GE VERNOVA

PROFICY® SOFTWARE & SERVICES

PROFICY iFIX HMI/SCADA

Optimizing Your iFIX System

GE VERNOVA

Proprietary Notice
The information contained in this publication is believed to be accurate and reliable. However, GE Vernova assumes no responsibilities for any errors, omissions or inaccuracies. Information contained in the publication is subject to change without notice.

Trademark Notices
"GE VERNOVA" is a registered trademark of GE Vernova. The terms "GE" and the GE Monogram are trademarks of the General Electric Company, and are used with permission.

Microsoft® is a registered trademark of Microsoft Corporation, in the United States and/or other countries.

All other trademarks are the property of their respective owners.

We want to hear from you. If you have any comments, questions, or suggestions about our documentation, send them to the following email address:
doc@ge.com

Table of Contents

Optimizing Your iFIX System

The Optimization Guide is intended for systems integrators and developers who want to achieve maximum performance from their iFIX® systems. The manual offers tips and strategies you can use while developing pictures, writing scripts, and implementing your iFIX system.

Reference Documents

For related information on subjects discussed in this manual, refer to the following manuals:

- Creating Pictures
- Writing Scripts
- Mastering iFIX

Introduction

With the introduction of the iFIX® product, we presented you with a powerful tool. GE has the first fully-integrated family of software automation products based on open, component-based technology. It is designed to remove the constraints of packaged software and allow easy integration and interoperability between your plant floor and business systems, as well as between components and third-party applications.

The inclusion of Visual Basic for Applications in iFIX presents you with a powerful scripting tool that allows you to quickly and easily automate operator tasks and create automation solutions. By providing you with an open graphical environment, iFIX enables you to incorporate internet controls (OCXs), reuse elements from other sources (such as bitmaps), or embed other OLE automation applications into your pictures.

Like all tools, the successful implementation of these technologies requires careful planning and consideration. Your overall design strategy should consider using this new functionality to its maximum benefit, while striving for peak system performance.

Each iFIX system will be different, so we don't offer any absolutes, but rather some suggestions to make your design and implementation more effective. This guide contains some strategies to consider while:

- Designing pictures
- Writing scripts
- Configuring the hardware and software in your iFIX system

Designing Your Pictures to Achieve Maximum Performance

The most time intensive development task is the planning and creation of your iFIX displays. iFIX offers you an abundance of features to use in your pictures. Each feature used independently can greatly enhance your picture. However, to increase the overall performance of your system, you want to choose your features carefully and implement them efficiently so they do not impact your system's overall performance. By using these tools wisely and in situations where they provide the greatest benefit to the display, you can greatly optimize your system's performance.

When designing your pictures, pay close attention to the overall number of objects in each picture. Minimizing the number of objects in your pictures will provide the greatest boost to your picture performance.

The following sections contain some suggestions you should consider when developing your iFIX displays:

Using Animations

Use animations instead of scripts whenever possible to set an object's properties. Animations are created with optimized C++ code and will always be faster than similar functionality scripted with VBA.

Blinking in iFIX Pictures

Blinking objects can be an effective device for attracting an operator's attention to a screen. However, when you use blinking in iFIX pictures, it forces a repaint of the region in which the blinking object resides. This can be costly, in terms of your system's resources, so you should use blinking judiciously and only in the displays in which it is truly needed.

Blink rate The default blink rate is .5 seconds. If you do not need to have the object blink that fast, adjust the rate to slower interval. This will help conserve system resources.

To change the blink rate:

1. Open the Animations dialog box for the object you want to animate and select the Color tab.
2. Click the Animate check box for the color property you want (ForegroundColor, BackgroundColor, or EdgeColor).
3. Click the Advanced button and, in the Other Options area, enter a blinking rate (in seconds) in the Toggle Rate field.

Using ActiveX Controls

You can and should use ActiveX controls to add functionality to your iFIX pictures. However, it is important to use these controls wisely and in moderation. The following subsections detail some of the finer points of using ActiveX controls in your pictures:

- Windowless Controls
- Using Shapes Instead of ActiveX Controls
- Animations on ActiveX Controls
- Keep Message Reflection Enabled

Windowless Controls

Use windowless controls whenever possible. These controls process messages faster and draw faster than their windowed counterparts.

When you are developing your iFIX pictures and developing any new controls, remember that windowless controls go behind shape objects in your displays; windowed controls will never go behind iFIX shapes. You can use this information to test whether a control is windowed or windowless.

To test a control:

1. Place the control in a picture.
2. Draw a rectangle in the same picture.
3. Select the control and then, in Classic view, select Send to Back from the Format menu. In Ribbon view, on the Format tab, in the Arrange group, click Arrange, and then click Send to Back.

If the control goes behind the rectangle, it is a windowless control. If the control stays on top of the rectangle even after you have selected the Send to Back command, then the control is windowed.

Using Shapes Instead of ActiveX Controls

Generally, shapes are faster than ActiveX controls, so wherever possible, use shapes instead. For example, the push button in the toolbox is an ActiveX control. Using this control several times in a picture is fine. However, if you have a display where you need to use many push buttons, or if you need to

use a push button in a picture whose performance is critical to your operation, create a bitmap of a rectangle, and then change the button style of the bitmap to be a push button.

Animations on ActiveX Controls

Minimize the use of animations on ActiveX controls. Using animation on an ActiveX control can be slow and expensive to the system's resources.

Keep Message Reflection Enabled

To improve the performance of your controls, we recommend that you keep Message Reflection enabled, which is the default setting, unless a specific control requires it to be disabled. As examples, the **MSTreeView** control, the **MSListView** control, and the **MSUpDown** control each need message reflection disabled.

Some messages are normally sent to the parent control. Under normal conditions, these messages are actually reflected back to the sending control, so that the control can handle its own message. This message reflection can be handled by the container, which will reflect the messages back as events.

