataFrame security tables
        SecurityBasics.SecurityDataSourceKey = "SECURITY";
}
```


Security Documentation

Send comments on this topic.

Retrieving Global Preferences from the Database

Purpose: This document describes the process of retrieving the global security preferences from the database and storing them in the proper shared (static) properties of the SecurityBasics class.

Retrieving Global Preferences

Global preferences are stored within the database in the SFSPreferences table; however, all global preferences are referenced through shared (static) properties on the SecurityBasics class. The MicroFour.StrataFrame.Security.BusinessObjects.SFSPreferencesBO.RetrieveSecurityPreferences() method is provided to retrieve all of the global security preferences from the SFSPreferences table and store them in the appropriate properties on the SecurityBasics class. For more information on the global preferences, click here.

The RetrieveSecurityPreferences() method should be called after the SecurityDataSourceKey property is set to ensure that the SFSPreferencesBO object can access the SFSPreferences table. The RetrieveSecurityPreferences() method is generally called within the InitApplication() method of the AppMain.vb (Program.cs) file. The following sample shows an example of calling the RetrieveSecurityPreferences() method.

Sample - RetrieveSecurityPreferences() [Visual Basic]

```
Imports MicroFour.StrataFrame.Security.BusinessObjects
...
Private Shared Sub InitApplication(ByVal e As InitializingApplicationEventArgs)
        '-- Retrieve the security preferences
        SFSPreferencesBO.RetrieveSecurityPreferences()
End Sub
```

Sample - RetrieveSecurityPreferences() [C#]

```
using MicroFour.StrataFrame.Security.BusinessObjects;
...
private static void InitApplication(InitializingApplicationEventArgs  e)
{
        //-- Retrieve the security preferences
        SFSPreferencesBO.RetrieveSecurityPreferences();
}
```


Security Documentation

Send comments on this topic.

CurrentUser Overview

Purpose: This document provides an overview of the use of the SecurityBasics.CurrentUser property that contains an object reference to the currently logged in user for the application.

CurrentUser Overview

The MicroFour.StrataFrame.Security.SecurityBasics.CurrentUser property contains an object reference to the currently logged on user for the application. This property defaults to an instance of the MicroFour.StrataFrame.Security.AdminUser class so that all permissions will be granted unless the CurrentUser is set to another object. This default functionality provides support for the application framework when security is not being used by the application.

The SecurityBasics.CurrentUser property returns an object reference that implements the MicroFour.StrataFrame.Security.ISecurityUser interface. This interface describes methods and properties that can be used to access:

- User's primary key (UserID or UserPK)
- User's login name
- User's login time
- User's session lockout time
- User's permissions

For more information on accessing the current user's permissions, see Accessing Permissions Programmatically, and for more information on accessing the user's properties, see Accessing CurrentUser Information.

The classes within the StrataFrame application framework and the StrataFrame security module that implement the ISecurityUser interface are:

- AdminUser - When the built-in administrator credentials are used to log into the application.
- SecurityMaintenanceUser - When the built-in security maintenance credentials were used to log into the application.
- LoggedInUser - When a standard user's credentials were used to log into the application.

Purpose: This document describes how to programmatically access the current user's permissions.

PermissionInfo Class

The MicroFour.StrataFrame.Security.PermissionInfo class contains all of the information needed to describe a permission that is assigned to a user, including the action to take, and the blocked message if necessary. When you access an assigned permission at runtime, a PermissionInfo object will be returned. The permission info class contains the following properties:

Property	Description
Action	The PermissionAction that determines that action that should be taken for this permission (Grant, Deny, Read-only.)
BlockedMessageOrKey	The message that will be shown to the end-user if the Action is Deny and the DenyAction property is either Message or MessageKey. If the DenyAction is Message, then this property is the actual message. If the DenyAction is MessageKey, then this property contains the localization key that will be used to retrieve the message.
DenyAction	The specific action to perform if the Action is Deny to inform the end-user that their access is denied. Possible values are NoMessage, Message, MessageKey, or ReplaceEachChar.

CurrentUser.GetPermission()

The MicroFour.StrataFrame.Security.SecurityBasics.CurrentUser contains the GetPermission() method that is used to programmatically retrieve the PermissionInfo for a given permission key. The method accepts a single string parameter that indicates the permission key for which you want to retrieve the PermissionInfo. Once the permission info is retrieved, you can determine the action that should be performed.

Sample – GetPermission() [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Sub CheckMyPermission()
    '-- Retrieve the permission and test it
    If SecurityBasics.CurrentUser.GetPermission("MyPerm").Action = _
        PermissionAction.Grant Then

        '-- Enable the control
        Button1.Enabled = True
    Else
        '-- Disable the control
        Button1.Enabled = False
    End If
End Sub
```

Sample – GetPermission() [C#]

```
using MicroFour.StrataFrame.Security;
...
private void CheckMyPermission()
{
    //-- Retrieve the permission and test it
    if (SecurityBasics.CurrentUser.GetPermission(@"MyPerm").Action ==
        PermissionAction.Grant)
    {
        //-- Enable the control
        Button1.Enabled = true;
    }
    else
    {
        //-- Disable the control
        Button1.Enabled = false;
    }
}
```

Determining Permissions

When a user logs into the system, a flat list of his/her permissions is compiled according to the permissions assigned to the user through his/her assigned roles and directly assigned permissions. The permission granted to the user is determined by the hierarchy of the assigned permissions (link). If the permission is not assigned to the user, then the default permission is assigned.

Purpose: This document describes how to access information relevant to the CurrentUser within the application.

CurrentUser Members

The MicroFour.StrataFrame.Security.SecurityBasics.CurrentUser property returns an ISecurityUser object that can be used to access information about the currently logged in user. The following members provide access to the common information:

Member	Description
GetUserLockoutTime() Method	Retrieves a System.TimeSpan indicating the amount of idle time the system will wait before the user's session times out and locks.
LoggedOnAt Property	Gets a Date that indicates the timestamp of when the user logged into the application.
UserName Property	Gets the username that identifies the user (the login name, not the full name).
UserPK Property	Gets the primary key that uniquely identifies the user (UserID).

Purpose: This document shows how to retrieve user information from the security database for display or reporting purposes.

Accessing Information for Users other than the CurrentUser

There are times when it is necessary to retrieve information for a user that is different than the current user. For instance, the primary key of the user that performed an inventory operation is recorded within the database. When a report is created detailing the inventory operation, the user's username or full name might be retrieved so that it can be added to the report.

The MicroFour.StrataFrame.Security.BusinessObjects.SFSUsersBO business object contains static (shared) methods that allow user information to be retrieved:

Method	Description
RetrieveUserName()	Accepts a user primary key and returns the user name (login name) for the user.
RetrieveFullName()	Accepts a user primary key and a Boolean value indicating whether the last name should be displayed first and returns the full name for the user.

Accessing Additional Information

Additional information beyond the user's login name and the full name of the user is stored within the us_Data field in the database. This field is encrypted and the pieces of data stored within the field can only be accessed by accessing them through the strong-typed properties on the SFSUsersBO business object. Therefore, to access additional information, you must create a new instance of the SFSUsersBO business object, populate it with the appropriate user record(s), and access the data through that business object.

