IWS FDA 21 CFR Part 11 Features

Introduction

The 21 CFR Part 11 regulations from the Food and Drug Administration (FDA) sets forth the criteria under which the agency considers electronic records and electronic signatures to be trustworthy, reliable, and generally equivalent to paper records and handwritten signatures executed on paper.

This document describes enhancements and other features that allow users to easily configure applications in conformance with the 21 CFR Part 11 regulation, using InduSoft Web Studio™ (IWS) v5.1 SP3 or higher as an engineering and runtime tool for Windows NT/2000/XP, and using InduSoft CEView as a runtime program for Windows CE.

Comments

General:

- The software (SCADA) cannot state that it complies with FDA Part 11. The software shall provide the necessary tools to allow a user to create a system (application) that is compliant with FDA Part 11.
- The SCADA system should not "force" the user to build an application that is compliant with FDA Part 11. FDA Part 11 compliance is optional during application development.
- An *Electronic Record* is any data that can be saved as electronic media and retrieved later.
  An *Electronic Signature* is a specific type of Electronic Record that contains the following information:
  - Timestamp
  - User name
  - Meaning of the signature

  A *Digital Signature* is a specific type of Electronic Signature, in which the data is encrypted.
- An *Open System* (such as the World Wide Web or Web) requires encryption for electronic reports and for the Electronic Signature (Digital Signature).
- Electronic records are associated with events (such as tag changes, load recipes, and so forth), whether the user triggered the event or not. Electronic signatures are associated with actions triggered by the user (such as pressing a button, changing a slider, entering a set-point manually, and so forth).

Electronic Records (Event Logger, Alarms, Reports)

- The Part 11 rule does not mention whether the electronic records must be stored in a standard database (such as Oracle, SQL Server, etc.) or in a proprietary format. When you use a standard database, the responsibility for guaranteeing the confidentiality of the database relies on the database itself (such as password protected databases). IWS v6.0 + SP3 or higher has direct interfaces to databases for Alarms, Events, Trend and Grid objects through ADO, OLE DB or ODBC.

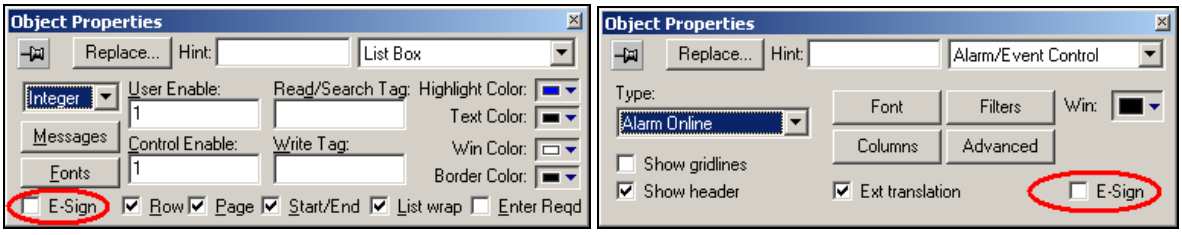
Electronic Signatures (Security System)

- The system administrator must be able to access the user account settings to create new accounts, lockout users, and de-authorize them. These changes must be logged, even if the runtime is not running.
- Nobody (not even the System Administrator) can have access to the password of any user.

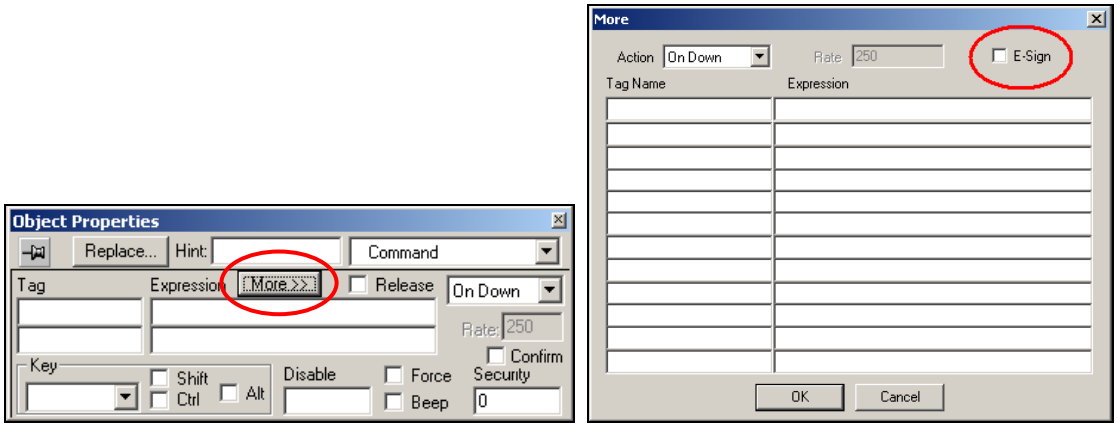
Electronic Signatures

This section describes the **E-sign** check-box, which can be found in all objects that enable users' runtime actions. This box requires the user to enter an electronic signature to use certain objects in an application.