Message reflection is an ambient property of the container. The message reflection feature is relevant to controls that are implemented by sub-classing a windows control. The iFIX WorkSpace supports message reflection to gain performance. If the container does not support message reflection, then every sub-classed control creates an extra window around itself.

To enable or disable Message Reflection:

1. Open your picture.
2. Select Picture from the right-click menu. The Edit Picture dialog box appears.
3. Enable or disable Message Reflection by selecting or clearing the Message Reflection check box.

Using Bitmaps in iFIX Pictures

You can use bitmaps strategically to help improve your system's performance. For example, if you have a picture with 40 objects located within 1 or 2 screen regions that have no animations attached to them, consider converting the objects to a bitmap. This ensures that the next time that display has to repaint, only one item (the bitmap) will need to be repainted, rather than the 40 separate objects.

Metafile imports, for example CAD drawings, create a large number of objects. Again, one strategy of optimizing a picture created with a metafile import is to convert these multiple objects to a bitmap.

When using bitmaps, remember that the size and the resolution of the bitmaps can impact performance, too. Larger bitmaps and bitmaps at high resolution will take longer to load and will tax your system's resources more than smaller bitmaps at a lower resolution. As a general rule of thumb, you will also achieve better performance if you convert multiple bitmaps within one screen region to a single bitmap.

You should try to minimize the use of full screen bitmaps, too. A full screen bitmap will require all screen regions to repaint, which is also costly to your system's resources.

To copy and paste an object as a bitmap image:

1. Select the object.
2. In Classic view, on the Edit menu, click Copy as Bitmap.

   -Or-

   In Ribbon view, on the Home tab, in the Clipboard group, click Copy as Bitmap.
3. In Classic view, on the Edit menu, click Paste.

   -Or-

   In Ribbon view, on the Home tab, in the Clipboard group, click Paste, and then click Paste.

Choosing the Type of Bitmap to Use

There are two format types of bitmap files that iFIX can use in creating pictures. The first type is the device dependent bitmap, or DDB. It is the default format used by iFIX. This type of format displays quickly but may be more taxing to system resources than the other bitmap format, the device independent bitmap, or DIB. This type of bitmap is less taxing to system resources and contains all of the color information within itself, independent of the current graphics card or driver.

If you find that you are having trouble displaying pictures due to running out of system resources, you should use the DIB format for all of your bitmaps.

To change the bitmap type:

1. In the LOCAL folder, locate and open the FixUserPreferences.ini file.
2. In the Picture Preferences section, add the following line:

   ```
   OnDrawUseDIB=1
   ```
3. Save and then close the file.

Refreshing iFIX Displays

You can use the Refresh Rate Expert to set the refresh rate of all the objects in your picture. What is important to remember about refresh rates is that the more unique refresh rates in a picture, the greater the potential impact to the overall system performance. You may certainly implement a slow rate for some objects and a fast rate for other objects, but if you try to finely tune multiple objects with many different rates, you may actually make your overall refresh performance slower.

The following recommendations may help you to optimize your system performance while implementing a refresh rate strategy:

- Leave the default rate of 1 second for the refresh rate of every animation in a picture. Alternatively, use a slower rate (such as 5) for the objects that can have slow updates, and a fast rate (such as .1 second) for objects requiring faster updates. More unique update rates will tend to result in slower performance.
- Avoid using a refresh rate of 0. This is the fastest refresh rate possible, and will, therefore, have the greatest impact to your system's performance. The performance impact will be greatest on the SCADA nodes which host the data.

- When you modify pictures that were created with early versions of iFIX, use the Refresh Rate Expert to set the older pictures' update rates to 1 second or greater.

Using Auto Scale

Auto scale is a powerful and important feature in iFIX, but it can be a costly operation in terms of system performance. When adjusting your picture size, you work with the window property, and you define the size as a percentage of screen. By sizing a window by the percentage of screen, iFIX auto scales your picture, maintaining its Logical Coordinate System and displaying the picture at the same size, regardless of the monitor's resolution it is viewed on.

NOTE: Auto scale does not apply to graphics that use the Enhanced Coordinate System.

If picture open or picture replacement performance is your key concern, make certain that you run resolve on the picture you want to auto scale at the resolution the picture will ultimately be viewed. For example, if you have developed a picture at a 1024 x 768 setting and it will ultimately run on a machine with a 1280 x 1024 setting, run resolve on the picture at the 1280 x 1024 setting, to improve performance.

Be aware that picture auto-scaling is enabled, by default, in the iFIX User Preferences dialog box, on the Picture Preferences tab. If you have multiple monitor configurations, each with different resolutions, select the Disable Auto Scale Feature check box on the Picture Preferences tab to override the logical units to pixel ratio when changing the resolution of your screen.

Picture auto-scaling is designed to work when moving pictures between monitors where the aspect ratio is the same. If you move a picture to a monitor with a different aspect ratio, upgrade the picture to use the Enhanced Coordinates feature. For more information on Enhanced Coordinates, refer to the Upgrading Pictures to Enhanced Coordinates section in the Creating Picture e-book.

Resolving iFIX Displays

When you upgrade your iFIX system, or add or delete tags from your database, it is important that you run resolve on all pictures and schedules.

Resolving Tags

If there are tag groups in the picture, run resolve from the Tag Group Editor once the final changes to the picture are made.

Using ReplacePicture

When using ReplacePicture, making both pictures the same size will improve this operation's performance. You can use the Generate Picture Expert, which you can access from the Toolbox, to achieve this result.

Simplify Overview Pictures

One strategy you may choose to implement when designing your iFIX displays is to simplify your overview pictures. While some of your more specific displays may require a high level of detail, it may be possible for you to use a more simple representation to create your overview displays.

Try to avoid combining many detailed displays to create your overview picture. This will almost certainly result in a picture with vastly more objects than are absolutely necessary, and may then impact your system's performance.