Sample – Accessing Additional User Data [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
Imports MicroFour.StrataFrame.Security.BusinessObjects
...
Private Sub ShowUserData(ByVal UserPK As Integer)
    '-- Create the user business object
    Dim user As New SFSUsersBO()

    '-- Get the user
    user.FillByPrimaryKey(UserPK)

    '-- Display the user's password and expiration
    MsgBox("Password: " & user.us_PasswordPlainText & ControlChars.CrLf & _
        "Never Expires: " & user.us_PasswordNeverExpires.ToString())

    '-- Dispose of the bo
    user.Dispose()
End Sub
```

Sample – Accessing Additional User Data [C#]

```
using MicroFour.StrataFrame.Security;
using MicroFour.StrataFrame.Security.BusinessObjects;
...
private void ShowUserData(int UserPK)
{
    //-- Create the user business object
    SFSUsersBO user = new SFSUsersBO();

    //-- Get the user
    user.FillByPrimaryKey(UserPK);

    //-- Display the user's password and expiration
    MessageBox.Show("Password" + user.us_PasswordPlainText + "\n" +
        "Never Expires: " + user.us_PasswordNeverExpires.ToString());

    //-- Dispose of the bo
    user.Dispose();
}
```

Security Documentation

Send comments on this topic.

Built-In Accounts

Purpose: This document provides information on configuring and using the built-in accounts within the StrataFrame security module.

Built-In Accounts

The StrataFrame security module allows you to specify the username and password for two built-in accounts to the application. The build-in accounts are disabled by default and both the username and password must be specified in order for the account to be active.

Administrator Account

The first account is an administrator account that has access to everything within the application (every permission is granted). This user is the default user that is set to the CurrentUser property (therefore, if the security module is not used, everything within the application is accessible). The built-in administrator account is generally used as a back door to the application.

Tip: It is generally recommended that the administrator's password not be a constant; an algorithm that calculates the password according to the day of the year or time of day is much more secure.

Security Maintenance Account

The second account is a security maintenance account that provides access to only the portions of the application that are used to maintain user information. This account is generally used as a bootstrap for the initial setup of the users for the application (so you do not have to deploy any initial users for the application).

Configuring the Built-In Accounts

Configuring the built-in accounts requires 3 steps:

1. Set the account username.
2. Set the account password.
3. Specify the pseduo primary key (user id) for the built-in account.

Sample – Configuring the Built-In Accounts [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Shared Sub InitApplication(ByVal e As InitializingApplicationEventArgs)
    '-- Configure the Administrator Account
    SecurityBasics.AdministratorUserName = "admin"
    SecurityBasics.AdministratorPassword = "pass" & DateTime.Now.Day.ToString()
    SecurityBasics.AdministratorUserPk = -1

    '-- Configure the security maintenance user
    SecurityBasics.SecurityMaintenanceUserName = "securitysetup"
    SecurityBasics.SecurityMaintenancePassword = "security"
    SecurityBasics.SecurityMaintenanceUserPk = -2
End Sub
```

Sample – Configuring the Built-In Accounts [C#]

```
using MicroFour.StrataFrame.Security;
...
private static void InitApplication(InitializingApplicationEventArgs e)
{
    //-- Configure the Administrator Account
    SecurityBasics.AdministratorUserName = "admin";
    SecurityBasics.AdministratorPassword = "pass" + DateTime.Now.Day.ToString();
    SecurityBasics.AdministratorUserPk = -1;

    //-- Configure the security maintenance user
    SecurityBasics.SecurityMaintenanceUserName = "securitysetup";
    SecurityBasics.SecurityMaintenancePassword = "security";
    SecurityBasics.SecurityMaintenanceUserPk = -2;
}
```


Security Documentation

Send comments on this topic.

Default Settings

Purpose: This document indicates the default values for most of the properties on the SecurityBasics class.

SecurityBasics Default Property Values

The following table lists the default values for the properties on the SecurityBasics class.

Property	Default Value
AdministratorPassword	String.Empty ("")
AdministratorUserName	"Administrator"
AdministratorUserPk	-1
AllowWindowsAuth	False
BlockedReplacementCharacter	'x'
BlockedReplacementRegex	[A-Za-z0-9@]

CurrentUser	AdminUser
DefaultBlockedMsg	"Access Denied"
DefaultBlockedMsgKey	String.Empty ("")
DefaultPermissionAction	Grant
DefaultPermissionInfo	PermissionInfo.GrantedInfo
InvalidLoginRetryDelay	5 seconds
MaximumInvalidLoginAttempts	4
MaximumInvalidLoginAttemptsTime	5 minutes
PasswordMaximumAge	42 days
PasswordMaximumLength	16
PasswordMinimumAge	2 days
PasswordMinimumLength	5
PasswordMustBeComplex	True
SecurityDataSourceKey	String.Empty ("")
SecurityKey	New Byte() {}
SecurityKeyVector	New Byte() {}
SecurityMaintenanceKeyPrefix	"Security_"
SecurityMaintenancePassword	String.Empty ("")
SecurityMaintenanceUserName	"AdminSecurity"
SecurityMaintenanceUserPk	-2
SessionTimeout	30 minutes
UniquePasswordsBeforeRepeat	10

Security Documentation

Send comments on this topic.

Login Class Methods

Purpose: This document describes the use of the Login class and its methods.

Login Class Methods

The MicroFour.StrataFrame.Security.Login class is a sealed, static class that contains shared (static) methods that are used to authenticate users against the security system. There are 4 main methods within the Login class that you will use most often:

- AuthenticateUser()
- SetLoggedInUser()
- ShowLoginAndAuthUser()
- ShowPasswordChangeForm()

AuthenticateUser()

The AuthenticateUser() method accepts a users credentials and returns a LoginResult of the authentication results. It does not change the SecurityBasics.CurrentUser object, but instead returns a new SFSUsersBO object through the ByRef (ref) UserInfo parameter containing the information on the current user if the user was successfully authenticated.

Sample - Authenticating a User [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
Imports MicroFour.StrataFrame.Security.BusinessObjects
...
Private Sub AuthUser(ByVal UserName As String, ByVal Password As String)
    '-- Establish locals
        Dim loUserInfo As SFSUsersBO = Nothing
        Dim loResult As Login.LoginResult

        '-- Attempt to authenticate the user
        loResult = Login.AuthenticateUser(UserName, Password, "", loUserInfo)

        '-- Do something based upon the result
        Select Case loResult
            Case Login.LoginResult.Success

            Case Login.LoginResult.Failure

            Case Login.LoginResult.UserDeactivated
            ...
        End Select
End Sub
```

Sample - Authenticating a User [C#]

```
using MicroFour.StrataFrame.Security;
using MicroFour.StrataFrame.Security.BusinessObjects;
...
private void AuthUser(string UserName, string Password)
```

```
{
    //-- Establish locals
    SFSUsersBO loUserInfo = null;
    Login.LoginResult loResult;

    //-- Attempt to authenticate the user
    loResult = Login.AuthenticateUser(UserName, Password, "", ref loUserInfo);

    //-- Do something based upon the result
    switch (loResult)
    {
        case Login.LoginResult.Success:

        case Login.LoginResult.Failure:

        case Login.LoginResult.UserDeactivated:

        ...
    }
}
```

SetLoggedInUser()

The SetLoggedInUser() method accepts a user's credentials and returns a `LoginResult` of the authentication results. This method will also set the `CurrentUser` object to the appropriate LoggedInUser, AdminUser, or SecurityMaintenanceUser object if the user is successfully authenticated.

Sample - Programmatically Changing the Current User [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Sub SetUser(ByVal UserName As String, ByVal Password As String)
    '-- Establish locals
         Dim loResult As Login.LoginResult

         '-- Attempt to authenticate the user
         loResult = Login.SetLoggedInUser(UserName, Password, "")
         '-- If the result if Success, AdminLoggedOn, or SecMaintUserLoggedOn, the
         '   SecurityBasics.CurrentUser object will be changed to the correct user

         '-- Do something based upon the result
         Select Case loResult
             Case Login.LoginResult.Success

             Case Login.LoginResult.Failure

             Case Login.LoginResult.UserDeactivated
             ...
         End Select
End Sub
```

Sample - Programmatically Changing the Current User [C#]

```
using MicroFour.StrataFrame.Security;
...
private void SetUser(string UserName, string Password)
{
    //-- Establish locals
    Login.LoginResult loResult;

    //-- Attempt to authenticate the user
    loResult = Login.SetLoggedInUser(UserName, Password, "");
         //-- If the result if Success, AdminLoggedOn, or SecMaintUserLoggedOn, the
         //   SecurityBasics.CurrentUser object will be changed to the correct user

    //-- Do something based upon the result
    switch (loResult)
    {
        case Login.LoginResult.Success:

        case Login.LoginResult.Failure:

        case Login.LoginResult.UserDeactivated:

        ...
    }
}
```

ShowLoginAndAuthUser()

The ShowLoginAndAuthUser() method is used to display the LoginForm currently set as the `Login.LoginFormType` to the end-user, gather the user's credentials, authenticate the user, and set the `CurrentUser` object to the appropriate user object. This method accepts a Boolean parameter that determines whether the end-user should be allowed to cancel out of the form to close the application. It is generally called from within the `ShowLoginAndInitMainForm()` method of the AppMain.vb (program.cs) file.