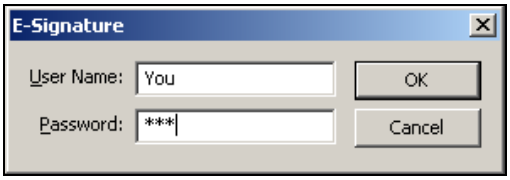
The **E-sign** check-box is a field on the *Object Properties* dialog (as shown in the following figures):



Also, for objects using the **Command** property, the **E-sign** check-box was added to the *More* dialog box:

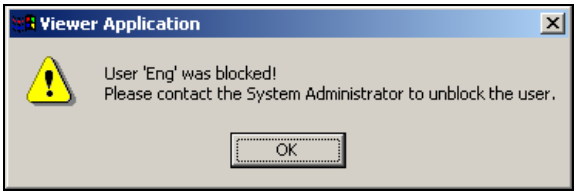


If you enable (check/click) the **E-Sign** box, users must enter their Electronic Signature before IWS can execute the command. An *E-Signature* dialog opens, and the user enters a **User Name** and **Password** into it:



If the user enters an invalid signature, the following message displays and IWS records this event in an Event Log file. For more information about the Event Log file, see the “Event Logging and Retrieval” section in this document.

You can specify how many attempts a user has to enter their electronic signature correctly. When the user enters an invalid signature, the information that displays in response will contain the number of attempts the user has left in which to enter the correct signature. After the last attempt, the account is blocked:



Instructions for unlocking an account are provided in the next section.

Security System Settings Related to FDA 21 CFR Part 11

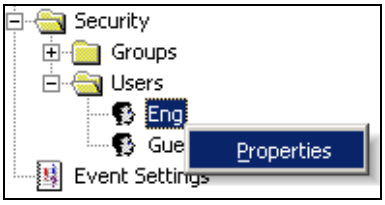
This section describes the following security system enhancements, based on the FDA requirements.

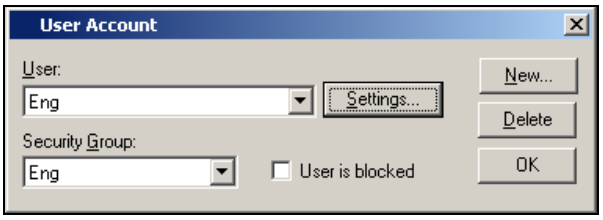
Settings Button

You can use the **User is blocked** check box and the **Settings** button to control a user's access to the application. You can access these features from the *User Account* dialog.

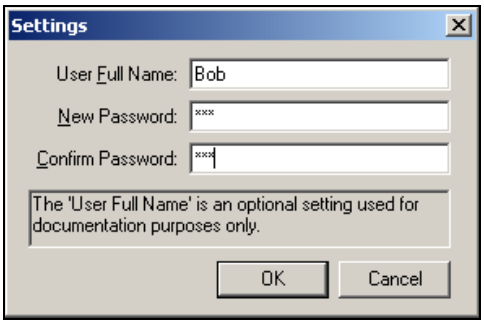
Use the following steps to open this dialog and configure user access:

1. In the *Workspace*, expand the *Security* folder, and right-click on a user name.
2. When the pop-up menu displays, select **Properties** to open the *User Account* dialog.





3. If necessary, click **User is blocked** check box to block the selected user.
4. Click the **Settings** button to open the *Settings* dialog box.



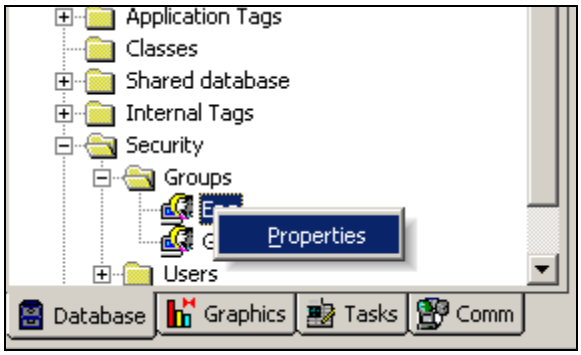
5. Configure the parameters on this dialog as follows
   - **User Full Name** (*optional*): Type the full name of the user.
   - **New Password**: Type the user's password.
   - **Confirm Password**: Re-type the same password.
6. When you are finished, click **OK** to apply the changes and close the *Settings* dialog.

Advanced Button

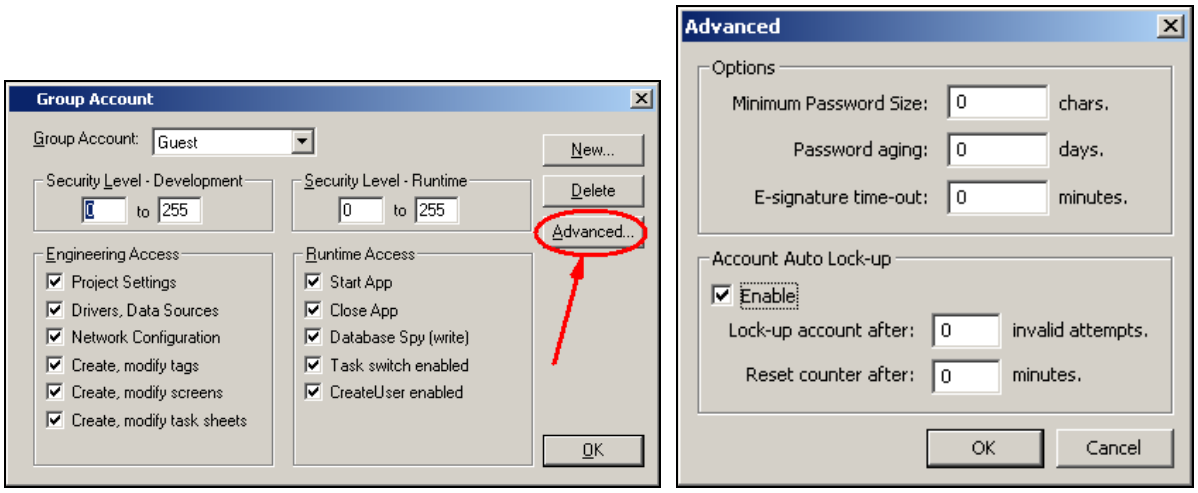
You can use the **Advanced** button to control a user's access to the application. You can access these features from the *Group Account* dialog.