Using the Alarm Summary Object

Using the Alarm Summary object in your pictures is an effective way to monitor, acknowledge, sort, and filter alarms. However, many people choose to place the Alarm Summary OCX in every picture. This results in a query to the database to update the Alarm Summary object every time an additional picture is open.

Consider a design strategy where you place the Alarm Summary OCX in one picture that always remains open.

Minimize the Number of Objects in Each Picture

Use global threshold tables and consider converting lots of small objects to bitmaps to minimize the number of objects in each picture.

Evaluating the Use of Groups

Groups are a convenient way of organizing pictures. You can duplicate groups to help you quickly develop your pictures. The important thing to remember about using groups is that each group is a separate object and the overuse of grouping, especially in the design of Dynamo objects, can lead to pictures with hundreds of unnecessary objects. This can negatively impact the call-up and refresh performance of pictures that use these objects.

You might want to consider a strategy where you use grouping freely during the design phase of your project and then selectively ungroup some objects before you use your pictures in run time. For instance, perhaps you have a group of objects (let's call this Group A), and this group does not have a script or an animation associated with it. Group A is also part of a larger group, Group B, which does have some animated properties. You may have chosen to group the objects in Group A at design time, because you duplicated this group and used it in many pictures. At run time, there really is no reason for Group A to remain grouped, so consider ungrouping it.

This strategy is not meant to imply that groups or Dynamo objects shouldn't have sub-groups; it simply suggests that you use groups judiciously. For example, if your main group has 6 objects, it probably doesn't need 4 sub-groups. However, if your group includes hundreds of objects, it may make sense to cluster the objects into a few smaller groups so that you can more easily duplicate or edit the objects in your picture.

Reviewing the Number of Open Pictures

Consider the number of pictures you have open on a node at any given time. The number of open pictures may impact the system's available memory as well as the speed with which you can access data.

Using Timer Objects

You should attempt to minimize the number of Timer Objects in any one picture. When you do include Timer Objects in a picture, set the Timer to run at the slowest rate acceptable to your application, in order to optimize your system's performance.

Using VBA and Scripting to Achieve Maximum Performance

The ability to use VBA and scripting throughout iFIX opens the system for almost limitless uses. VBA can be used to significantly customize and extend the functionality of iFIX. As with all good tools, you need to develop a sound strategy for implementing VBA in your applications. It is important to remember that you should use VBA to leverage additional functionality, rather than using it to replace existing, built-in iFIX functionality.

While addressing the fundamentals of how to write solid VBA code is beyond the scope of this document, this chapter does contain some suggestions for effectively using VBA in your iFIX process. This chapter includes the following sections:

Tuning Your Scripting Performance

You can adjust a setting in the FixUserPreferences.ini file to fine tune your script's performance. In the [Scripting] section of the FixUserPreferences.ini, which is located in your C:\Program Files (x86)\Proficy\iFIX\Local directory, the ScriptIdleTimeProcessingDelayCount is set to -1, by default. At this default setting, your script performance will take precedence over your graphic's performance. You can change this setting to 0 or 1 to improve the performance of your graphics.

One strategy you may want to use is to maintain the setting at -1 for faster processing of scripts on your SCADA node, and to change the setting to 0 or 1 for faster loading of your graphics on your View nodes.

NOTE: Do not change the setting to a number larger than 1. A larger number will not improve your performance.

Cleaning Your Code

In the course of developing your displays, you may place some objects into VBA. As your displays develop, you may realize that you do not need VBA functionality on all of these objects.

Your system's performance will improve if you reduce the number of VBA objects and VBA procedures to only those that you truly need. As part of a clean up process before putting your pictures into run time,

remove all unused event procedures and remove empty VBA scripts. When there are no scripts configured for an object and the object is not referenced by other scripts, it will be omitted from the VBA processing queue.

Accessing the Database from Scripting

You can access the database from scripting by:

- Using the Fix32 object
- Executing a FindObject on the tag name
- Using the FixDataSystem object
- Accessing animated objects

When evaluating the first three methods above, consider how each method ranks in terms of ease of use and performance.

Scripting Method	Ease of Use	Performance
Using the Fix32 Object	High	Low
Executing a FindObject on the tag name	Medium	Medium
Using the FixData System object	Low	High

Each of these methods has its own advantages and disadvantages, which are detailed in the following subsections.

Using the Fix32 Object

The Fix32 object is by far the easiest to use, but it is expensive in terms of performance. Every time you execute a line of code that uses the Fix32 object to retrieve or set values, the WorkSpace will:

1. Instantiate a new data item.
2. Validate the item.
3. Read/write a value.
4. Destroy the object.

This will be especially slow if the process is attempting to access a tag that is remote to the local node.

Example

The following code example demonstrates using the Fix32 object to access thisnode:tag.f_cv:

```
'Writes
Fix32.thisnode.tag.f_cv = 1.0 ' Write a 1 to thisnode:tag.f_cv

'Reads
Dim X as Single
x = Fix32.thisnode.tag.f_cv 'Read the value to thisnode:tag.f_cv and assign it to single precision variable x
```

Using the FindObject Method

The FindObject method will return a data item that will persist as long as the variable you assign it to exists. This method is faster than using the Fix32 object because the object will only validate itself when the FindObject executes and returns a data item. As long as the variable does not get destroyed or is not assigned a new object, you can read and write to it without the WorkSpace re-validating it.

Example

The following code example demonstrates using the FindObject method to access thisnode:tag.f_cv:

```
Dim DI as Object
Dim X as Single

set DI = System.FindObject("Fix32.thisnode.tag.f_cv")

X = DI  'Read the value of thisnode:tag.f_cv and assign it
        'to single precision variable x
DI = 1  'Write a 1 to thisnode:tag.f_cv
```

Using the FixDataSystem Object

The FixDataSystem object is slightly more difficult to configure, but is ultimately the most efficient method of accessing the database. This method retains groups of data item objects. This method is faster than using the FindObject method because data items are read and written collectively as a group. In other words, when you execute the group read method of a FixDataSystem object, it will read the data for all the tags added to it in a single call. When you retrieve the value from the data item inside of the DataItems collection, it actually retrieves it from a buffer rather than requesting the data from the database. If you are continually reading from or writing to a large number of tags, you should consider using the FixDataSystem object.