Sample - Showing a Login Form [Visual Basic]

```
Private Shared Sub ShowLoginAndInitMainForm(ByVal e As ShowLoginAndInitFormEventArgs)
         '-- Set the login form to your custom login form (optional)
         Login.LoginFormType = GetType(MyCustomLoginForm)

         '-- Show the login form and authenticate the user
         e.ShowMainForm = Login.ShowLoginAndAuthUser(True)
End Sub
```

Sample - Showing a Login Form [C#]

```
private static void ShowLoginAndInitMainForm(ShowLoginAndInitFormEventArgs e)
```

```
{
        //-- Set the login form to your custom login form (optional)
        Login.LoginFormType = typeof(MyCustomLoginForm);

        //-- Show the login form and authenticate the user
        e.ShowMainForm = Login.ShowLoginAndAuthUser(true);
}
```

ShowPasswordChangeForm()

The ShowPasswordChangeForm() method shows a password change dialog to the end-user to allow them to change their password. All security preferences are followed when the end-user attempts to set their password (password history, max length, min length, complexity, etc.). This method returns a Boolean value indicating whether the end-user's password was successfully changed or whether the user canceled out of the form.

Sample - Showing the Password Change Dialog [Visual Basic]

```
Private Function ChangeUsersPassword() As Boolean
    Return Login.ShowPasswordChangeForm()
End Function
```

Sample - Showing the Password Change Dialog [C#]

```
private bool ChangeUsersPassword()
{
    return Login.ShowPasswordChangeForm();
}
```

Security Documentation

Send comments on this topic.

Login Class Properties

Purpose: This document describes properties available within the Login class.

Login Class Properties

The Login class contains two shared (static) properties that can be used to control settings pertaining to the logging of users into your application.

- LoginFormType
- LoginSecurityKey

LoginFormType

The LoginFormType property gets or sets a `System.Type` object that represents the form type of the login form that will be presented to the end-user to gather the user's credentials. The default value for this property is `MicroFour.StrataFrame.Security.LoginForm` which is the included login form. If you create a custom login form for your application you will need to set this property to your custom login form type before the first call to a StrataFrame security method that will show a login form.

Sample - Setting the LoginFormType before the First Call to ShowLoginAndAuthUser [Visual Basic]

```
Private Shared Sub ShowLoginAndInitMainForm(ByVal e As ShowLoginAndInitFormEventArgs)
        '-- Set the login form to your custom login form (optional)
        Login.LoginFormType = GetType(MyCustomLoginForm)

        '-- Show the login form and authenticate the user
        e.ShowMainForm = Login.ShowLoginAndAuthUser(True)
End Sub
```

Sample - Setting the LoginFormType before the First Call to ShowLoginAndAuthUser [C#]

```
private static void ShowLoginAndInitMainForm(ShowLoginAndInitFormEventArgs e)
{
        //-- Set the login form to your custom login form (optional)
        Login.LoginFormType = typeof(MyCustomLoginForm);

        //-- Show the login form and authenticate the user
        e.ShowMainForm = Login.ShowLoginAndAuthUser(true);
}
```

LoginSecurityKey

The LoginSecurityKey property gets or sets the key of the permission that is required for users to be allowed to login to the application. The default value for this property is an empty string ("") which ensures that it will always be Granted to the CurrentUser object.

It is not recommended to globally control whether a user can log into the application (you should deactivate a user to completely prevent them from logging in) but is meant to be used in conjunction with a restriction set to limit where (workstation) and when (hours of operation) the user can log into the application. To use this property, you must create a permission within the application's permissions and specify the name when setting this property.

Sample - Setting the LoginFormType before the First Call to ShowLoginAndAuthUser [Visual Basic]

```
Private Shared Sub ShowLoginAndInitMainForm(ByVal e As ShowLoginAndInitFormEventArgs)
        '-- Set the permission required to login to the application
        Login.LoginSecurityKey = "MyLoginKey"

        '-- Set the login form to your custom login form (optional)
        Login.LoginFormType = GetType(MyCustomLoginForm)

        '-- Show the login form and authenticate the user
        e.ShowMainForm = Login.ShowLoginAndAuthUser(True)
End Sub
```

Sample - Setting the LoginFormType before the First Call to ShowLoginAndAuthUser [C#]

```
private static void ShowLoginAndInitMainForm(ShowLoginAndInitFormEventArgs e)
{
        //-- Set the permission required to login to the application
        Login.LoginSecurityKey = "MyLoginKey";

        //-- Set the login form to your custom login form (optional)
        Login.LoginFormType = typeof(MyCustomLoginForm);

        //-- Show the login form and authenticate the user
        e.ShowMainForm = Login.ShowLoginAndAuthUser(true);
}
```


Security Documentation

Send comments on this topic.

Login.LoginResult Enumeration

Purpose: This document indicates the enumeration values for Login.LoginResult.

Login.LoginResult Enumeration

The following table lists the enumeration values for Login.LoginResult.

Value	Description
Success	Indicates that the user was successfully authenticated. This value is only returned for users that are contained within the database, not the built-in users.
Failure	Indicates that the user was not successfully authenticated. Either the username or password was incorrect.
UserDeactivated	Indicates that the user is inactive or was deactivated. The given username/password/domain combination was valid, but the user was already inactive or expired, and was therefore deactivated.
InvalidLoginsExceeded	Indicates that the same username failed to authenticate and exceeded the maximum number of failed authentications within the allotted time. The account was subsequently deactivated.
AdminLoggedOn	Indicates that the built-in admin username/password was supplied and successfully authenticated.
SecMaintUserLoggedOn	Indicates that the build-in security maintenance username/password was supplied and successfully authenticated.
LoginPermissionDenied	Indicates that the given username/password/domain is valid, but the permission required to login (Login.LoginSecurityKey) was denied to the user.

Security Documentation

Send comments on this topic.

SessionLock Methods

Purpose: This document discusses the SessionLock class's methods and using them to control the current user's session.

SessionLock Methods

The `MicroFour.StrataFrame.Security.SessionLock` class is a sealed, static class that contains shared (static) methods that are used to lock the application session allowing quick user switching and monitor idle user sessions to automatically lock them. There are 7 public methods within the SessionLock class:

- LockSession()
- StartSessionMonitoring()
- StopSessionMonitoring()
- PauseSessionMonitoring()
- ResumeSessionMonitoring()
- PauseSessionTimer()
- ResumeSessionTimer()

LockSession()

The LockSession() method is used to explicitly lock the session. When a session locks, all forms within the application are minimized, and a login form is show that forces the user to re-authenticate or allows another user to log into the system. When a user is successfully authenticated, the application resumes normally. This method can be called explicitly to lock the session and is called internally when the session lock shortcut key is pressed and when a user's session times out.