Use the following steps to open this dialog and configure user access:

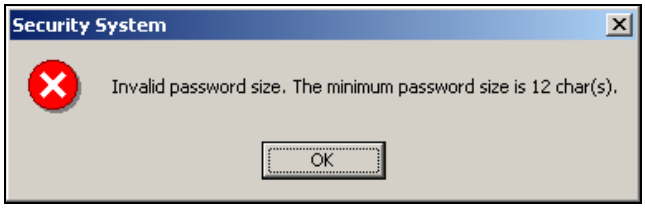
1. In the *Workspace*, expand the *Security* and *Groups* folders, and then right-click on a group name.
2. When the pop-up menu displays, select **Properties** to open the *Group Account* dialog.



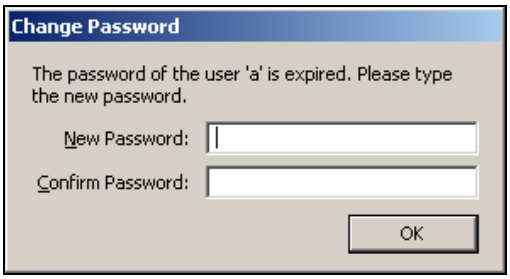
3. Click the **Advanced** button, which is located just below the **New** and **Delete** buttons, to open the *Advanced* dialog.



4. Configure the parameters on the *Advanced* dialog as follows:
   - **Minimum Password Size** text box: Type a value greater than **0** into this field to require a minimum number of characters for a password. All users assigned to this group must provide a password containing at least the minimum number of characters. If a user tries to create a password with fewer than the required number of characters, IWS will reject the password and display the following warning:



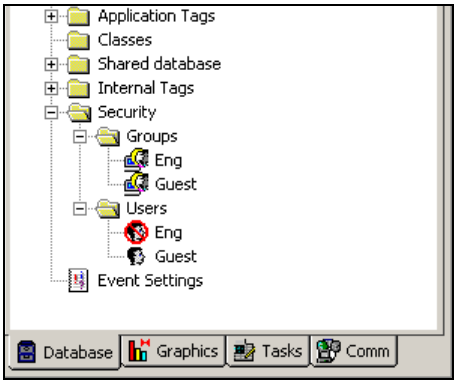
- **Password aging** text box: Type a value greater than **0** into this field to establish the longevity (in days) of a password. After the specified number of days, IWS will force users assigned to this group to change their passwords. When the user tries to log in, the *Change Password* dialog box will display (see the following figure) automatically, and the user will not be able to complete the log-in process until they provide a new password.

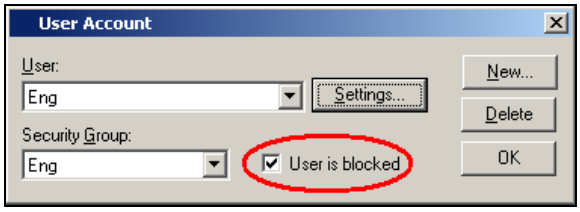


- **E-signature time-out** text box: Type a value greater than **0** into this field to specify a time-out period (in minutes). Users assigned to this group must enter their **User Name** and **Password** before the specified time-out period expires to execute commands requiring an electronic signature. Before the time-out expires, the user is asked for the **Password** only – the system automatically assumes the **User Name** logged in with the last electronic signature. The system resets the time-out counter just after an electronic signature is executed.
- **Enable** check box: Enable (check/click) this box to activate the following Account Lock-up features.
- **Lock-up account after** text box: Type a value into this field to define the maximum number of times a user can try to log on to an account. If the user attempts to log on with an invalid password the specified number of times, IWS will lock the user account.
- **Reset counter after** text box: Type a value into this field to define how long after an invalid log-on attempt IWS will wait (in minutes) before it resets the log-on attempts counter.

> **Note:**
> When a user exceeds the specified number of log-on attempts, IWS automatically blocks the account and will not reset the counter — even after the **Reset counter after** time expires. The System Administrator must reset the user account by disabling (unchecking) the **User is blocked** check box on the *User Account* dialog box, or by executing the **`UnblockUser()`** function. See the “Security System Functions” section in this document.

A red circle with a slash mark over a user name in the Workspace indicates that the user is blocked. In addition, the **User is blocked box** will be enabled (checked). For example, the following figures indicate that the *Eng* user is blocked.

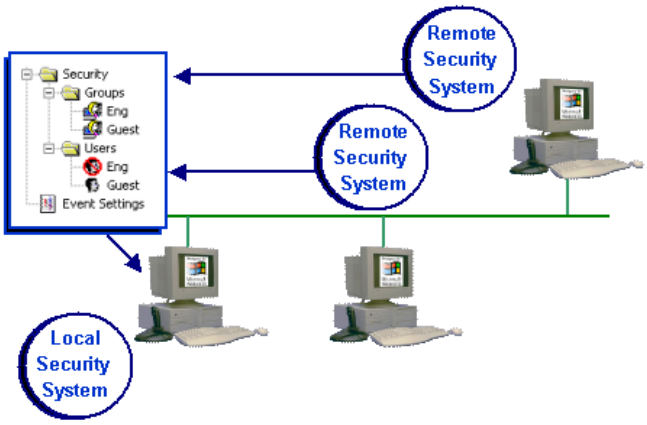




5. When you are finished, click **OK** to apply your changes and close the *Advanced* dialog box.

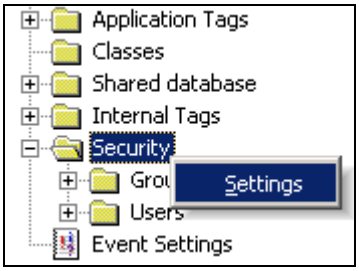
Remote Security System

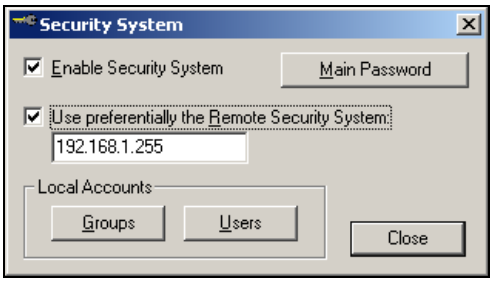
If your system applications connect through a TCP/IP link, it is now possible to designate one of your computer stations as the *Central* security system, from which other stations can use the Users and Groups definitions. The following figure illustrates this configuration:



Use the following procedure to configure a central security system:

1. Right-click the *Security* folder on the Database tab, and select **Settings** from the pop-up menu. The *Security System* dialog will open.





2. Enable (check/click) the **Use preferentially the Remote Security System** check box to designate a remote security system.
   - If the remote applications successfully connect to the security system from the *Server* station, they will use the security system configured on the *Server* station. In this case, any change implemented in the security system of the *Server* station will be assumed automatically by the remote applications. Also, the security system functions (such as **`CreateUser()`**, **`RemoveUser()`**, **`ChangePassword()`**, and so forth) will update the *Server* station's security system — even if the functions are executed from the remote applications. As a result, all applications on a distributed system can share the same security system settings.
   - If the applications cannot connect because the remote system is not running or cannot be reached, a message (similar to the following) will be logged in the *Output* window and saved in the *event* file:

     `Error connecting to Remote Security Server '192.168.1.255'`

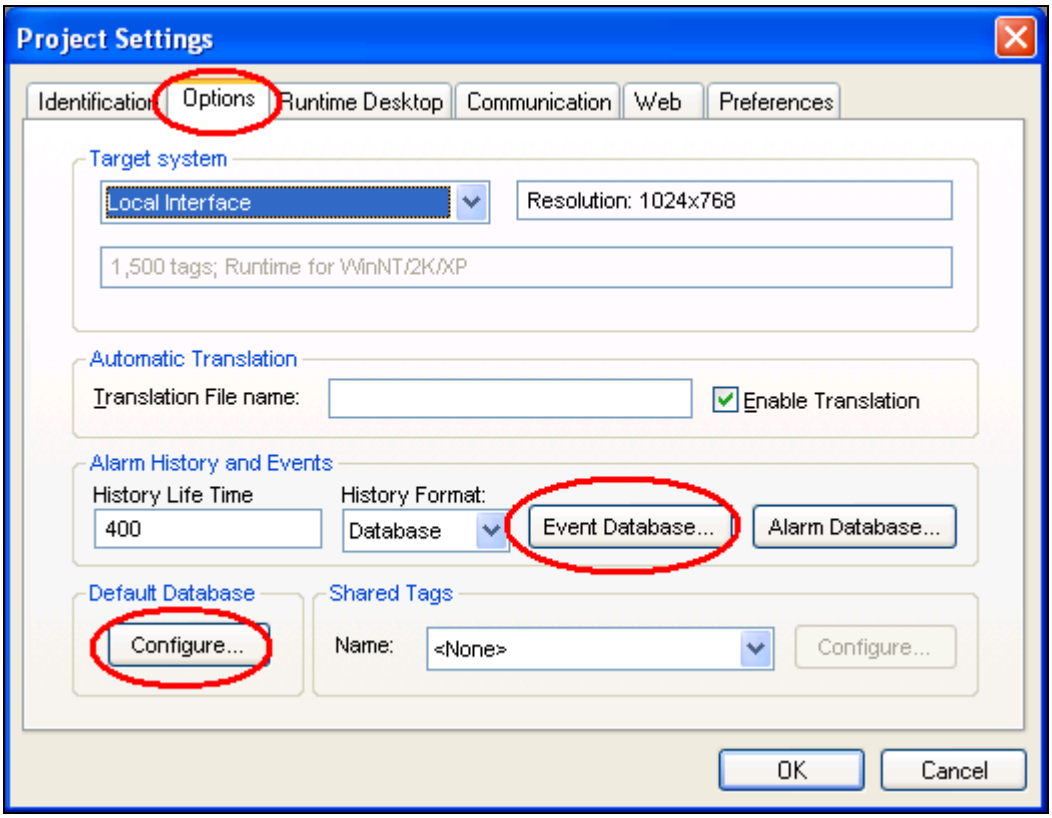
     In addition, the application(s) will revert to using the local computer's security settings. The remote applications will attempt to connect to the *Server* station's security system only when there is an event associated with the security system (such as a user logging on). In other words, there is no polling between the remote applications and the *Server* station during runtime.

Event Logging and Retrieval

This section describes IWS's event logging and retrieval features. An event can be any tag change, which can generate reports or recipes, open and close screens, log onto and log off the security system, etc. IWS saves all of these events in a log file, which can be retrieved by the Alarm/Event Control object.

The user can select to save the Events either in the InduSoft proprietary format or in a SQL Relational Database, such as Oracle, SQL Server, etc. Use the following procedure:

1. Click on the *Project* menu, then click *Settings*.
2. Select the *Options* tab.



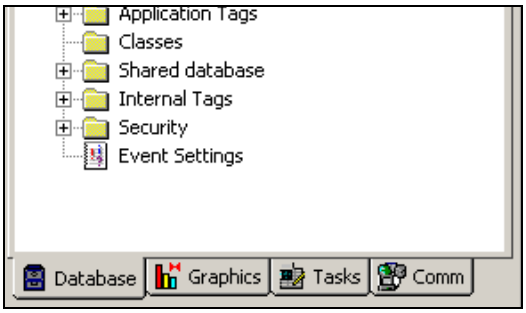
3. In the *Alarm History and Events* frame, select a **History Format** (either **Proprietary** or **Database**). If you select **Database** the **Event Database** and **Alarm Database** buttons will activate, allowing you to configure the applicable settings.
4. In the *Default Database* frame, click the **Configure** button to configure one database into which both Events and Alarms will be logged.
5. If you select **Proprietary** in the **History Format** field, Event log files are stored in the application's **`\Alarm`** folder (the same folder where IWS saves alarm historical files). The event log file names must conform to the **`evYYMMDD.evt`** format, where:
   - **`YY`** represents the last two digits of the year in which the event log file was generated
   - **`MM`** represents the month in which the event log file was generated
   - **`DD`** represents the day on which the event log file was generated

For example, a log file for June 7, 2005 would be **`ev050607.evt`**.

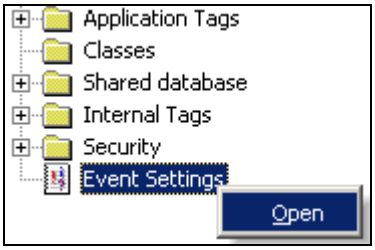
Configuring the Events Settings

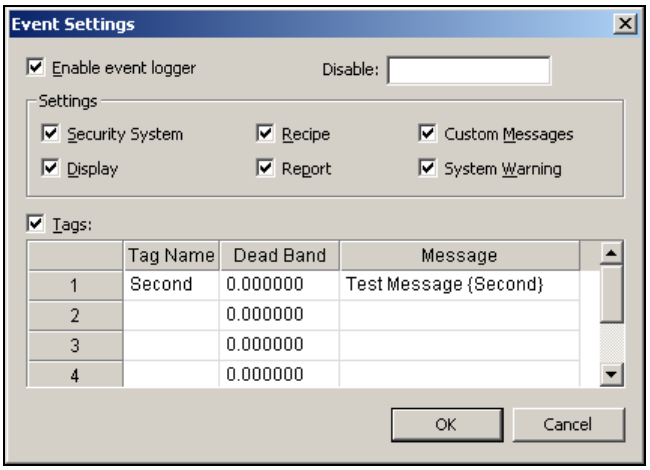
Use the following instructions to configure the Event retrieval feature.

1. Select the *Database* tab. This tab contains an icon called **Event Settings**, as shown in the following figure.



2. Right-click the **Event Settings** icon, and select **Open** from the pop-up to open the *Event Settings* dialog.





3. Configure the parameters on the *Event Settings* dialog as follows:
   - **Enable event logger** check-box: Enable (check/click) this box to enable Event logging.
   - **Disable** text box: If you enter a tag value other than **0** (*false*), IWS will automatically disable the Event Logger.
   - **Security System** check box: Enable (check/click) this box to include security system events in the historic event file. IWS logs the following security system events:
     - Log On / Log Off users
     - User created/removed using the **`CreateUser()`** or **`RemoveUser()`** functions
     - User blocked/unblocked using the **`BlockUser()`** or **`UnblockUser()`** functions
     - User blocked by the security system after several attempts to enter an invalid password
     - Password expired
     - Password modified
     - Invalid Log On attempt
   - **Display** check box: Enable (check/click) this box to include screen Open and Close events in the historical event file.
   - **Recipe** check box: Enable (check/click) this box to include Recipe Load, Save, Init, and Delete events in the historical event file.
   - **Report** check box: Enable (check/click) this box to include Reports Saved to Disk or Send to Printer events in the historical event file.
   - **Custom Messages** check box: Enable (check/click) this box to include events generated by the **`SendEvent( strEvent )`** function in the historical event file.
   - **System Warning** check box: Enable (check/click) this box to include general system warnings (such as `Division by zero`, `Attempted to access invalid array index`, and so forth) in the historical event file. IWS logs the following system warning events:
     - Errors that occur when sending alarms by email
     - Tag was blocked/unblocked
     - Division by zero
     - Connection/Disconnection of the remote security system
   - **Tags** check box: Enable (check/click) this box to enable and log tag changes in the historical event file. Configure the tags you want to log in the Tags table as follows:
     - **Tag Name** column: Type the name of the tag you want to log in the event file.
     - **Dead Band** column: Type a value to filter acceptable changes against.

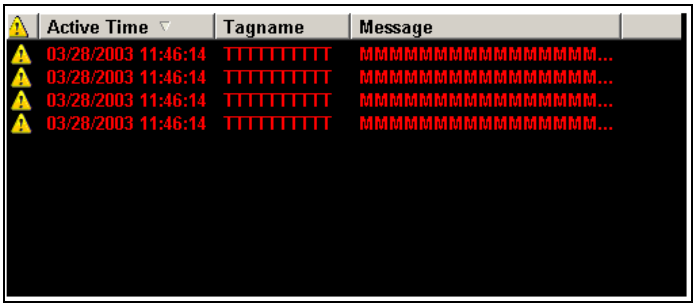
       For example, if you specify a **Dead Band** value = 5 for a tag value = 50 and the tag value changes to 52, the system will not register this variation in the event log file, because the variation is less than 5. However, if the tag value change is equal to or greater than 5, the system will save the new value to the history file.
     - **Message** column: Type a string (message) related to this tag change. You can specify tags in messages using the **`{tag name}`** syntax.

The **Tags** parameter can be useful if you want to generate a log file of events that are not necessarily alarm conditions (for example, **`Motor On`**, **`Motor Off`**, etc.).

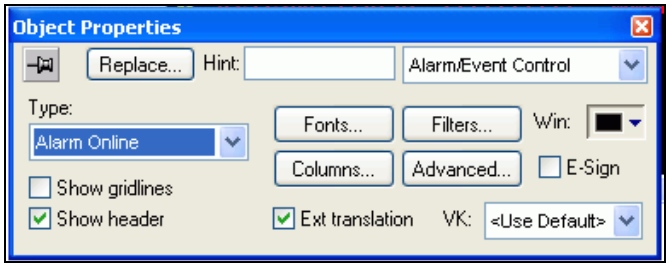
Viewing Logged Events

You can use the **Alarm/Event Control** screen object to view a list of all logged events. Use the following instructions to configure this object:

1. Click the **Alarm/Event Control** icon to add an alarm/event control object to your application screen.



2. Double-click on the object to open an *Object Properties* dialog.



3. Click the **Type** combo box drop-down arrow, and select **Event** or **Alarm History + Event** from the list.

   You use most of the same parameters to configure Event objects and Alarm History objects. The only difference is that the **Selection**, **Group**, and **Priority** filters are disabled for the Event objects.

> **Note:**
> If you are not familiar with the Alarm/Object Control screen object, please read the *Using the Active Objects Toolbar* section in Chapter 7 of the *InduSoft Web Studio Users Guide and Technical Reference Manual*.

IWS Functions Related to FDA 21 CFR Part 11

This section describes the IWS functions associated with the features discussed previously.

Security System Functions

This section describes the following IWS Security functions:

- `BlockUser(strUserName)`
- `CreateUser(strUserName, strGroupName, strPassword)`
- `GetUserNames (strUsers , nUserType , strGroups)`
- `GetUserState(strUserName)`
- `RemoveUser(strUserName)`
- `SetPassword(strUserName, strOptionalNewPassword)`
- `UnblockUser(strUserName)`

`BlockUser(strUserName)`

Group	Security
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Used to block an existing user from the security system.
- **Parameters**:

`StrUserName`	String tag containing the name of the user to block.

- **Returned Values**:

0	User blocked successfully.
1	Invalid number of parameters
2	Wrong parameter type
3	User does not exist.
4	It is not possible to write the data successfully.

- **Examples**:

Tag Name	Expression
Tag	BlockUser(“Bob”)
Tag	BlockUser(“Albert”)

Note:
You cannot use this function to create a user name that is already being used in the application. Users created with this function do not display in the development environment *Security* folder because they are stored in a secondary database.

Tip:
You can use the **`ExtUser.exe`** program (located in the **`Bin`** folder) to manage the users in this secondary database.

`CreateUser(strUserName, strGroupName, strPassword, strOptUserFullName)`

Group	Security
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Creates a new user.
- **Parameters:**

`StrUserName`	String tag containing the name of the user to be created
`StrGroupName`	String tag containing the name of the group to which the user will belong
`StrPassword`	String tag containing a password for the user
`StrOptUserFullName`	String tag containing the full name of the user. This parameter is *optional*.

- **Returned Values:**

0	New user created successfully.
1	Invalid number of parameters
2	Wrong parameter type
3	User already exists.
4	Group does not exist.
5	It is not possible to safely write the data.
6	It is not possible to use the **`CreateUser()`** function.

- **Examples:**

Tag Name	Expression
Tag	CreateUser(“Bob”, “Admin”, “Chocolate”)
Tag	CreateUser(“Albert”, “Engineering”, “EMC2”)

> **Note:**
> You cannot use this function to create a user name that is already being used in the application. Users created with this function do not display in the development environment *Security* folder because they are stored in a secondary database.

> **Tip:**
> You can use the **`ExtUser.exe`** program (located in the **`Bin`** folder) to manage the users in this secondary database.

GetUserNames ("tagUsers" , optnumUserType , "opttagGroups")

Group	Security
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Not Supported

- **Description:**
- **Parameters:**

"tagUsers"	Name of the array tag that will receive users
optnumUserType	0- Return all users 1- Only users created during runtime 2- Only users created using the development environment
"opttagGroups"	Name of the array tag that will receive the group for each specific user

- **Returned Values**: Number of users or a negative number that can be one of the following:

-1	Invalid number of parameters
-2	"tagUsers" is invalid.
-3	optnumUserType is invalid.
-4	opttagGroups is invalid.
-5	Error; function cannot be called in the Web Thin Client.

Natural number set: Number of users

- **Examples:**

Tag Name	Expression
NumberOfUsers	GetUserNames("UsersArray") //Retrieves all users, stores their names in the UsersArray tag and the number of users in the NumberOfUsers tag.
NumberOfUsers	GetUserNames("UsersArray", 1) //Retrieves all users created during runtime, stores their names in the UsersArray tag and the number of users in the NumberOfUsers tag.
NumberOfUsers	GetUserNames("UsersArray", 2) //Retrieves all users created in the development environment, stores their names in the UsersArray tag and the number of users in the NumberOfUsers tag.
NumberOfUsers	GetUserNames("UsersArray", 2, "Groups") //Retrieves all users created in the development environment, stores their names in the UsersArray tag and the number of users in the NumberOfUsers tag. The group name per each user is stored in the Groups tag.

GetUserState(strUserName)

Group	Security
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description**: Use to see the current status of a selected user.
- **Parameters**:

StrUserName	String tag containing the name of the user

- **Returned Values**:

0	User is unblocked.
1	User is blocked.
3	User does not exist.
4	It is not possible to safely write the data.

- **Examples**:

Tag Name	Expression
Tag	GetUserState(“Bob”)
Tag	GetUserState(“Albert”)

`RemoveUser(strUserName)`

Group	Security
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Removes a user from the system.
- **Parameters:**

`StrUserName`	String tag containing the name of the user to be removed

- **Returned Values:**

0	User removed successfully.
1	Invalid number of parameters
2	Wrong parameter type.
3	User does not exist.
4	It is not possible to safely write the data.

- **Examples:**

Tag Name	Expression
Tag	RemoveUser(“Bob”)
Tag	RemoveUser(“Albert”)

> **Note:**
> You can use this function to remove only those users created with the **`CreateUser()`** function.

`SetPassword(strUserName, strOptionalNewPassword)`

Group	Security
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Use to specify a new user password.
- **Parameters:**

`StrUserName`	String tag containing the name of the user
`StrOptionalNewPassword`	*Optional* string tag containing the new password

- **Returned Values:**

0	The new password has been set.
1	User is blocked.
3	User does not exist.
4	It is not possible to safely write the data.
5	Operation was cancelled.

- **Examples:**

Tag Name	Expression
Tag	SetPassword(“Bob”)
Tag	SetPassword(“Albert,” “anemarie”)

> **Note:**
> If you omit the **strOptionalNewPassword** parameter, the **`SetPassword()`** function will launch an *Enter a new password* dialog, so the user can configure a new password.

UnblockUser(strUserName)

Group	Security
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Use to unblock a blocked user in the system.
- **Parameters:**

StrUserName	String tag containing the name of the user to unblock.

- **Returned Values:**

0	User unblocked successfully.
1	Invalid number of parameters.
2	Wrong parameter type.
3	User does not exist.
4	It is not possible to safely write the data.

- **Examples:**

Tag Name	Expression
Tag	UnblockUser(“Bob”)
Tag	UnblockUser(“Albert”)

Generate Events Function
`SendEvent(strEvent, optBooFlag, optStrComment)`

Group	Event Logger
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description**: Use to send an event to the **Event Log** file.

  This function has an option that allows you to create a comment. When this option is enabled, the user is prompted to enter a comment after executing the `SendEvent()` function. This comment will be saved in the **Event Logger** file.

- **Parameters**:

`strEvent`	String value or tag containing the text to be saved in the Event Log file
`optBooFlag`	If omitted or **0** (zero), the event does not have a comment; otherwise, there is a comment associated with the event.
`optStrComment`	String value of tag containing the text of the comment for the event saved in the database. If omitted, the user is prompted with a standard dialog in which the comment can be typed.

- **Returned Values**:

0	Success
1	Event Logger is disabled in the *Event Settings* dialog.
2	Event Logger is enabled, but Custom Messages are disabled in the *Event Settings* dialog.

- **Examples**:

Tag Name	Expression
Tag	SendEvent("Valve Open") // Saves the event message.
Tag	SendEvent("Valve Open Oven No." + OvenID) // Saves the event message concatenated with the value of the **`OvenID`** tag.
Tag	SendEvent("Valve Open", 1) // Displays the dialog in which the operator can type comments.
Tag	SendEvent("Valve Open", 1, TagComment) // Saves the event message with the comment configured in the TagComment tag.

> **Caution:**
> This function is synchronous. Therefore, the execution of the function finishes only after the event data (including the comment, if any) is saved in the database file. It is recommended that you do not configure this function in background tasks (e.g. *Math* and *Scheduler*), unless you do not plan to use the comment or configure it directly (type from the dialog) in the function.

Database Functions

This section describes the IWS Database functions:

- SyncAlarm(strOptionalStartDate, strOptionalEndDate)
- SyncEvent(strOptionalStartDate, strOptionalEndDate)
- SyncTrend(numGroup, strOptionalStartDate, strOptionalEndDate)
- SyncAlarmStatus
- SyncEventStatus
- SyncTrendStatus(numGroup)

SyncAlarm(strOptionalStartDate, strOptionalEndDate)

Group	Database
Execution	Asynchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Not supported

- **Description:** Synchronizes the alarm [Event, Trend] database.
- **Parameters:**

strStartDate	String with the start date. If this parameter is not specified, the current date is used.
strOptionalEndDate	String with the end date. If this parameter is not specified, the function uses the start date.

- **Returned Values:**

1	Failure to start synchronization; the database is probably being synchronized.
0	Success
-1	Invalid group number
-2	The format is not set to “Database.”
-4	Start date specified is invalid.
-5	End date specified is invalid.
-6	Start date is greater (later) than the end date.

- **Examples:**

Tag Name	Expression
Tag	SyncAlarm() //Synchronizes the database using the current date
Tag	SyncAlarm(“10/20/2004”) //Synchronizes the database only for the day 10/20/2004
Tag	SyncAlarm(“10/20/2004”, “10/28/2004”) //Synchronizes the database from 10/20/2004 to 10/28/2004

SyncEvent(strOptionalStartDate, strOptionalEndDate)

Group	Database
Execution	Asynchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Not supported

- **Description:** Synchronizes the alarm [Event, Trend] database.
- **Parameters:**

strOptionalStartDate	String with the start date. If this parameter is not specified, the current date is used.
strOptionalEndDate	String with the end date. If this parameter is not specified, the functions uses the start date.

- **Returned Values:**

1	Failure to start synchronization; the database is probably being synchronized.
0	Success
-1	Invalid group number
-2	The format is not set to “Database.”
-4	Start date specified is invalid.
-5	End date specified is invalid.
-6	Start date is greater (later) than the end date.

- **Examples:**

Tag Name	Expression
Tag	SyncEvent() //Synchronizes the database using the current date
Tag	SyncEvent(“10/20/2004”) //Synchronizes the database only for the day 10/20/2004
Tag	SyncEvent(“10/20/2004”, “10/28/2004”) //Synchronizes the database from 10/20/2004 to 10/28/2004

SyncTrend(numGroup, strOptionalStartDate, strOptionalEndDate)

Group	Database
Execution	Asynchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Not supported

- **Description:** Synchronizes the alarm [Event, Trend] database.
- **Parameters:**

numGroup	Trend group number
strOptionalStartDate	String with the start date. If this parameter is not specified, the current date is used.
strOptionalEndDate	String with the end date. If this parameter is not specified, the functions uses the start date.

- **Returned Values:**

1	Failure to start synchronization; the database is probably being synchronized.
0	Success
-1	Invalid group number
-2	The format is not set to “Database.”
-4	Start date specified is invalid.
-5	End date specified is invalid.
-6	Start date is greater (later) than the end date.

- **Examples:**

Tag Name	Expression
Tag	SyncTrend(1) //Synchronizes the group 1 database using the current date
Tag	SyncTrend(1, “10/20/2004”) //Synchronizes the group 1 database only for the day 10/20/2004
Tag	SyncTrend(“10/20/2004”, “10/28/2004”) //Synchronizes the group 1 database from 10/20/2004 to 10/28/2004

SyncAlarmStatus( )

Group	Database
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Not supported

- **Description:** Returns the synchronization status.
- **Returned Values:**

3	Synchronization has finished.
2	Failure to synchronize
1	Still synchronizing
0	No synchronization is being executed.
-1	The format is not set to “Database.”

- **Examples:**

Tag Name	Expression
Tag	SyncAlarmStatus()

SyncEventStatus( )

Group	Database
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Not supported

- **Description:** Returns the synchronization status.
- **Returned Values:**

3	Synchronization has finished.
2	Failure to synchronize
1	Still synchronizing
0	No synchronization is being executed.
-1	The format is not set to “Database.”

- **Examples:**

Tag Name	Expression
Tag	SyncEventStatus()

SyncTrendStatus(numGroup)

Group	Database
Execution	Synchronous
Windows NT/2K/XP	Supported
Windows CE	Supported
Web Thin Client	Not supported

- **Description:** Returns the synchronization status.
- **Parameters:**

numGroup	Trend group number

- **Returned Values:**

3	Synchronization has finished.
2	Failure to synchronize
1	Still synchronizing
0	No synchronization is being executed.
-1	The format is not set to “Database.”

- **Examples:**

Tag Name	Expression
Tag	SyncTrendStatus(1)

Revision History

Revision	Author	Date	Comments
A	André Bastos	March 26, 2003	Document created
B	Fabio Terezinho	April 2, 2003	Content revision
C	Fabio Terezinho	October 3, 2003	General revision
D	André Bastos	March 11, 2005	General revision