Example

The following code example demonstrates using the FixDataSystem object to access thisnode:tag.f_cv:

```
'Create the Data System OCX
Dim FDS As Object
Dim Y as single

Set FDS = CreateObject("FixDataSystems.Intellution FD Data _
     System Control")

FDS.Groups.Add ("DataGroup1")
'Add a group to the Groups collection

FDS.Groups.Item("DataGroup1").DataItems.Add("Fix32.thisnode._
     tag.f_cv")  'Add the item to the group

FDS.Groups.Item("DataGroup1").Read
'Read all the items in the group

Y = FDS.Groups.Item("DataGroup1").DataItems.Item(1).Value
'Get the value which was cached in the read

FDS.Groups.Item("DataGroup1").DataItems.Item(1).Value = 3.0  'Change the cache

FDS.Groups.Item("DataGroup1").Write
'Write out all the caches in the group (only write those
'items which have changed)
```

Accessing Animated Objects

Another method of performing efficient reads is accessing the value of an object that is already connected to a tag in the database. When you animate an object's property to a tag in the database, it creates a subscription to the database. Whenever this tag changes, that property will be changed to the tag's new value. If you read this property, you will be reading the value that is in the database without adding any additional overhead. You can use any object in any picture or in the globals page including event objects, variable objects, rectangle objects, and so forth.

Example

This example assumes that there exists a picture ("Picture") that contains an animated rectangle ("Rect1"). Rect1's horizontal position is animated to thisnode.tag.f_cv

```
Dim AO As Object
Dim Y As Single

Set AO = System.FindObject_
     ("Picture.Rect1.AnimatedHorizontalFillPercentage")

Y = AO.InputValue
'Get the current value of the animated horizontal position input
'which gets the value of thisnode:tag.f_cv
```

Placing Code in the Initialize and Activate Events

Large amounts of code in the Initialize and Activate events will impact when you can first interact with the picture when it is opened. For example, a picture with a button on it may open almost immediately, and you can click on that button, but you will not be able to trigger any event tied to that button until all the scripts in the Initialize and Activate events have run to completion.

Also, avoid adding tags to the DataSystem OCX in the picture initialize. Move this action to the FixGlobals_Initialize event of the User Project, design it to load the tags on demand, or initialize it in a global script.

Using iFIX Subroutines and Experts

iFIX includes several subroutines that can help simplify scripts that are intended to perform common tasks, such as acknowledging alarms or replacing pictures. Since these subroutines are stored in the FactoryGlobals project, they can be automatically accessed directly through the Visual Basic Editor.

In addition to providing the code necessary to perform the task at hand, subroutines offer several "extras" that you would normally have to code yourself, including:

- Generic error handling.
- Posting of operator messages to alarm areas.
- Conformance to VBA naming conventions.

Working with the Scheduler

Before you start scheduling entries, you should understand how the Scheduler executes scripts. The Scheduler can run as a foreground or background task.

When you run a schedule in the foreground, it runs on the same thread as the scripts in the iFIX WorkSpace, allowing you to quickly test and debug scripts in the run-time environment.

When you run a schedule in the background, it runs on a separate thread from the scripts in the iFIX WorkSpace and therefore is not competing with the other scripts that may be running in the WorkSpace.

Refer to the following sections for more information on working with the Scheduler:

- Resolving Schedules
- Running Schedules
- Using Command Wrappers
- Using a Single Timer or Event Object

Resolving Schedules

Database information for each data source in a schedule is saved as part of the file on disk. When a schedule executes in the run-time environment, and the database information is not up-to-date, iFIX queries the database and resolves the tag definitions. This can be a time-consuming process the first time the schedule loads, which can slow the execution of the schedule.

Resolving schedules removes the need for iFIX to query the database. Therefore, resolved schedules execute faster in the run-time environment. We recommend resolving your schedules as a final step in the configuration process. The Resolve feature is available for both schedules and pictures.

If the process database is modified after you resolve a schedule, you need to run Resolve again.

To resolve a schedule:

1. Click the Resolve Files button on the Utilities Toolbar.
2. In the File Types area, select the Schedule check box.
3. In the Directories area, verify that your schedule directory is correct. If it is not, enter the correct path in the Schedules field.
4. In the Resolve Files area, select the schedule file you want to resolve. To select every file in the list, click Select All.
5. Click the Update Schedule Files button and click OK.

Running Schedules

Run schedules that do not pop up forms in the background. When forms or message boxes pop up, that schedule's VBA queue is paused until the form is closed or the message box is acknowledged.

Using Command Wrappers

If you are using command wrappers in schedules that run as a background task, you can specify an Error Option as an additional, optional parameter. By including an Error Option, you can prevent a message box from appearing and delaying the complete processing of your script.

By using 2 as your Error Option parameter, you can send errors to Alarm History. By using 1 as your Error Option parameter, you can handle the error within your timer or event script. From there, for example, you can send it to the Windows Event Log.

Using a Single Timer or Event Object

If you want to run a timer or an event, you can put it directly into a picture. From a performance standpoint, it is not necessary to put one timer or event object into a schedule if it is only used when a particular picture is open. Simply put the timer or event object into that picture.

Referencing Pictures or Global Pages

Do not reference other pictures or global pages in VBA unless they will already be opened in your runtime environment. Referenced pictures that are not already open will need to be opened, which will impact your system's performance.

Using UNC Pathing with VBA

If you use UNC pathing in your VBA references, it may cause your pictures to open very slowly. If this happens, you can correct it by modifying the FixUserPreferences.ini file so that the system searches for copies of the referenced file on the local disk. Use the following procedure to address the situation.