Sample - Locking the Current Session [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Sub Lock()
    '-- Call the method to lock the session
    '   This call blocks until a user is re-authenticated
    SessionLock.LockSession()
End Sub
```

Sample - Locking the Current Session [C#]

```
using MicroFour.StrataFrame.Security;
```

```
...
private void Lock()
{
    //-- Call the method to lock the session
    //   This call blocks until a user is re-authenticated
    SessionLock.LockSession();
}
```

StartSesionMonitoring() and StopSessionMonitoring()

The StartSessionMonitoring() and StopSessionMonitoring() methods are used in conjunction to start and stop session monitoring respectively. Session monitoring consists of two things:

1. Session Timer - The session timer watches for idle sessions and automatically locks the session if the idle time exceeds the current user's idle timeout.
2. Keyboard Monitoring - Keyboard activity is monitored for the session lock shortcut key. StartSessionMonitoring() installs a low-level Windows input hook into the current process to monitor both keyboard and mouse input and determine when the user's session timed-out. This low-level hook also watches for the session lock shortcut key.

> **Note:** The hook installed by the StartSessionMonitoring() method must be in place for the automatic session monitoring to work. Therefore, the StartSessionMonitoring() method must be called to attach to the Windows hook before the application executes; this is usually done within the InitApplication method of the AppMain.vb (program.cs) file.

StopSessionMonitoring() manually removes the low-level Windows hook. This method is generally never called since the hook is automatically removed when the process exits.

Sample - Starting Session Monitoring [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Shared Sub InitApplication(ByVal e As InitializingApplicationEventArgs)
    '-- Start the session locking monitor
    SessionLock.StartSessionMonitoring()
End Sub
```

Sample - Starting Session Monitoring [C#]

```
using MicroFour.StrataFrame.Security;
...
private static void InitApplication(InitializingApplicationEventArgs e)
{
    //-- Start the session locking monitor
    SessionLock.StartSessionMonitoring();
}
```

PauseSessionMonitoring() and ResumeSessionMonitoring()

The PauseSessionMonitoring() and ResumeSessionMonitoring() are used in conjunction to respectively pause and resume the session monitoring.

When PauseSessionMonitoring() is called, the session timer is paused and the session lock shortcut key is ignored. This method is generally used when the application needs to perform a processing intensive task that requires the computer to be idle for an extended period of time.

For example, if a user requests a report that takes a considerable amount of time to compute, but the user's session timeout is very short, the session might timeout and lock while waiting on the report to complete. In such situations, using the PauseSessionMonitoring() and ResumeSessionMonitoring() methods allows you to wrap critical sections of the application within a Pause/Resume block to make sure that the session does not lock while in that block.

Sample - Pausing Session Monitoring in a Critical Application Section [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Sub ComputeReport()
    '-- Pause the session
    SessionLock.PauseSessionMonitoring()

    '-- Compute the report
    '  This code takes a while to complete and the session should not
    '  be allowed to lock while inside it

    '-- Resume the monitoring
    SessionLock.ResumeSessionMonitoring()
End Sub
```

Sample - Pausing Session Monitoring in a Critical Application Section [C#]

```
using MicroFour.StrataFrame.Security;
...
private void ComputeReport()
{
    //-- Pause the session
    SessionLock.PauseSessionMonitoring();

    //-- Compute the report
    //  This code takes a while to complete and the session should not
    //  be allowed to lock while inside it

    //-- Resume the monitoring
    SessionLock.ResumeSessionMonitoring();
}
```

PauseSessionTimer() and ResumeSessionTimer()

Like the PauseSessionMonitoring() and ResumeSessionMonitoring() methods, the PauseSessionTimer() and ResumeSessionTimer() methods are used in conjunction to pause the session timer and prevent a user's session from automatically timing out during a critical code block within the application. However, unlike the pause/resume monitoring methods, these pause/resume timer methods do not prevent the session lock shortcut key from locking the session, they only pause the automatic timer.

Sample - Pausing the Session Timer While in a Critical Application Block [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Sub ComputeReport()
    '-- Pause the session
```

```
    SessionLock.PauseSessionTimer()

    '-- Compute the report
    '  This code takes a while to complete and the session should not
    '  automatically lock while inside it

    '-- Resume the monitoring
    SessionLock.ResumeSessionTimer()
End Sub
```

Sample - Pausing the Session Timer While in a Critical Application Block [C#]

```
using MicroFour.StrataFrame.Security;
...
private void ComputeReport()
{
    //-- Pause the session
    SessionLock.PauseSessionTimer();

    //-- Compute the report
    //  This code takes a while to complete and the session should not
    //  automatically to lock while inside it

    //-- Resume the monitoring
    SessionLock.ResumeSessionTimer();
}
```


Security Documentation

Send comments on this topic.

SessionLock Properties

Purpose: This document discusses the SessionLock class's properties and their use in monitoring and controlling the current user's session.

SessionLock Properties

The SessionLock class contains 2 shared (static) properties that contain information and control aspects of the session locking:

1. LastUserInput
2. SessionLockKey

LastUserInput

The LastUserInput property is a read-only property that returns the `System.DateTime` of the last user input within the application.

Note: The LastUserInput property only applies to the application on which it is used. In other words, the DateTime returned will not be the value of the last user input within any running application, but will instead be the last user input, either mouse or keyboard, within the StrataFrame application from which it was called.

SessionLockKey

The SessionLockKey property gets or sets a `System.Windows.Forms.Keys` enumeration that represents the keyboard shortcut for the session locking. If the shortcut key is pressed and the SessionLock class is monitoring user input, the session will immediately lock.

The SessionLockKey is generally defined by the application developer and is set during the `InitApplication()` method. However, the key may be canged anywhere within the application and may even be customized by the end-user.

Sample - Setting the SessionLockKey [Visual Basic]

```
Private Shared Sub InitApplication(ByVal e As InitializingApplicationEventArgs)
    '-- Start the session locking monitor & set the quick key to lock the application
    SessionLock.StartSessionMonitoring()
    SessionLock.SessionLockKey = Keys.F11
End Sub
```

Sample - Setting the SessionLockKey [C#]

```
private static void InitApplication(InitializingApplicationEventArgs e)
{
    //-- Start the session locking monitor & set the quick key to lock the application
    SessionLock.StartSessionMonitoring();
    SessionLock.SessionLockKey = Keys.F11;
}
```


Security Documentation

Send comments on this topic.

SessionLock Events

Purpose: This document discusses the SessionLock class's events and their use in monitoring and controlling the current user's session.

SessionLock Events

The SessionLock class contains 2 shared (static) events that provide hooks into the session locking:

1. AfterSessionLock
2. BeforeSessionLock

AfterSessionLock

The AfterSessionLock event is raised immediately after the session lock is released, meaning that a user has re-authenticated and the application is resuming. The AfterSessionLock event is raised after the SecurityBasics.CurrentUserChanged event if a different user has logged into the application.

BeforeSessionLock

The BeforeSessionLock event is raise immediately before the application's session locks. This event is raised regardless of whether the application is being explicitly or automatically locked and provides a hook to enable any custom processing that must take place before the session locks (such as saving off the current user's settings before another user is allowed to log into the application).

Security Documentation

Send comments on this topic.

Differences in Web Security

Purpose: This document describes the differences between security on a web project and security within a Windows project.

Differences in Web Security

Security within a web project is different than security within a Windows project in the following ways:

- CurrentUser - SecurityBasics.CurrentUser must be configured to store an ISecurityUser object for each session.
- Maintenance Forms - To maintain users, roles, and restriction sets, you must either create custom web-based maintenance forms or use the winform-based maintenance forms provided.
- Object Permissions - Object permission function much the same as when using winforms, with the primary exception being the lack of form-level security keys.
- Session Locking - Session locking is not supported within web applications.
- Programmatic Access - Programmatic access within a web project is exactly the same as within a Windows project.

CurrentUser

When using security on the web, the SecurityBasics.CurrentUser property must be configured to store a different ISecurityUser object for each session, rather than just one for the whole AppDomain.

This is accomplished via a configuration setting which indicates that the application is being run within a web environment. This tells the CurrentUser property to use the current session object to retrieve and store the current ISecurityUser. For more information, refer to the Required Global.asax Code topic.

Maintenance Forms

No user, role, or restriction set maintenance forms are available for web projects. Therefore, all users, roles, and selected permissions must either be set using the SecurityDialog in a windows form, or must be set via custom forms created within your web project to maintain user permissions and roles.

Object Permissions

Web-based object permissions function as follows:

- Field-Level - All field-level permissions work identically on WebForms controls as they do on their corresponding WinForm controls.
- Business Object-Level - Business object-level permissions work the same as well, but the forms will not automatically handle the SecurityDenied event of the business objects.
- Form-Level - There is no form-level security key, so permissions must be checked programmatically to prevent the viewing of a complete WebForm.

Session Locking

Session locking is not supported within web applications.

Programmatic Access

Programmatic access within a web project is exactly the same as within a Windows project. The CurrentUser object can be queried for available permissions, and the proper ISecurityUser object will be returned from the CurrentUser property.

Security Documentation

Send comments on this topic.

Required Global.asax Code

Purpose: This document shows the code required inside the Global.asax file to configure the security module for work within a web application.

Required Global.asax Code

All of the code required to configure a StrataFrame web application for use with the StrataFrame Security Module should be placed within the Application_Start() method. This is since the method is only called once, during the first page request after the website's application pool is restarted.

IsWebEnvironment

The MicroFour.StrataFrame.Security.SecurityBasics class contains the shared (static) property IsWebEnvironment. This property is used to indicate to the security module that the security is being run from within a web application and that the session objects should be used to store the CurrentUser and other session specific data. The IsWebEnvironment property should be set at the top of the Application_Start() method within the Global.asax file.

Sample - Setting the IsWebEnvironment Property [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Protected Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
```

```
    '-- Set the property
    SecurityBasics.IsWebEnvironment = True
End Sub
```

Sample - Setting the IsWebEnvironment Property [C#]

```
using MicroFour.StrataFrame.Security;
...
protected void Application_Start(object sender, EventArgs e)
{
    //-- Set the property
    SecurityBasics.IsWebEnvironment = true;
}
```

Other Required Global.asax Code

Other than the `IsWebEnvironment` variable, the code required to enable specific pieces of the security module within a web application is the same as required within a Windows application. Therefore, any AppMain.vb (program.cs) code that is required for a Windows application should be placed within the Global.asax file of a web application.

A complete description of all code required in AppMain.vb (program.cs) may be found under the *Programmatic Access* section of this help file.

Purpose: This document discusses how to authenticate users in a web environment using the StrataFrame security module.

Authenticating Users

Authenticating users within a web application is very similar to authenticating users within a Windows environment; however, no default login form or form template is provided for web applications. As such, authentication must be done programmatically using methods of the `MicroFour.StrataFrame.Security.Login` class.

Note: The reasoning behind the omission of a default login form or login form template is that each web-based login form is fundamentally different depending on requirements. Also most login forms are not just login forms, but frames or regions within more complex forms.

Login Methods Used within a Web Application

The `AuthenticateUser()` and `SetLoggedInUser()` methods of the Login class are typically used to authenticating users within a web application:

- AuthenticateUser() - The AuthenticateUser() method does not change the `CurrentUser` object, but returns a value containing the results of the authentication request.
- SetLoggedInUser() - The SetLoggedInUser() method returns a value containing the results of the authentication request. If the authentication is successful, it also changes the `CurrentUser` object to be the new user.

Sample - Changing the Logged In User - Web Application [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Sub cmdLogin_Click(ByVal sender As Object, ByVal e As EventArgs) Handle cmdLogin.Click
    '-- Establish locals
    Dim loResult As Login.LoginResult

    '-- Attempt to authenticate the user
    loResult = Login.SetLoggedInUser(UserName, Password, "")
    '-- If the result if Success, AdminLoggedOn, or SecMaintUserLoggedOn, the
    '   SecurityBasics.CurrentUser object will be changed to the correct user

    '-- Do something based upon the result
    Select Case loResult
    Case Login.LoginResult.Success

    Case Login.LoginResult.Failure

    Case Login.LoginResult.UserDeactivated
    ...

    End Select
End Sub
```

Sample - Changing the Logged In User - Web Application [C#]

```
using MicroFour.StrataFrame.Security;
...
private void cmdLogin_Click(object sender, EventArgs e)
{
    //-- Establish locals
    Login.LoginResult loResult;

    //-- Attempt to authenticate the user
    loResult = Login.SetLoggedInUser(UserName, Password, "");
    //-- If the result if Success, AdminLoggedOn, or SecMaintUserLoggedOn, the
    //   SecurityBasics.CurrentUser object will be changed to the correct user

    //-- Do something based upon the result
    switch (loResult)
    {
        case Login.LoginResult.Success:

        case Login.LoginResult.Failure:
```

```
        case Login.LoginResult.UserDeactivated:

        ...
    }
}
```


Purpose: Provide a step-by-step procedure on adding security to an existing application.

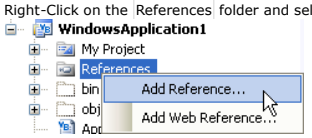
Adding Security to an Existing Application

For a new application, security may be easily included by using the StrataFrame Windows Application w/ Security template when creating the Visual Studio project. However, adding security to existing applications requires some manual configuration.

To add security to an existing application, follow the steps below:

1. Add Security Reference - Add a reference to MicroFour StrataFrame Security within your Visual Studio project.
   a. Open your application's visual studio project.
   b. Within the solution explorer, ensure that the Show All Files option is selected.

   c. Right-Click on the References folder and select the Add Reference option.



   d. Select the MicroFour StrataFrame Security component from the list, and click OK.
2. Add Security Namespaces - Add the security namespaces to AppMain.vb (Visual Basic) or program.cs (C#).

Add Security Namespaces (Visual Basic)

```
Imports MicroFour.StrataFrame.Security
Imports MicroFour.StrataFrame.Security.BusinessObjects
```

Add Security Namespaces (C#)

```
using MicroFour.StrataFrame.Security;
using MicroFour.StrataFrame.Security.BusinessObjects;
```

3. Set Data Source Key - Define and set the Security Data Source Key within the SetDataSources() method of AppMain.vb or program.cs. For a detailed description on the data source key, including what it does and how to set it, refer to the Setting the Security Data Source Key help topic.

Security Data Source Key (Visual Basic)

```
'-- Set the data source key for the security tables
SecurityBasics.SecurityDataSourceKey = ""
```

4. Create and Show the Login Form - The application login form is launched within the AppMain.vb (program.cs) file. If desired, a custom form may be used instead of the default StrataFrame login form.
   a. ShowLoginAndInitMainForm() - The login form must be launched within the ShowLoginAndInitMainForm() method of AppMain.vb or program.cs. For more information on configuring the ShowLoginAndInitMainForm() method, refer to the Showing the Initial Login Form topic.

ShowLoginAndInitMainForm (Visual Basic)

```
Private Shared Sub ShowLoginAndInitMainForm(ByVal e As ShowLoginAndInitFormEventArgs)
    '-- Set the login form to your custom login form (optional)
    'Login.LoginFormType = GetType(MyLoginForm)

    '-- Show the login form and authenticate the user
    e.ShowMainForm = Login.ShowLoginAndAuthUser(True)
End Sub
```

   b. Custom Login Form - If a custom login form is desired, it must be created as a new SF Login Form item within the Visual Studio project. For more information on creating a custom login form, refer to the Creating a Custom Login Form topic.
5. Configure the InitApplication() Method - The following items are added to the InitiApplication() method within the AppMain.vb or program.cs file to configure the remaining security options. The code snippits shown contain all default values for the listed options.
   a. Global Preferences - Required - Retrieve the global preferences from the database using the following method call. For more information, refer to the Retrieving Global Preferences from the Database help topic.

Retrieve Global Preferences(Visual Basic)

```
'-- Retrieve the global preferences
SFSPreferencesBO.RetrieveSecurityPreferences()
```

   b. Set Encryption Key - Required - Set the encryption key to be used for user data using the following method call. For more information, refer to the Specifying the Encryption Key for User Data help topic.

Encryption Key (Visual Basic)

```
'-- Set the encryption key and vector for the user data
SecurityBasics.SetSecurityKeyAndVectorForUserAuthentication("MySecurityKey")
```

c. Configure Session Locking - Possibly Required - Start the session locking monitor and configure the quick lock key using the following lines of code. This is only required if dynamic session locking will be used within your application. For more information, refer to the Initializing Session Locking help topic.