To cause the system to search the local disk for copies of referenced files:

1. Shut down iFIX if you haven't already done so.
2. Open the FixUserPreferences.ini file in a text editor, such as Notepad.
3. In the FixUserPreferences.ini file, add the following section and key:

```
[VBAReferences]
LoadLocalReferencesFirst=1
```

4. Save the FixUserPreferences.ini file and restart iFIX.

User Globals

Place scripts that will be referenced by many pictures in the User Globals page. The User page is the location where you can put your own objects, methods, forms, and variables that you want to use globally.

If made public, the objects, methods, forms, and variable objects contained in the User page can be accessed from anywhere within your system. Since the items that you define as public in the User page can be accessed from anywhere in the system, make sure that what you enter is really what you want to expose. If you create a global public variable, remember that it can be changed from any script at anytime.

System-wide Optimizations

It is difficult to make hard and fast rules regarding hardware, because each iFIX configuration is so different. However, when it comes to hardware, the following guidelines are true:

- With hardware, more is better. More memory, more processing speed, and better graphic cards can all help to improve performance.
- Distribute the other applications your system is running.
- Try to minimize the amount of CPU used in steady state.

These methods of optimizing your system's hardware, as well as some additional system-wide optimizations, are discussed in the following subsections:

Upgrading Your Hardware

The minimum hardware requirements listed in the Setting Up the Environment manual are a suggested starting point. While that hardware configuration will run your iFIX system effectively, upgrading to an even faster processor and adding additional RAM to your computers will increase your system's performance.

You may also want to consider using SCSI drives rather than IDE drives. The use of SCSI drive systems may improve the speed of your open picture and replace picture operations.

Additionally, you may want to consider upgrading the graphics card in your PC to one with additional memory.

Installing an Accelerator Card

Consider adding an accelerator card to your computer, especially a node that is processing lots of graphics. An accelerator card typically has 1 MB of special video memory. Your computer's performance may be enhanced because this memory will hold a screen image and allow the computer's main memory to be allocated for other processing tasks. If your computer is displaying very high resolution graphics, animations, or pictures containing many colors, you may want to upgrade to a card with 2 or more MB of video memory. Many of the available graphics accelerators contain at least 2 MB of video memory in their video cards.

Video Bit Count

The number of data channels the video card can load at one time is called the data bit count. The more data channels your card has, the faster it will be able to display an image on your screen. More data

channels also translates into higher resolution images and better picture quality. Most VESA accelerator cards use 24 or 32 data channels. These are also referred to as 24-bit or 32-bit cards. PCI accelerator cards are 64-bit and the high-end AGP cards are 128-bit accelerators. Essentially, you want a card with the most bits.

NOTE: If you decide to use an accelerator card, be sure to match your video card to your bus type.

Defragment the Hard Disk

A fragmented hard drive can greatly reduce your computer's performance. Check regularly for disk fragmentation and schedule time to run utilities like Norton Speed disk. Taking the time to defragment your hard drive may increase the speed in which non-cached graphics and historical files will open.

Distributing Additional Applications

Distributing additional applications to other nodes in your system can help to maximize iFIX's performance. SQL Server or other databases used in your system can greatly impact the performance of your iFIX nodes. By setting up machines dedicated to reporting, e-mail, and other peripheral applications, you allow the iFIX dedicated machines to allocate all their resources to iFIX applications, therefore improving your iFIX system's performance.

Minimizing the Amount of CPU Used in Steady State

Give some consideration to the demands on a node in its steady state, as you design your overall system. If a significant amount of the CPU on your SCADA node is used in the steady state, a plant event or an operation that suddenly triggers multiple alarms, fires several scripts, and opens many pictures may strain the computer's resources. Remember that much of what a process control system does is change-driven, and therefore the demands on the system will be higher at these times than under steady state.

Reporting applications and Historical Collection can be distributed over several nodes to increase overall throughput and an individual computer's performance. Remember that iFIX's distributed architecture allows you to access data anywhere on the network, so distribute the processing load throughout the nodes on the system.

Evaluating Network Performance

Be aware that the refresh rate on pictures on the iClient computer is what impacts the network performance most. Consider decreasing the refresh rate of your pictures to optimize the performance of a busy iFIX network. For more information, refer to the Refreshing iFIX Displays section.

If your facility's network speed is slow, you should consider keeping your iFIX pictures on a local machine, rather than retrieving them over the network from a server.

Best Practices for Managing Multiple iFIX Users

You will likely have multiple iFIX users to manage when running iFIX with Microsoft Terminal Server. You could also have multiple iFIX users, when you have different user types using a single node, for instance.

When working with multiple iFIX users, you should first plan the types of users your iFIX application includes, as well as the iFIX folders that these users might share. You then define a project for each user type. In some cases, such as with developers, each user of a given type may require his own project.

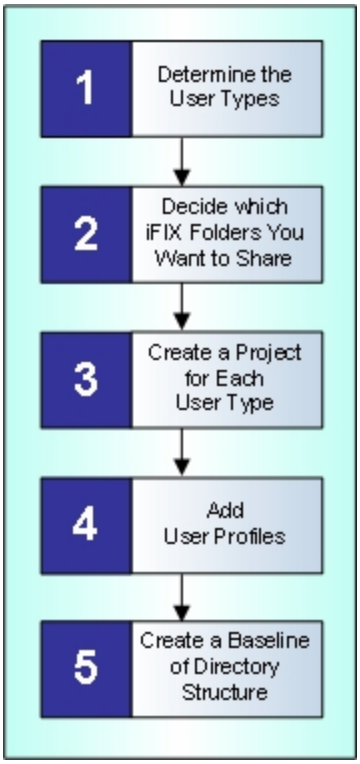
Examples of user types include: Operators, Supervisors, and Developers. Other user types could include other logical groupings such as: Line1, View, or SCADA users.

When you create the project, you organize the iFIX project folders, specifying which folders you want to share, if any, among users of different types. Once all the projects are created, you can generate a baseline snapshot of the folders and files, for maintaining these projects in the future.