Session Locking (Visual Basic)

```
'-- Start the session locking monitor & set the quick key to lock the application
SessionLock.StartSessionMonitoring()
SessionLock.SessionLockKey = Keys.F11
```

d. Specify Initial Usernames and Passwords - Optional - Custom users for administration and security maintenance may be created using the code below.

Note: These users will *not* appear to end-users within the Security Maintenance and cannot be configured outside of the below code assignments. As such, dynamic usernames and/or passwords (such as the default administrator password which dynamically includes the current date) are recommended.

Custom Administrative Usernames and Passwords (Visual Basic)

```
'-- Set the administrative and security maintenance usernames and passwords
SecurityBasics.AdministratorUserName = "Administrator"
SecurityBasics.AdministratorPassword = "admin" & DateTime.Now.Day.ToString() '-- set the admin password so that it changes from day to day
SecurityBasics.AdministratorUserPk = -1

SecurityBasics.SecurityMaintenanceUserName = "SecurityUser"
SecurityBasics.SecurityMaintenancePassword = "mySecurityUserPass1"
SecurityBasics.SecurityMaintenanceUserPk = -2
```

e. Specify Default Security Settings - Optional - The default permission action, blocked message (using plain text or a message key), replacement character, and replacement regex can be specified using the following assignments.

Security Settings (Visual Basic)

```
'-- Set the default actions for security enabled objects within the application
SecurityBasics.DefaultPermissionInfo = New PermissionInfo(PermissionAction.Deny, _
    "Access Denied.", DeniedActions.Message)
SecurityBasics.DefaultPermissionAction = PermissionAction.Deny
SecurityBasics.DefaultBlockedMsg = "Access Denied."
'SecurityBasics.DefaultBlockedMsgKey = "AccessDeniedKey"
SecurityBasics.BlockedReplacementCharacter = "*"c
SecurityBasics.BlockedReplacementRegex = "[A-Za-z0-9@]"
```

f. Allow or Deny Windows Authentication - Optional - Windows authentication may be explicitly denied using the follwoing line of code. To allow windows authentication within the application, simply change the below **False** to a **True**.

Windows Authentication (Visual Basic)

```
'-- Determine whether to allow Windows authentication
SecurityBasics.AllowWindowsAuth = False
```


Security Documentation

Send comments on this topic.

Creating a New Application with Security

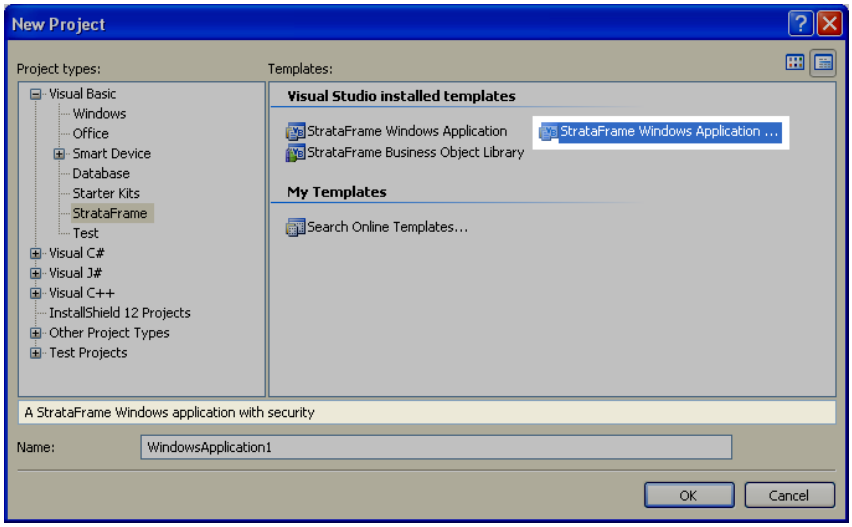
Purpose: Provide a step-by-step procedure on creating a new application with security.

Creating a New Application with Security

For a new applications, inclusion of security is greatly streamlined by using the `StrataFrame Windows Application w/ Security` template. To create a new application using this template:

1. Create New Project - Create a new Visual Studio Project using the StrataFrame template.
   a. Within Visual Studio, select `New->Project` under the `File` menu. The New Project dialog will display.
   b. Select the `StrataFrame` option under the Visual Basic or Visual C# item to see the available StrataFrame project templates.
   c. Select the `StrataFrame Windows Application w/ Security` template and click the OK button.

   Note: The template name may be abbreviated by the Visual Studio dialog. The Application w/ Security template may still be identified by clicking on each template and viewing the description.



2. Custom Login Form - If a custom login form is desired, it may be created as a new `SF Login Form` item and launched within the AppMain.vb or program.cs file. For more information, refer to the Creating a Custom Login Form help topic.
3. Recommended Customizations - For security reasons, it is highly recommended that the following options be changed from the StrataFrame defaults within the AppMain.vb or program.cs file:
   a. Set Data Source Key - Set the Security Data Source Key within the `SetDataSources()` method of AppMain.vb or program.cs. For a detailed description on the data source key, including what it does and how to set it, refer to the Setting the Security Data Source Key help topic.

   Security Data Source Key (Visual Basic)
   ```
   '-- Set the data source key for the security tables
   SecurityBasics.SecurityDataSourceKey = ""
   ```
   b. Change Encryption Key - Change the encryption key to be used for user data using the following method call within the `InitApplication()` method. For more information, refer to the Specifying the Encryption Key for User Data help topic.

   Encryption Key (Visual Basic)
   ```
   '-- Set the encryption key and vector for the user data
   SecurityBasics.SetSecurityKeyAndVectorForUserAuthentication("MySecurityKey")
   ```
   c. Change Initial Usernames and Passwords - Custom users for administration and security maintenance are defined within the below code, found in the `InitApplication()` method. For security reasons, a custom dynamic password is recommended for each application.

   Note: These users will *not* appear to end-users within the Security Maintenance and cannot be configured outside of the below code assignments.

   Custom Administrative Usernames and Passwords (Visual Basic)
   ```
   '-- Set the administrative and security maintenance usernames and passwords
   SecurityBasics.AdministratorUserName = "Administrator"
   SecurityBasics.AdministratorPassword = "admin" & DateTime.Now.Day.ToString() '-- set the admin password so that it changes from day to day
   SecurityBasics.AdministratorUserPk = -1

   SecurityBasics.SecurityMaintenanceUserName = "SecurityUser"
   SecurityBasics.SecurityMaintenancePassword = "mySecurityUserPass1"
   SecurityBasics.SecurityMaintenanceUserPk = -2
   ```
4. Optional Customizations - The following configurations are optional, and may be changed from the listed defaults as desired to suit the needs of the application:
   a. Configure Session Locking - If session locking will not be used within the application, the following lines of code may be removed or commented out within the `InitApplication()` method. If session locking will be used, they SessionLockKey may be changed. This is only required if dynamic session locking will be used within your application. For more information, refer to the Initializing Session Locking help topic.

   Session Locking (Visual Basic)
   ```
   '-- Start the session locking monitor & set the quick key to lock the application
   SessionLock.StartSessionMonitoring()
   SessionLock.SessionLockKey = Keys.F11
   ```
   b. Specify Default Security Settings - The default permission action, blocked message (using plain text or a message key), replacement character, and replacement regex are specified using the following assignments within the `InitApplication()` method. These may be changed as desired.

   Security Settings (Visual Basic)
   ```
   '-- Set the default actions for security enabled objects within the application
   SecurityBasics.DefaultPermissionInfo = New PermissionInfo(PermissionAction.Deny, _
       "Access Denied.", DeniedActions.Message)
   SecurityBasics.DefaultPermissionAction = PermissionAction.Deny
   SecurityBasics.DefaultBlockedMsg = "Access Denied."
   'SecurityBasics.DefaultBlockedMsgKey = "AccessDeniedKey"
   SecurityBasics.BlockedReplacementCharacter = "*"c
   SecurityBasics.BlockedReplacementRegex = "[A-Za-z0-9@]"
   ```
   c. Allow Windows Authentication - By default, Windows authentication is disabled. Windows authentication may be allowed by changing the follwoing assignment to **True** within the `InitApplication()` method.