To assist with managing multiple users, use the following iFIX applications:

- System Configuration Utility (SCU)
- Startup Profile Manager
- Application Validator

The steps described in the following figure explain the best practices for configuring iFIX to be used with multiple users. Click a block in the diagram to jump to that section.



Overview How to Configure iFIX for Multiple Users

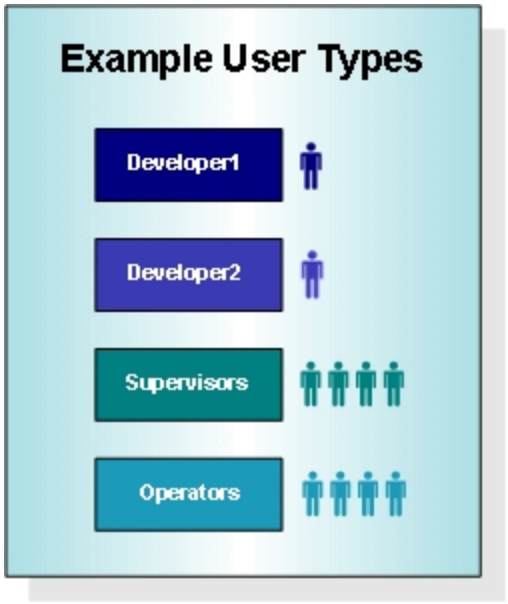
Determine the User Types

Before defining any iFIX projects, determine what types of users you need for your iFIX application. Planning user types before you actually create projects for them, instead of as an afterthought, assists in project management and saves you time in the long run.

Start by thinking about the names that you might want to organize users by. For instance, you might want to organize users by function, location, or even node type. The following table shows some examples.

User Grouping	Example User Types
By Function	Operator, Supervisor, Maintenance, Developer
By Location	Line1Viewers, Line2Viewers
By Node Type	VIEW, SCADA

The following figure shows one example of the names that you might give to each user type when organizing users by function. Your iFIX application can use these groupings, or other ones. This is just one example of many scenarios.



Example of User Type Groupings

Once you decide on names to group users by, consider which groups will:

- Share the same preferences?
- Be able to configure their own historical collection settings?
- Work with recipes?
- Develop pictures?
- Have development rights?

- Have run-time rights?
- Work with one or more iFIX projects?

Anticipating the needs of your users allows you to successfully determine the configuration of your projects.

Decide What Folders You Need To Share

For each iFIX user type, you create a project. Before creating the project, however, you should plan the folder hierarchy. When you go to create a project, you will need to specify the paths for the following folders:

- LOCAL
- PDB
- PIC
- APP
- HTR
- HTRDATA
- ALM
- RCM
- RCC

For a description of what each of these folders contain, refer to the Planning SCU Directories section.

Each project can have its own set of these folders. For instance, if you have different groups of developers, you may want to create a separate directory that includes all of these folders for each developer's project, so that multiple developers can work on different types of projects at the same time.

Other projects may share some of these folders. For instance, you might only want to share the iFIX picture path in the C:\Program Files (x86)\Proficy\iFIX folder, but have all the other folders be project-specific.

NOTE: The Base and the Language folders should be the same for all users. These folders are not project-specific.

Recommendations on Folder Sharing

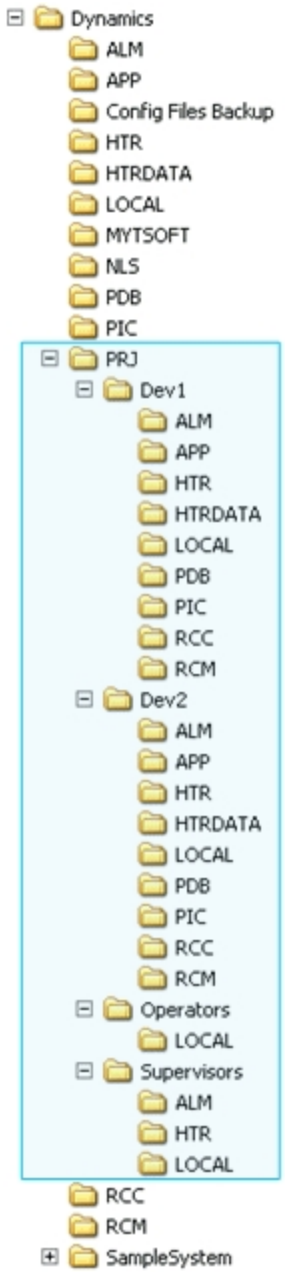
For each project, it is recommended that certain types of users only share specific directories. The following table outlines these recommendations.

	User Types and Directories to Share		
Project	**Project-Specific Directories**	**Shared Directories**	**Example User Type**
Run Time Only	LOCAL	PDB, PIC, APP, HTR, HTRDATA, ALM, RCM, and RCC	Operator

Run Time with Historical Collection	LOCAL, HTR	PDB, PIC, APP, HTRDATA, ALM, RCM, and RCC	Supervisor
Special Alarming Run Time	LOCAL, ALM	PDB, PIC, APP, HTR, HTRDATA, RCM, and RCC	Supervisor
Development	LOCAL, PDB, PIC, APP, HTR, HTRDATA, ALM, RCM, and RCC	None. Developers should have all unique directories except for NLS and the BASE path, which should always be shared.	Developer

Example of the iFIX Directory Structure for Multiple Projects

The following example shows how your iFIX directory structure might look if you created projects for Operators, Supervisors, and Developers. This directory structure is created when you create your projects.



Example of the iFIX Directory Structure for Multiple Projects

Create a Project for Each User Type

After you finish planning user types and directories, you are ready to create projects that organize these user types and directories. To create projects, you use the iFIX System Configuration Utility (SCU). For each project there is a separate, unique SCU file.

To access the SCU, double-click the System Configuration icon in the WorkSpace system tree, or click the Start button and point to Programs, iFIX, and then System Configuration. You can also start the

SCU directly from the iFIX Startup dialog box, also known as the Launch application. Make sure you shut down iFIX before starting the SCU.

To create a project in the SCU:

1. Start the SCU.
2. On the Configure menu, click Paths. The Path Configuration dialog box appears.

   IMPORTANT: The Base and Language Paths should be the same for each project. For instance, if you leave the Base and Language set to the defaults, the Base is set to C:\Program Files (x86)\Proficy\iFIX and the Language is set to C:\Program Files (x86)\Proficy\iFIX\NLS for each project. Leave the Base and Language paths set to the default, and proceed by editing the Project path fields.
3. In the Project field, enter a path for the project. For example, a valid path that you might enter for a developer would be: C:\Program Files (x86)\Proficy\iFIX\PRJ\Dev1.
4. Click the Change Project button. A message box appears asking if you want to add the default iFIX files to the new project.

   IMPORTANT: The SCU will not copy the files or subdirectories from the existing directories to the new directories.
5. Click Yes. The project path information from the Project field is appended to the other project path fields, such as Local, Database, Picture, Application, and so on.

   For instance, if you enter C:\Program Files (x86)\Proficy\iFIX\PRJ\Dev1 in the Project field, the SCU automatically adds \PRJ\Dev1 to the other project fields as well. For instance the Local path in this dialog box would now read: C:\Program Files (x86)\Proficy\iFIX\PRJ\Dev1\Local. The Database path would read: C:\Program Files (x86)\Proficy\iFIX\PRJ\Dev1\PDB, and so on.

   NOTE: While the paths displayed in the Path Configuration dialog box appear to be added at this point, the actual folders for these paths have not been created yet. While you can view the new paths from this dialog, you will not be able to view them from the Windows Explorer until you complete the remaining steps in this section.
6. If there are any paths that you want to change, such as to a shared directory, manually edit the path fields.
7. Click OK from the Path Configuration dialog box. A message box appears asking you to create the folders for the configured paths.
8. Click Create All. A message box may appear indicating you do not have a valid Alarm Area Database file.
9. Click Proceed to continue.

   iFIX creates the paths for the project folders. You should be able to view the new folders in Windows Explorer.
10. On the File menu, click Save As. The Save File As dialog box appears.
11. Browse to the Project's Local folder. For instance, for Developer1 browse to the C:\Program Files (x86)\Proficy\iFIX\PRJ\Dev1\Local folder.
12. Enter a name for the SCU file.

    Valid SCU file names can be up to eight characters long. SCU file names can include alphanumeric characters, but must begin with a letter. Special characters, such as symbols and punctuation marks, cannot be used.

    For example for Developer1, you might enter Dev1.

13. Click Save. A message box appears asking you if this is the SCU file that you want to use next time you restart iFIX.
14. Click No.
15. If you have a template folder with iFIX files that you want to copy into one or more project folders, use the Windows Explorer to copy the files from the template folders into the new project folders.

Once you have created all the projects, you can go back later and edit all the other SCU settings for each project that require changing. For more detailed information about configuring the rest of the options for the SCU files, refer to the Configuring iFIX Using the SCU chapter in the Setting Up the Environment manual.

After you finish creating project directories and configuring the SCU files, you may want to use the Application Validator to create a baseline snapshot of the project directories. For more information on the Application Validator, including the command line options, refer to the Validating an Application section in the Mastering iFIX manual.

Notes on Security Configuration for Multiple Projects

When working with multiple projects for multiple users, make sure that you also enable global security paths in the Configuration dialog box of the Security Configuration. When you enable the global security paths option (Use these paths for all startup profiles), all iFIX user sessions on a computer share the same security configuration. To access the Security Configuration application, select Security from the Configure menu of the SCU. iFIX must be running to access this application.

Example of SCU Path Configuration

The tables that follow describe how to configure the Path Configuration dialog box in the SCU for each of the user types described in the Example of the iFIX Directory Structure for Multiple Projects section. In the following examples, the PRJ folder is the location where project-specific files are saved. This folder is created by you, the end-user, not during the iFIX install.

Once you create and configure an SCU file for each project, save the SCU to the Local path defined for this project.

IMPORTANT: For iFIX to work correctly when multiple users are configured, the SCU file for each project must be saved into the Local folder for that project.

Dev1

For the first developer, the project name is Dev1. This developer will have his own set of folders. None of the project folders in this example are shared. When you click OK from the Path Configuration dialog box, the SCU automatically creates the new project directories. Save the SCU file for this project to the Local path, C:\Program Files (x86)\Proficy\iFIX\PRJ\Dev1\Local, as defined in the Path Configuration dialog box. The following table lists the paths in this dialog box, properly configured for the Dev1 developer. Subfolders that specifically belong to the Dev1 developer appear in bold.

Path	Location
Base	C:\Program Files (x86)\Proficy\iFIX
Language	C:\Program Files (x86)\Proficy\iFIX\NLS
Project	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1**
Local	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\LOCAL**
Database	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\PDB**

Picture	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\PIC**
Application	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\APP**
Historical	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\HTR**
Historical Data	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\HTRDATA**
Alarms	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\ALM**
Master Recipe	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\RCM**
Control Recipe	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\RCC**
Alarm Areas (AAD)	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV1\PDB**

Dev2

For the second developer, the project name is Dev2, as shown in the following table. This developer will also have his own set of folders. None of the project folders in this example are shared. When you click OK from the Path Configuration dialog box, the SCU automatically creates the new project directories. Save the SCU file for this project to the Local path defined in the Path Configuration dialog box, the C:\Program Files (x86)\Proficy\iFIX\PRJ\Dev2\Local folder. The following table lists the paths in this dialog box, properly configured for the Dev2 developer. Subfolders that specifically belong to the Dev2 developer appear in bold.

Path	Location
Base	C:\Program Files (x86)\Proficy\iFIX
Language	C:\Program Files (x86)\Proficy\iFIX\NLS
Project	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2**
Local	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\LOCAL**
Database	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\PDB**
Picture	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\PIC**
Application	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\APP**
Historical	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\HTR**
Historical Data	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\HTRDATA**
Alarms	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\ALM**
Master Recipe	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\RCM**
Control Recipe	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\RCC**
Alarm Areas (AAD)	C:\Program Files (x86)\Proficy\iFIX**PRJ1\DEV2\PDB**

Supervisors

For all supervisor users, the project name is Supervisors, as shown in the following table. The supervisors in this example, have their own LOCAL, HTR, and ALM directories. They share the PDB, PIC, APP, HTRDATA, RCM, and RCC directories with other user types, such as Operators. When you click OK from the Path Configuration dialog box, the SCU automatically creates the new project directories. Save the SCU file for this project to the Local path defined in the Path Configuration dialog box, the C:\Program Files (x86)\Proficy\iFIX\PRJ\Supervisors\Local folder. The following table lists the paths in this dialog box, properly configured for the supervisors. Subfolders that specifically belong to the supervisors appear in bold.

Path	Location
Base	C:\Program Files (x86)\Proficy\iFIX
Language	C:\Program Files (x86)\Proficy\iFIX\NLS
Project	C:\Program Files (x86)\Proficy\iFIX**PRJ1\SUPERVISORS**
Local	C:\Program Files (x86)\Proficy\iFIX**PRJ1\SUPERVISORS\LOCAL**
Database	C:\Program Files (x86)\Proficy\iFIX\PDB

Picture	C:\Program Files (x86)\Proficy\iFIX\PIC
Application	C:\Program Files (x86)\Proficy\iFIX\APP
Historical	C:\Program Files (x86)\Proficy\iFIX**PRJ1\SUPERVISORS\HTR**
Historical Data	C:\Program Files (x86)\Proficy\iFIX\HTRDATA
Alarms	C:\Program Files (x86)\Proficy\iFIX**PRJ1\SUPERVISORS\ALM**
Master Recipe	C:\Program Files (x86)\Proficy\iFIX\RCM
Control Recipe	C:\Program Files (x86)\Proficy\iFIX\RCC
Alarm Areas (AAD)	C:\Program Files (x86)\Proficy\iFIX\PDB

Operators

For the all operator users, the project name is Operators, as shown in the following table. The operators in this example have their own LOCAL directory. They share the PDB, PIC, APP, HTR, HTRDATA, ALM, RCM, and RCC directories with other user types, such as Supervisors. When you click OK from the Path Configuration dialog box, the SCU automatically creates the new project directories. Save the SCU file for this project to the Local path defined in the Path Configuration dialog box, C:\Program Files (x86)\Proficy\iFIX\PRJ\Operators\Local folder. The following table lists the paths in this dialog box, properly configured for the Operators. Subfolders that specifically belong to the operator appear in bold.

Path	Location
Base	C:\Program Files (x86)\Proficy\iFIX
Language	C:\Program Files (x86)\Proficy\iFIX\NLS
Project	C:\Program Files (x86)\Proficy\iFIX**PRJ1\OPERATORS**
Local	C:\Program Files (x86)\Proficy\iFIX**PRJ1\OPERATORS\LOCAL**
Database	C:\Program Files (x86)\Proficy\iFIX\PDB
Picture	C:\Program Files (x86)\Proficy\iFIX\PIC
Application	C:\Program Files (x86)\Proficy\iFIX\APP
Historical	C:\Program Files (x86)\Proficy\iFIX\HTR
Historical Data	C:\Program Files (x86)\Proficy\iFIX\HTRDATA
Alarms	C:\Program Files (x86)\Proficy\iFIX\ALM
Master Recipe	C:\Program Files (x86)\Proficy\iFIX\RCM
Control Recipe	C:\Program Files (x86)\Proficy\iFIX\RCC
Alarm Areas (AAD)	C:\Program Files (x86)\Proficy\iFIX\PDB

Use the Startup Profile Manager to Define Profiles for Each User in a Project

Each project can have multiple users. User profiles are used to associate a user with the iFIX project they will use.

To define a user profile, you use the iFIX Startup Profile Manager. Each profile associates a Windows user name with a specific iFIX Project Configuration. The iFIX Project Configuration includes:

- SCU path and file name that you want the specified Windows user to use when starting iFIX.
- Node name that you want the specified Windows user to use when starting iFIX.
- Restrictions on whether the user can modify the Nodename or SCU fields in the iFIX Startup dialog box (Launch.exe).

iFIX must be running in order to use the Startup Profile Manager application to create startup profiles. To access the Startup Profile Manager, in the iFIX WorkSpace's system tree, double-click the Startup Profile Manager icon. The Startup Profile Manager can also be accessed from the Start menu by pointing to Programs, iFIX, Tools, and then Startup Profile Manager.

To configure your startup profiles, follow the steps in these sections:

- Configuring the Default Profile
- Adding Startup Profiles

Use the Application Validator to Take a Snapshot of Your Project Folders

Once you configure your iFIX projects, you can create a baseline of the iFIX files and folders plus other files and folders associated with your process in the Application Validator. You can then monitor which files and folders changed since you created the baseline.

For instance, if you ever run into problems with your iFIX configuration and want to go back to an original configuration, you can use the Application Validator to determine which files were changed or added, and then you can manually rollback to that configuration.

To access the Application Validator, double-click Application Validator icon in the system tree in the iFIX WorkSpace. You can also access the application by locating and running the AppValidator.exe file in the iFIX folder, which is the folder where you installed iFIX.

For detailed steps on how to create a baseline, refer to the General Overview of the Steps for Using the Application Validator section in the Mastering iFIX manual.

For more general information on the Application Validator, including the command line options, refer to the Validating an Application section in the Mastering iFIX manual.

Index

T

U

V