   Windows Authentication (Visual Basic)
   ```
   '-- Determine whether to allow Windows authentication
   SecurityBasics.AllowWindowsAuth = False
   ```

Purpose: Provide a step-by-step procedure on creating and implimenting a custom login form within an application.

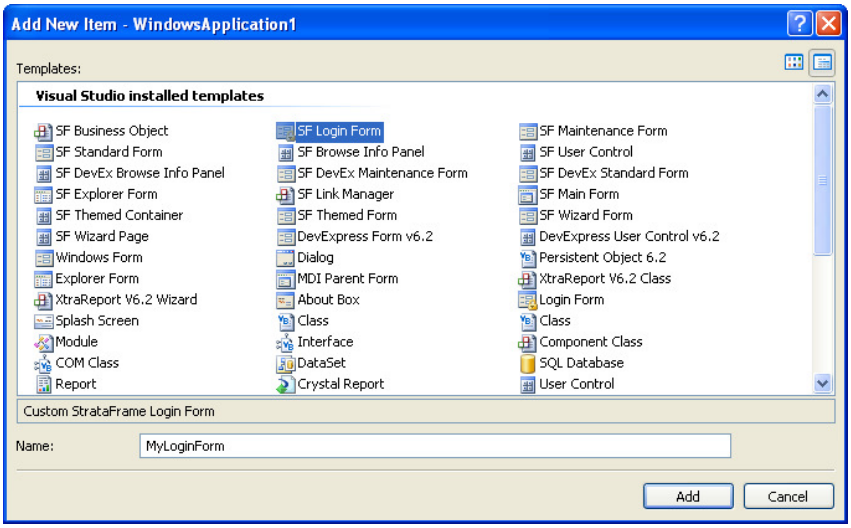
Creating and Implimenting a Custom Login Form

By default, the StrataFrame login form will be used to obtain access to a secured StrataFrame application. In order to change this login form, a custom login form must first be created. That login form must then be called within the AppMain.vb or program.cs file.

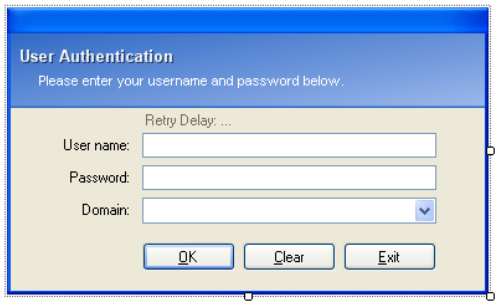
Creating the Custom Form

to create a custom login form:

1. Open the desired project within Visual Studio.
2. Right-click within the Solution Explorer and select `Add->New Item...`
3. Select `SF Login Form` from the available options, change its Name as desired, and click `Add`.



4. A form resembling the one pictured below will be created. Its look, feel, and functionality may be modified as desired to suit the needs of the application.



Calling the Login Form within AppMain.vb or program.cs

Once the custom login form has been created, it may be called within AppMain using the following code within the `ShowLoginAndInitMainForm()` method, where `MyLoginForm` is the name of your custom login form:

Calling a Custom Login Form (Visual Basic)

```
'-- Set the login form to your custom login form
Login.LoginFormType = GetType(MyLoginForm)
```

Note: If the `StrataFrame Windows Application w/ Security` template was used, this code will already be present, but will be commented out. To impliment the custom form, simply uncomment the line, and change 'MyLoginForm' to be the name of your custom login form.

Calling the Security Maintenance Dialog

Purpose: The purpose of this document is to discuss how to show the Security Maintenance Dialog (SecurityDialog) within your application to allow the end-users of your application to maintain roles and users.

The SecurityDialog

The same security maintenance dialog that is used within the design-time role-based security editor can be shown within your application to allow end-users to maintain users and roles within their database. This form is the `MicroFour.StrataFrame.Security.SecurityDialog` form. There are several overloads to the constructor of this dialog, but only 2 of them can be used at runtime.

The `SecurityDialog` behaves differently at runtime than it does at design-time. Since permissions are defined by the developer and must be tied into the application, it is not logical for end-users to be able to define new permissions or alter existing ones. However, end-users should be able to create or alter users, roles, and optionally restriction sets. Therefore, while permissions can be shown within the `SecurityDialog` at runtime, they cannot be edited.

The following samples can be used to show the `SecurityDialog` within your application. The default constructor for the `SecurityDialog` class will allow both the permissions and the restriction sets to be visible within the dialog. The second overload accepts two Boolean arguments: the first determines whether the permissions will be visible, the second determines whether the restriction sets will be visible and editable.

Code Samples

Showing the Security Dialog

Sample - Showing the SecurityDialog [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Sub ShowSecurityDialog()
    '-- Create the form and show it
    Using loDialog As New SecurityDialog()
        loDialog.ShowDialog()
    End Using
End Sub
```

Sample - Showing the SecurityDialog[C#]

```
using MicroFour.StrataFrame.Security;
...
private void ShowSecurityDialog()
{
    //-- Create the form and show it
    using (SecurityDialog loDialog = new SecurityDialog())
    {
        loDialog.ShowDialog();
    }
}
```

Showing the SecurityDialog Without Permissions or Restriction Sets

Sample - Showing the SecurityDialog without Permissions or Restriction Sets [Visual Basic]

```
Imports MicroFour.StrataFrame.Security
...
Private Sub ShowSecurityDialog()
    '-- Create the form and show it
    Using loDialog As New SecurityDialog()
        loDialog.ShowDialog(False, False)
    End Using
End Sub
```

Sample - Showing the SecurityDialog without Permissions or Restriction Sets [C#]

```
using MicroFour.StrataFrame.Security;
...
private void ShowSecurityDialog()
{
    //-- Create the form and show it
    using (SecurityDialog loDialog = new SecurityDialog())
    {
        loDialog.ShowDialog(false, false);
    }
}
```


Security Documentation

Send comments on this topic.

Deploying Security Data

Purpose: This topic gives a basic step-by-step on deploying security data using the Database Deployment Toolkit, and gives an account of all required information for those wishing to deploy security data without using the Database Deployment Toolkit.

Deploying Security Data

Whenever a secured application is deployed to the end-user, a number of required tables must exist within the specified database with the structure specified below. These tables should be deployed with all security data pertaining to the desired security project. If the Database Deployment Toolkit is being used, both the deployment data packages and the required table structures may be created automatically via the Deployment Package Wizard.

Deploying Security Data Using the DDT

It is *highly* recommended that the Database Deployment Toolkit be used to generate both the structure and deployment data for the security system. Not only does the available Deployment Package Wizard greatly streamline the deployment process, it also ensures accuracy and completeness for the structure and data.

To generate the required structure and deployment packages using the Database Deployment Toolkit:

1. Open Visual Studio.
2. Select Database Deployment Toolkit under the StrataFrame menu.
3. Select the desired DDT Profile and click the Open button.
4. Right-click on the Deployment Data item in the left panel and select Deployment Package Wizard from the context menu.
5. Choose Role Based Security Data as the Deployment Data Type.
6. When prompted, select the desired source database, destination database, and security project.
7. Once the wizard is complete, both the security table structures and deployment data will be generated and ready for distribution.

Note: A more in-depth step-by-step for the above process is available in the Deployment Package Wizard topic within the StrataFrame DDT help file.

Deploying Security Data Without the DDT

Even if the Database Deployment Toolkit is not being used, the required table structures and deployment data will still need to be created on the end-user system.

For convenience, the required table structures and relationships are listed below. Again, if using the Database Deployment Toolkit, both the below structures and all deployment data present in the development databases may be packaged automatically using the Deployment Package Wizard.

Table Structures SQL Script

The SecurityTables.sql script is a SQL Server 2005 script that can be used to create the StrataFrame security tables within a database. You can run this script independently, or you may incorporate it into your application's own database creation script.

To run the script:

1. Download the Script - Click here to download the script. Once downloaded, extract the .sql file from the Zip.
2. Replace Database Name - Before the script may be run, a single line must be changed. The very first non commented line in the script reads "USE [DatabaseName]". Replace "DatabaseName" with the name of the desired database.
3. Run the Script - The script is now ready to be run.

Note: This script is designed for SQL Server 2005 only, and must be modified before it can be used on SQL Server 2000.

Explicit Table Definitions

The explicit table definitions for all security tables along with their required relationships are listed below.

Preferences - SFSPreferences

SFSPreferences

Column Name	Data Type	Length	Description
sp_pk	int	4	Primary Key
sp_sproj_pk	int	4	Foreign key to security project
sp_WaitTimeAfterLoginFailure	bigint	8	Time delay on password entry after invalid attempt
sp_MaxInvalidLoginAttempts	int	4	Maximum login attempts before making user inactive
sp_MaxInvalidLoginAttemptsTime	bigint	8	Amount of time to expire during invalid login attempts
sp_PwMinAge	bigint	8	Minimum Password Age
sp_PwMaxAge	bigint	8	Maximum Password Age
sp_PwMaxLength	int	4	Maximum length of the password
sp_PwMinLength	int	4	Minimum length of the password
sp_PwAreComplex	bit	1	Enforces password complexity
sp_PwBeforeRepeat	int	4	Number of unique passwords before allowing a repeat of a particular password
sp_SessionTimeout	bigint	8	Session Inactivity Timeout
sp_AllowWindowsAuth	bit	1	Allows windows authentication
sp_Version	int	4	Concurrency row versioning

Permissions - SFSPermissions

SFSPermissions

Column Name	Data Type	Length	Description
pm_pk	int	4	Primary Key
pm_sproj_pk	int	4	Security project primary key
pm_Key	nvarch...	50	Application Object Key
pm_Description	nvarch...	-1	Permission Description
pm_Category	nvarch...	50	Permission category
pm_BlockedAction	int	4	The action to take when this permission is denied
pm_BlockedMsgOrKey	nvarch...	1000	The message or localization key for the blocked message
pm_AlwaysAuditApp	bit	1	Always audit application events
pm_AlwaysAuditData	bit	1	Always audit data events
pm_Version	int	4	Row version
pm_CreatedBy	int	4	User
pm_CreatedAt	datetime	8	Created Timestamp
pm_DontAllowReadOnly	bit	1	Allow ReadOnly for this Permission

Roles - SFSRoles

SFSRoles

Column Name	Data Type	Length	Description
rl_pk	int	4	Primary Key
rl_sproj_pk	int	4	Security project primary key
rl_Role	nvarch...	255	Security Role
rl_Description	nvarch...	-1	Description of Security Role
rl_Version	int	4	Row versioning (Concurrency)
rl_AlwaysAuditApp	bit	1	Always audit application events
rl_AlwaysAuditData	bit	1	Always audit data events
rl_CreatedBy	int	4	User Id creator
rl_CreatedAt	datetime	8	TimeStamp

Users - SFSUsers

SFSUsers

Column Name	Data Type	Length	Description
us_pk	int	4	Primary Key
us_sproj_pk	int	4	Foreign key to security project
us_Username	nvarch...	25	User Identification for logon
us_Data	nvarch...	-1	Data containing user password, etc.
us_FirstName	nvarch...	50	First Name
us_MiddleName	nvarch...	50	Middle Name
us_LastName	nvarch...	50	Last Name
us_Version	int	4	Concurrency Versioning
us_CreatedBy	int	4	User Id that created record
us_CreatedAt	datetime	8	Time Stamp

Restrictions - SFSRestrictions

SFSRestrictions

Column Name	Data Type	Length	Description
rs_pk	int	4	Primary Key
rs_sproj_pk	int	4	Foreign key to security projects
rs_Name	nvarch...	255	Restriction name
rs_Description	nvarch...	-1	Description
rs_CreatedBy	int	4	Created by user
rs_CreatedAt	datetime	8	Created timestamp
rs_Version	int	4	Row version

Restriction Items - SFSRestrictionItems

SFSRestrictionItems

Column Name	Data Type	Length	Description
ri_pk	int	4	Primary key
ri_rs_pk	int	4	Foreign key to SFSRestrictions
ri_Action	int	4	Action to take during this restriction item
ri_DaysOfWeek	int	4	Enumeration containing days of the week
ri_StartTime	int	4	Minutes since midnight
ri_EndTime	int	4	Minutes since midnight
ri_Workstation	nvarch...	255	Workstation
ri_CreatedBy	int	4	Create by user
ri_CreatedAt	datetime	8	Created timestamp
ri_Version	int	4	Row version

Roles to Permissions Link Table - SFSRolesXPermissions

SFSRolesXPermissions

Column Name	Data Type	Length	Description
rp_pk	int	4	Primary key
rp_rl_pk	int	4	Primary Key Roles
rp_pm_pk	int	4	Primary Key Permissions
rp_Action	int	4	Type of Context Action (Default)
rp_AlwaysAuditApp	bit	1	Always audit application events
rp_AlwaysAuditData	bit	1	Always audit data events
rp_CreatedAt	datetime	8	Created Timestamp
rp_CreatedBy	int	4	User Id of Creator
rp_Restriction	int	4	Foreign key to Restriction

Users to Permissions Link Table - SFSUsersXPermissions

SFSUsersXPermissions

Column Name	Data Type	Length	Description
up_pk	int	4	Primary Key
up_us_pk	int	4	Users Primary Key
up_pm_pk	int	4	Primary Key Permissions
up_Action	int	4	Type of Context Action (Default)

Users to Roles Link Table - SFSUsersXRoles

SFSUsersXRoles

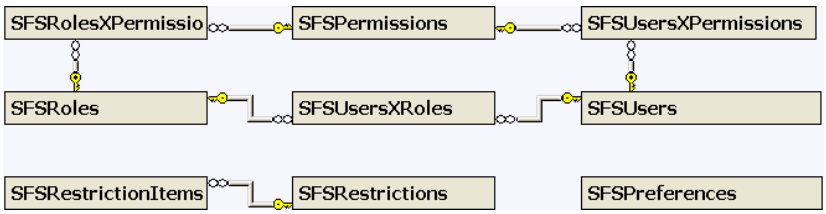
Column Name	Data Type	Length	Description
ur_pk	int	4	Primary Key
ur_us_pk	int	4	Primary Key of Users
ur_rl_pk	int	4	Primary Key of Roles
ur_CreatedBy	int	4	User Id of Creator
ur_CreatedAt	datetime	8	Created Timestamp

Table Relationships

The above listed tables must exist with the following relationships:

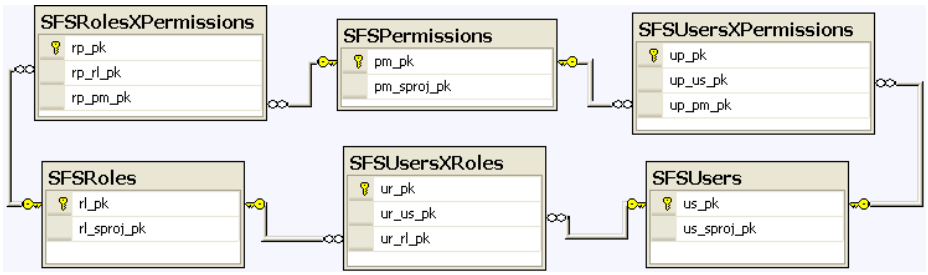
Security Relationships

The following diagram shows the basic outline of all table relationships within security:



Permission, Roles, and Users Diagram

The following diagram shows a more detailed breakdown of the relationships concerning all Permission, Role, and User tables:



Workstation Restrictions Diagram

The following diagram shows a more detailed breakdown of the relationship between the SFSRestrictions and SFSRestrictionItems tables: