InTouch - Industrial Application Server Migration and Coexistence Planning Guide

Eduardo Ballina

Fernando Gonzalez

Invensys Systems, Inc.

May 2004

Table of Contents

1. White Paper Summary

This white paper was written to help existing InTouch® customers in planning and designing Industrial Application Server (IAS) applications that must work side-by-side with applications built with FactorySuite® 2000 or that must use existing applications as a starting point. Thorough planning is an important part of the migration process, whether you are ready to implement IAS today or in the future.

2. Benefits of Using the Industrial Application Server

Industrial Application Server was developed primarily to help current InTouch customers with the development, management, and maintenance of larger and more complex supervisory systems. As a result, the benefits of using the Industrial Application Server with InTouch include:

- Improved development productivity
- Remote system maintenance
- One single, global namespace for tag access
- Common system services for alarming, security, history, events, scripting, communications, and documentation
- Easier system maintenance and change management

Industrial Application Server incorporates many of the functions contained in a single node HMI application and simplifies them from one software entity to a series of componentized objects. This allows you to scale a system very effectively from a single computer to hundreds of nodes without a high development and maintenance burden. The Industrial Application Server is truly revolutionary and sets the standard for the industry.

3. Introduction

If you are a current FactorySuite user, you very likely already have a system in place that works to your satisfaction. However, you may be considering an expansion or addition to your site using the latest technology: FactorySuite A^{2}™ and the Industrial Application Server.

Extending an existing FactorySuite 2000 application can be approached in a number of ways:

- You may decide to implement the extension using FactorySuite A^{2} components and do so with a "classic" approach, but will prepare for future migration to the Industrial Application Server.
- You may choose to introduce Industrial Application Server in the new portion of the project and simply integrate to the existing system while keeping the HMI nodes "as is."
- You may elect to migrate your existing FactorySuite 2000 application to the Industrial Application Server.

This document provides basic recommendations for all three choices: co-existence, designing today for easy future migration, and migration to Industrial Application Server.

4. Basic Concepts

In order to better understand the possibilities, you should review certain basic concepts about traditional FactorySuite 2000 InTouch applications, as well as Industrial Application Server.

4.1. Connectivity

As always, as we at Wonderware® expand our options in connectivity, we continue to support protocols that our users have embraced for their existing applications. Industrial Application Server can access the very same device data leveraged by your existing InTouch nodes by using client ApplicationObjects.

- DDESuiteLinkClient object

  The DDESuiteLinkClient object provides IAS connectivity to DDE and SuiteLink™ data from sources such as I/O Servers, InTouch, InControl™, IndustrialSQL Server™, Microsoft Excel and any other source that supports DDE or SuiteLink. This object is hosted by an AppEngine running on an ApplicationObject Server node.

- InTouchProxy object

  The InTouchProxy object is an integration object of particular interest in a coexistence scenario. This is a specialized client object, similar to the DDESuiteLinkClient object. However, this proxy object provides the ability to browse the tagname dictionary of an existing InTouch application, so that you can select the tags to be exposed to the Industrial Application Server.

- OPCClient object

  If you are currently using OPC Servers along with OPCLink, you will want to use the Industrial Application Server OPCClient object. This client object provides IAS connectivity to OPC 2.0x DA Servers. This object is also hosted by an AppEngine running on an ApplicationObject Server node.

Note The newly introduced Message Exchange protocol is used for communications between IAS Platforms. It is not used for communications with data servers. Thus, it is not a replacement for DDE, SuiteLink, or OPC for device integration communications. Therefore, your existing device interfaces will work with any FactorySuite A^2 product.

4.2. Alarms

InTouch and InTouchView nodes that are part of an Industrial Application Server based system can access alarms from existing FactorySuite InTouch nodes (version 7.11 or higher), in addition to alarms coming from the Industrial Application Server itself. This is because the same distributed alarming system introduced with InTouch 7.11 is also part of Industrial Application Server.

Note Nodes running InTouch 7.1 or earlier must be upgraded to InTouch 7.11 or later in order to be able to interact with the new distributed alarm system.

- Alarm Logger™

  The Alarm Logger allows you to store alarms from alarm providers, which include both InTouch 7.11 nodes and the Industrial Application Server. You can configure alarm display objects on any visualization node in the plant to display all alarms.

4.3. History

Existing InTouch nodes can continue to use the InTouch historian and the InTouch historical trend object. However, IndustrialSQL Server presents new opportunities.

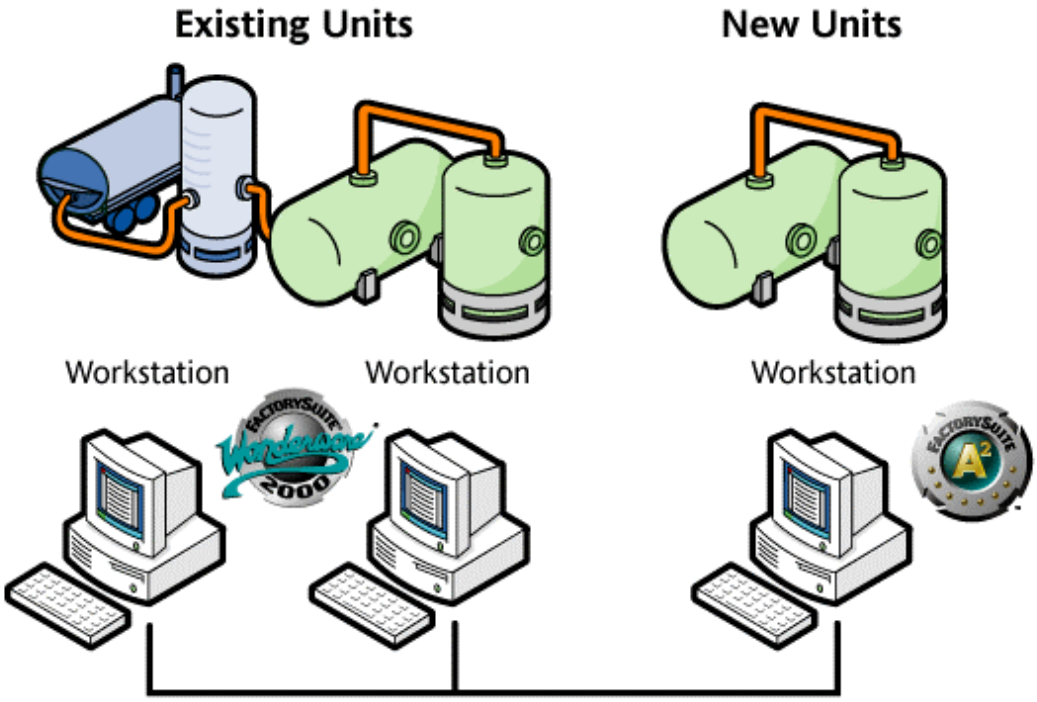
IndustrialSQL Server is the historian for Industrial Application Server. Data from your existing FactorySuite 2000 nodes can be collected and made available by the same IndustrialSQL Server that performs the historization work for an Industrial Application Server.

5. Coexistence with Industrial Application Server

This section provides information on how a new system based on Industrial Application Server can be integrated to an existing FactorySuite 2000 system. Two possible scenarios are discussed.

Scenario 1: The FactorySuite 2000 application remains unchanged.

In some cases, it may be desirable to upgrade an existing system to FactorySuite A^2 in order to leverage the benefits of Industrial Application Server. However, there will be cases where you may decide to keep the existing system "as is" for a number of reasons. For example, perhaps the system has already been certified, or the system already complies with all your requirements.

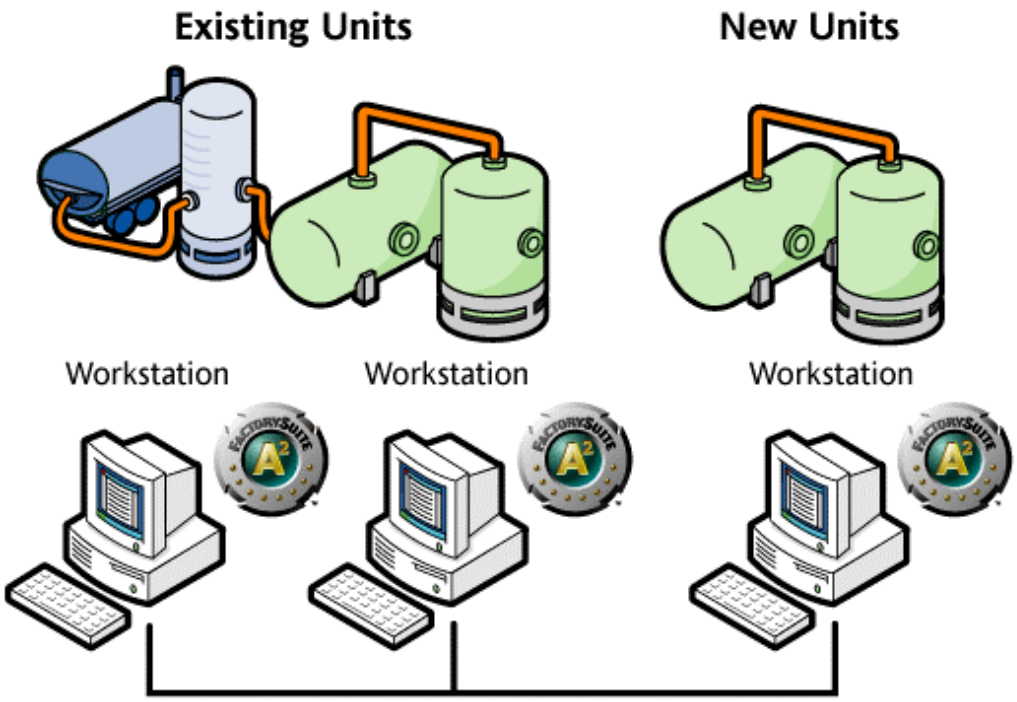


For this scenario:

- Use the corresponding InTouchProxy object and other client objects in the IAS plant model to connect to the existing data sources.
- Use Alarm Logger to capture alarms from both IAS and legacy systems.
- Use IndustrialSQL Server to collect data history from both legacy and new systems.

Scenario 2: FactorySuite 2000 nodes are upgraded to InTouch 8.0 or later, but not migrated to IAS.

You may be interested in upgrading the existing InTouch nodes to the latest version of InTouch (8.0 or later) without migrating the existing application to an IAS model. You want to bring data from the new IAS expansion to new windows in your existing InTouch nodes.



For this scenario:

- Upgrade existing InTouch nodes to InTouch 8.0 or later.
- If IAS data needs to be displayed at the legacy side, install a bootstrap and deploy a Platform at every existing InTouch node.

You can now create new InTouch windows that access data from the objects in the IAS Galaxy. You do not need to create new InTouch tags in order to access data from IAS. Data is sent to InTouch nodes via remote tags that reference the built-in "Galaxy" access name.

This configuration scenario allows you to:

- Keep all of your visualization nodes running the same version of InTouch.
- Browse the entire Galaxy namespace from any InTouch node.
- Use existing nodes to access data from the additional IAS nodes, as well as use new visualization nodes to get data from legacy ones.
- Leverage the Message Exchange protocol for communications across all nodes.
- Consolidate user accounts into a single security system.

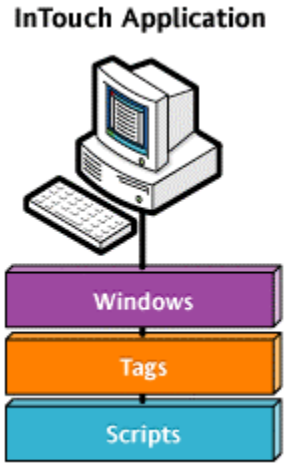
6. Designing Applications that are Easy to Migrate

If you are already implementing a solution with a FactorySuite "classic" approach, there are a number of recommended practices for you to consider. These recommendations apply to any InTouch system today, regardless of whether or not the application will be eventually migrated to an Industrial Application Server model. However, by following these suggestions, you will lay a foundation for an easy migration in the future.

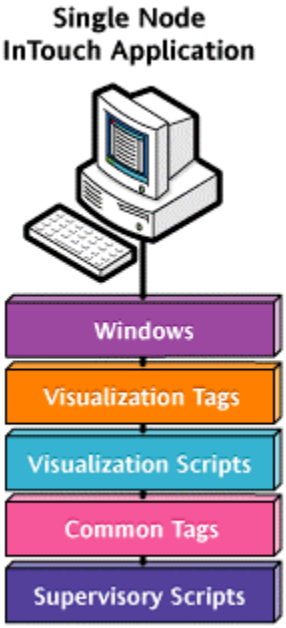
Before reviewing the list of recommendations, you should analyze the way "classic" applications are currently implemented.

6.1. Structure of a Typical "Classic" InTouch Application

A traditional single-node InTouch application consists primarily of a number of windows, scripts of various types, different types of tags, alarming, historization, and security. As illustrated in the following diagram, the three main building blocks of an application are tags, windows, and scripts. Historization and alarming are closely associated with tag definitions.



From a functionality point of view, you can make a distinction between tags and scripts used for visualization only and those used for supervisory control.



Visualization tags are tags used to perform graphic animations, window manipulation, and window navigation. Visualization tags are mostly memory tags. Indirect tags are a good example of tags used for visualization purposes.

Visualization scripts are scripts that do not manipulate process data but instead process information for the purpose of handling the graphical aspects of an InTouch application.

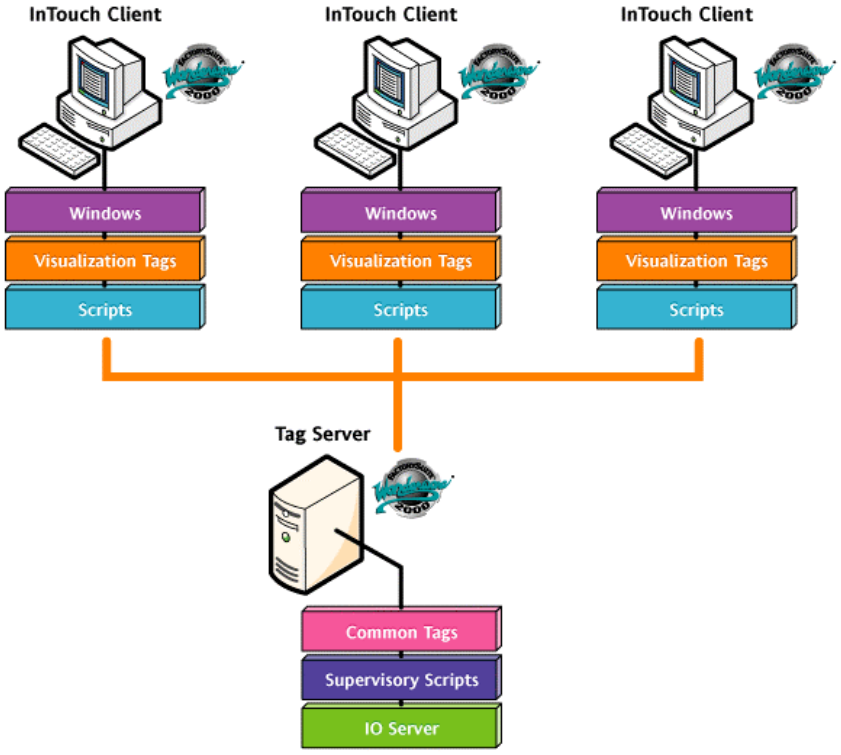
Common tags are primarily I/O tags that are used to connect the application to field devices and other sources of external data.

Supervisory scripts are scripts that perform supervisory control functions such as process related calculations, monitoring and managing communications, writing data to field devices, and so on.

6.1.1. Tag Server Architecture

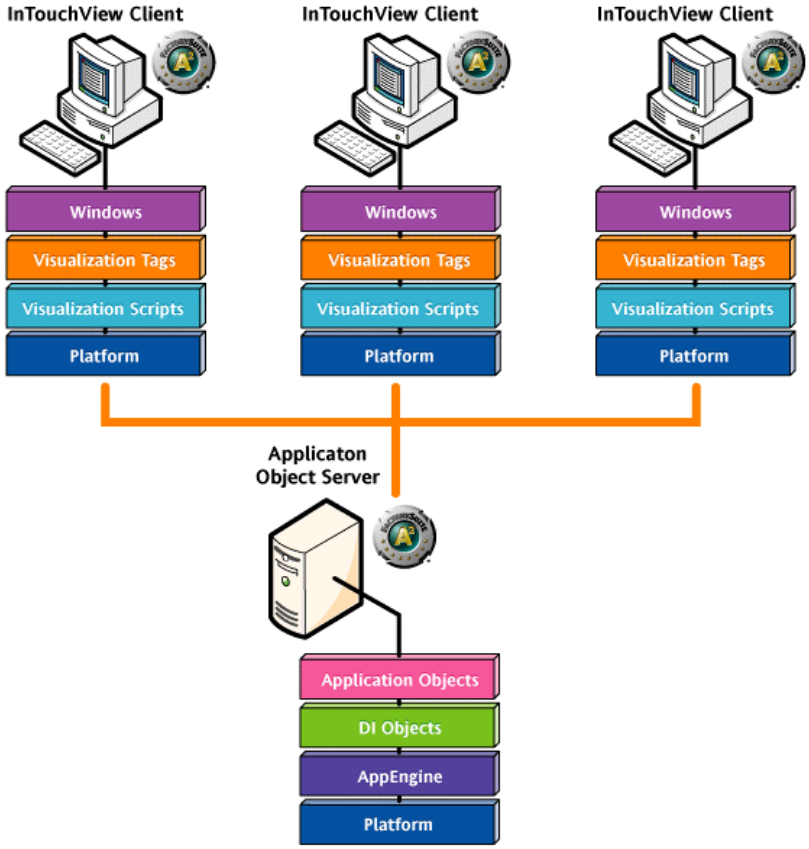
If you have implemented an InTouch based system in a distributed environment, you probably are using a tag server architecture. In this type of topology, visualization and supervisory control functions are split between the tag server and view nodes. A tag server node hosts an InTouch application with a tagname dictionary that holds common tags used by all InTouch applications in the system. This tag server application also runs supervisory control scripts, connects to the I/O data sources, and performs the historical logging and alarming functions. View nodes host an InTouch application that focuses primarily on the visualization aspect of the solution. Local tags and scripts in a view node are mostly those required for visualization only. Common tags, which reside at the tag server, are accessed by client nodes using remote tag references.

The following diagram illustrates the tag server topology and shows how this functionality is distributed in a system:



This tag server topology is a recommended architecture for an existing distributed system implemented using FactorySuite in a "classic" approach. *The separation of functionality will ease future migration to Industrial Application Server.*

In a multi-node IAS application, a typical topology would be as follows:



6.1.2. The Tagname Dictionary in a Classic Application

An important point to consider is the nature of the InTouch tagname dictionary. In InTouch applications, tags are part of a flat namespace. In the Industrial Application Server, the plant is modeled in a hierarchical way. The attributes of an object (which in a way are the equivalent of InTouch tags) maintain a relationship with the object they belong to. This provides a more true-to-life representation. Keeping all aspects of an entity grouped together (tags, scripts, alarming configuration, history configuration, and more) provides many advantages.

Following consistent naming conventions that allow easy identification of InTouch tags that relate to an entity (that is, to a valve, motor, pump, and so on), will go a long way in facilitating migration to Industrial Application Server in the future. You should first identify all devices, areas, and other logical groupings in your application and then apply naming conventions and standards established by your company in consistency with IAS naming. Not only is it recommended that the naming be consistent, but also that the naming reflects a hierarchy.

For example, an InTouch application uses the following InTouch tag:

HiddenValley_Well5_Pump_Pressure

This tag could later point to the following object attribute in an IAS model:

HiddenValley.Well5.Pump.Pressure

In the IAS model, HiddenValley is an Area, Well5 is a sub-Area, Pump is an object under that sub-Area, and Pressure is an attribute of the Pump object.

Use InTouch supertags. These are one of the tools you have in InTouch that can help you bring some structure to tagnames.

In summary, to create an InTouch application today that you can easily migrate to IAS in the future:

- Identify areas and devices in your application.
- Adopt and follow tag naming standards.
- Use supertags.
- Use tag server architecture and remote tag references.
- Split visualization and supervisory control functionality.

7. Migration to Industrial Application Server

Converting from an InTouch application to Industrial Application Server can provide significant benefits. However, you need to first determine whether the best approach would be to convert an existing InTouch application into an Industrial Application Server model or whether some form of coexistence is the right balance. Migration costs may be significant for some projects in terms of the engineering effort, but the benefits will outweigh the cost in most cases. Remember that the Industrial Application Server uses InTouch as the HMI, so complete conversion is not always necessary to gain the benefits needed.

Migration to IAS:

- Eliminates tag limitations.
- Provides a single namespace.
- Allows distribution of loads.
- Facilitates reuse of code from one application to another.
- Adds structure to projects (using the plant model).
- Simplifies maintenance and change propagation.
- Provides enhanced communications.
- Offers centralized security features.

Transitioning from an InTouch tag dictionary to IAS automation objects will help to optimize the new supervisory application. In the Industrial Application Server plant model, attributes are similar to tags in InTouch. IAS ApplicationObjects inherit their attributes from templates in an object-oriented environment, are grouped together based on containment that reflects a plant model, and have built-in security. Also, object attributes can be created or extended as necessary.

The second task you will need to perform when migrating to IAS is to identify and then remove functionality no longer required by an InTouch application (for example, certain types of scripts, tags, windows). InTouch programmers have multiple options when implementing functionality such as security, communications, diagnostics, or alarming in InTouch. One of the key goals of Application Server is to reduce the amount of work (scripting) previously required to support those features. Much of this functionality is now built into IAS or can be executed and managed by an ApplicationObject and could therefore be removed from an InTouch application. You will need to identify scripts, tags and animated graphical objects that perform those functions and determine which ones are candidates for removal.

After you have decided to migrate an existing application, the next question is how to approach the migration. You can create reusable templates, one of the strengths of IAS, to build an object-based plant model. Since this functionality is not available in InTouch, you may have created tag dictionaries, scripts, and windows in a way that was appropriate for each project. The next section describes some of the considerations to keep in mind when migrating an InTouch application.

7.1. Considerations when Migrating an InTouch Application

Regardless of how migration is performed, IAS programmers must handle functionality in IAS that is similar to that in existing InTouch applications. The process to convert an InTouch application to be used with Industrial Application Server is as follows:

- Security

  Security is pervasive in InTouch applications because it is accomplished at the link (UI) or window level. Nearly all scripts that handle security are no longer necessary in IAS and can be removed. There are some exceptions in which an application may still need to check for tags such as $AccessLevel, $Operator, or $OperatorName to perform some validation.

- Graphic Scripting

  Objects in InTouch windows can have a number of scripts to trigger Action, Animation, Visibility, or Color changes, among others. Also, it is very common to have On Show, While Showing, or On Hide scripts. It is necessary to convert all links to InTouch tags into remote references to instances of objects in the IAS model. This is easier if the naming structure of object attributes follows the InTouch tag naming structure, in which case the references (items) can easily be changed to remote tag references pointing to IAS.

- System Error Detection

  Error checking in InTouch is implemented via scripts. There are some system tags and tools (such as Status and IOStatus) that can aid in error detection, and functionality in the dot fields for I/O tags to monitor errors. IAS includes a vast number of built-in attributes that can easily be extended for alarms or configured for history.

- Device Integration

  InTouch interfaces with I/O sources via access names. IAS includes DeviceIntegration objects that interface with SuiteLink, DDE, or OPC-aware devices. IAS also includes a proxy object to connect to existing InTouch nodes.

- History

  InTouch history files are stored in a single directory specified during development. The format for history files includes an index file (.idx) for faster retrieval and a corresponding history file (.lgh). IAS uses IndustrialSQL Server as the historical archive. IndustrialSQL Server 8.0 SP1 or later includes a wizard to import InTouch history files from existing .lgh and .idx files into IndustrialSQL Server history blocks. For more information about the "InSQLITHist.exe" utility, see the IndustrialSQL Server documentation.

- Alarms

  InTouch and IAS rely on a common alarm sub-system based on a publish-subscribe model. Alarms are generated by providers such as InTouch nodes or Platforms in IAS; alarms are displayed by consumers such as the Distributed Alarm Object (DAO) or AlarmView ActiveX control grids inside InTouch. Since a DAO can query alarms from InTouch or IAS, converting an alarm window is a very simple process. If IAS alarms must be acknowledged, then the AlarmView ActiveX Control must be used.

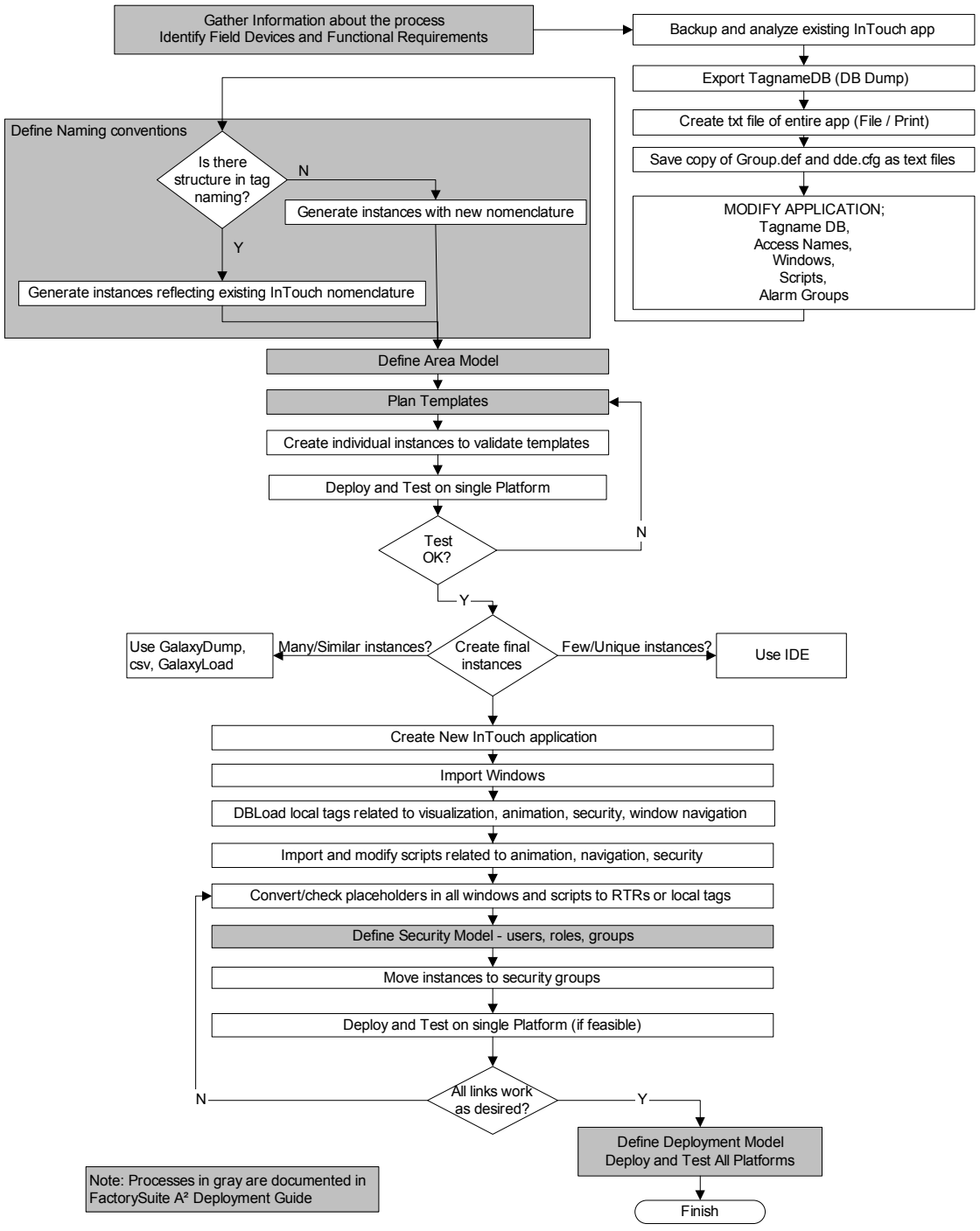
- SQL Access Manager, Recipe Manager, and SPC Pro

  These modules are popular options for InTouch applications. While IAS does not have base templates to replace them, it is possible to develop templates that perform similar operations. For example, the FactorySuite A² Demo CD includes a SQLDataAccess template. For more information on FactorySuite integration, see the *FactorySuite A² Deployment Guide*.

7.2. Workflow for Migrating an InTouch Application

The *FactorySuite A² Deployment Guide* provides a suggested workflow when planning a project with Industrial Application Server. The workflow diagram presented in this section extends that workflow to include the tasks that are required when migrating an existing InTouch application as part of the development of an Industrial Application Server project. All processes explained in the deployment guide are shown with a gray background in the flowchart; for more information, see the deployment guide.

Suggested Project Workflow for Migrating an Existing InTouch Application to FactorySuite A² With Industrial Application Server



7.2.1. Gather Information about the Process and Identify Field Devices and Functional Requirements

It is very important to become familiar with the existing InTouch application and understand what it does. To facilitate this, you can:

- Backup and analyze the existing InTouch application.
- Review any additional documentation available for the project.
- Talk with the person or group that originally developed the application.
- Talk with the operators that use the application. Nobody knows the application better than the operators that use it on a regular basis. If possible, have them walk you through all of the windows and procedures that are part of the supervisory system.

7.2.2. Export the Tagname Database

By using DBDump™ in the InTouch Application Manager, you can create a .csv file that includes all access names, alarm groups, and tagnames defined in the application. This file will serve as the starting point to build the Galaxy in IAS. You may be able to identify patterns in tag naming that may suggest potential ApplicationObjects, alarm groups that could become Areas, and access names that may indicate the type of DeviceIntegration Objects to use.

7.2.3. Create a .txt File of the Entire Application

The file/print option in WindowMaker™ (InTouch 8.0 or later) allows you to send the output to a text file instead of a printer. This file is very helpful when you are looking for uses of a particular script function or for any other search not related to the use of tagnames. Tagname searching is available in the Cross Reference tool in WindowMaker.

7.2.4. Save a Copy of Group.def and DDE.cfg as Text Files

These files are included in the directory where the InTouch application is stored. Group.def is a text file that contains the definition of all alarm groups in the application and their hierarchy. In many cases this hierarchy represents the Area model to be used in IAS or at least provides you with a good starting point.

The DDE.cfg file contains details about all access names defined for the InTouch application. Most of the information from this file is already available in the DB.csv file that is created when the Tagname dictionary is exported into a .csv file via DBDump.

7.2.5. Modify the Original InTouch Application

The original InTouch application may be modified for the new project with Industrial Application Server. The goal is to create a new InTouch application that reuses some of the scripts and windows from the original one; therefore, the original application should be modified by removing any unnecessary tagnames, access names, windows, scripts, and alarm groups.

Migrating the InTouch application to an IAS-based architecture will be easier if the original application was implemented as part of a client-server architecture, where one InTouch node (the tag server) has all I/O definitions and associated scripts and other InTouch nodes (the clients) have remote tag references to the server.

- Tagname DB

  Most, if not all, I/O tags will be removed from the .csv file because they will become part of the ApplicationObjects in IAS. Most memory tags, particularly those that handle animation, operator settings, window navigation, and security, will be reused in the InTouch application.

- Access Names

  Access names define connections from InTouch to external sources (DDE and SuiteLink servers). All access names must be removed from the application as most of them will be implemented as DeviceIntegration (DI) objects in IAS.

- Windows

  Delete any unnecessary windows. Review any window logic to determine which tags and scripts handle the navigation. This logic can be reused in the new InTouch application.

- Scripts

  It is very important to review all of the scripts in the original application and separate the functionality that will remain as part of the new InTouch application from the functionality that will be implemented in IAS.

  If scripts are embedded in windows, they will be more difficult to manage later. Only the scripts that are required to be on the window itself should be placed there; otherwise, move them to a common location and use quick functions for repeated tasks.

  When migrating scripts, determine what script functionality has been (or can be) replaced by the infrastructure provided by IAS. This will help you identify the tags and scripts that can be removed.

- Alarm Groups

  The Group.def file and the .csv files show the area definitions and are no longer needed in the new InTouch application for the IAS-based system.

7.2.6. Define Naming Conventions

The second step in the workflow is to define the naming conventions for templates, objects, and object attributes. Naming conventions should adhere to:

- Conventions that you use within your company.
- ArchestrA® naming restrictions. For information on allowed names and characters, see the IDE documentation.

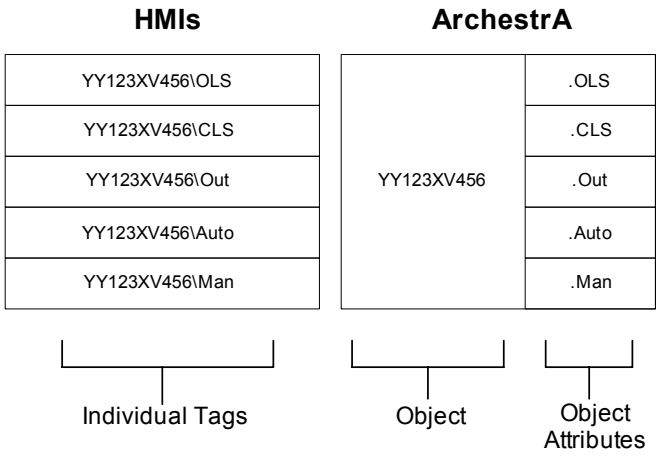
For example, you might have an instance tagname of:

YY123XV456

with the following attributes:

OLS, CLS, Out, Auto, Man

The following illustration shows how the naming convention in a traditional Human-Machine Interface (HMI) is different from the naming within ArchestrA:



For ArchestrA, references are created using this naming convention:

<objectname>.<attributename>

For example:

YY123XV456.OLS

7.2.7. Define the Area Model

The next step of the project workflow is to define the Area model. An Area is a logical grouping within your application that represents a portion of the layout of your plant. In a typical plant, you would define the following Areas: Receiving Area, Process Area, Packaging Area, and Dispatch Area. You will need to define and document all of the Areas of your plant that will be part of your application.

Each object will need to be assigned to a particular Area. When you install the Industrial Application Server, a single Area is created by default, called "unassigned." Unless you specify otherwise, all object instances will be assigned to this Area.

The following are a few tips for creating Areas:

- If you create all of your Areas first, you can easily assign an object instance to the correct Area; otherwise, you will have to move them out of the unassigned Area later.
- It is helpful to create a system Area to which you can assign instances of WinPlatform and AppEngine objects. WinPlatform and AppEngine objects are used to support communications for the application, and do not necessarily need to belong to a plant-related Area.
- You can group alarms according to Areas.
- Areas can be nested.

When building an Area hierarchy, keep in mind that the base Area that is assigned to a Platform determines how the underlying objects will be deployed. If a plant area (physical location) is going to contain two computers running AutomationObject Server Platforms, then two logical Areas will have to be created for the one physical plant area.

One approach for creating instances of an object is to create an instance for one Area at a time. If you use this approach, then mark the Area as the default, so that each instance is automatically assigned to the selected Area. Before you begin to create instances in another Area, change the default to the new Area.

A final consideration for constructing Areas is that the various Areas equate to alarm groups. It is at the Area level that alarm displays can easily be filtered. For more information on Areas, see the IDE documentation.

7.2.8. Plan Templates

The next step is to determine the templates that you will need. A template is an element that contains common configuration parameters for objects that are used multiple times within a project. Templates are instantiated to represent specific objects within the application. Both the templates and the instances created from them are called ApplicationObjects.

For example, you might need multiple instances of a valve within your application, so you would create a valve template that has all of the required properties. This allows you to define once, and reuse multiple times. If you change the template, the changes can be propagated to the instances. You can use simple drag-and-drop within the IDE to create instances from templates.

The Industrial Application Server is shipped with a number of pre-defined templates to help you create your application quickly and easily. Review these templates and determine if any of their functionally match the requirements of the devices on your list. If not, you can create (derive) new templates from the supplied UserDefined template.

For your project planning, document which existing template can be used for which objects, and what templates you will need to create yourself. For information on a particular object template, see the Help file for that object. A child template that you derive from a parent template can be highly customized. You can implement user-defined attributes (UDAs), scripting, and alarm and history extensions.

Note You can use the Galaxy Dump and Load Utility to create a .csv file, which you can then modify using a text editor and load back into the Galaxy Repository. This allows you to make bulk edits to the configuration quickly and easily. For more information on templates and template derivation, see the IDE documentation.

7.2.9. Create a New InTouch Application

After you have modified the original InTouch application, create a new application and import the changes. By importing the windows and scripts, you get a chance to review all of the visualization links and scripts that remained after you modified the original application.

- Import windows.

  Bring all of the windows into the new application; the placeholders will be converted later.

- Use DBLoad™ to load local tags related to visualization, animation, security, and window navigation.

  Before performing the load, set the .csv file for the Test mode (:MODE=TEST) in order to validate the file. For more information on DBLoad, see the *InTouch User's Guide*.

- Import and modify scripts related to animation, navigation, and security.

  After you have imported all of the scripts, convert them to local tags. Revise security related scripts to include the functionality available in InTouch 8.0 or later. For example, rather than checking for *$AccessLevel*, the *IsAssignedRole()* function should be used to determine what the current user is allowed to do.

- Convert/check all placeholders in all windows and scripts to remote tag references or local tags.

  At this point, enabling the object browser to point to the Galaxy Repository will simplify the process of validating all the links. For the browser to work, the IDE must be installed and a Platform deployed to the development node where WindowMaker is running.

In a client/server InTouch application, windows are already using links to remote tag references, so migrating this type of system is easier. For this type of application, you first need to replace the original access name used by the window links with the built-in "Galaxy" access name that connects to the Galaxy via the local Platform and Message Exchange. In a non-client/server application, windows may include links to tags that no longer exist in the new application; if so, modify these links to use remote tag references pointing to object attributes in IAS.

Look for links that were referencing dot fields in the original application; those dot fields must be modified to point to the proper object attribute or attribute property in IAS. Also, make sure to use the proper syntax for the default attribute (that is, .pv or .value) and use the #VString whenever possible. For more information, see the *FactorySuite A² Deployment Guide*.

7.2.10. Define the Security Model - Users, Roles, Groups

The next step is to define the security model. The following basic concepts are important for understanding the ArchestrA security model:

- Users

  A user is each individual person that will be using the system. For example, John Smith and Peter Perez.

- Roles

  Roles define groups of users within the security system. Roles usually reflect the type of work performed by different groups within your factory environment. For example, Operators and Technicians.

- Permissions

  Permissions determine what users are allowed to do within the system. For example, Operate, Tune, and Configure.

- Security Groups

  A security group is a logical Area to which you want to assign users and/or roles. Security groups typically map on to Areas and reflect a physical location of your plant. For example, you might want to assign Technicians to the Line_1 security group, but not to the Line_2 security group.

Define and document the users, roles, permissions, and security groups that you will need in order to implement security for your factory environment. You can use users and roles that have already been defined within the operating system security, or you can define them within the IDE. You can also use a mixture of both types. Using operating system users and roles facilitates deployment and makes future maintenance easier. You will also need to determine the security settings for writeable attributes of objects. The security options for writeable attributes consist of: Read Only, Operate, Tune, Secured Write, Verified Write, and Configure.

Review your functional worksheet that lists the objects (and their attributes) that you plan to create and document the setting for each one. For example, you might have an input attribute for a valve that you want to be read-only.

The basic steps for setting up security within the IDE are:

1. Configure the attribute security for objects. You will need to do this at the template level.
2. Create security groups.
3. Create roles and assign them to security groups.
4. Select permissions and grant them to roles.
5. Define users and assign them to roles.

For more information on security and how to configure it, see the IDE documentation.

7.2.11. Move Instances to Security Groups

For more information, see the ArchestrA IDE documentation.

7.2.12. Deploy and Test in Single/Staging Platform (if feasible)

For more information, see the *FactorySuite A² Deployment Guide*.

7.2.13. Define the Deployment Model and Deploy and Test All Platforms

One of the strengths of ArchestrA is the distributed environment in which it runs. Within ArchestrA, you can deploy objects to various computers on the network. You will need to develop a model that specifies where you will deploy certain objects. When you actually perform the deployment, the objects will be deployed and executed on the target computers.

Each computer that participates in your ArchestrA network will need to have a WinPlatform object, AppEngine object, and Area object deployed to it, at a minimum, in addition to ApplicationObjects.

The objects that you should deploy on particular Platforms and engines are a function of the "load" of the objects. The load is based on the number of I/O points used, the number of user-defined attributes (UDAs), and so on.

After you deploy objects, you could use the Object Viewer application to check communications between nodes and determine if the system is running optimally. You may find that you have a node that is executing more objects than it can easily handle, and you will need to deploy one or more objects to another computer.

For more information on deployment, see the IDE documentation.

7.3. Leveraging InTouch 9.0 SmartSymbols

SmartSymbol technology brings the same type of object-based development productivity to InTouch graphics development as the Industrial Application Server does for the control system services. SmartSymbols also simplify and synchronize the development of InTouch graphics with IAS templates. This reduces the overall engineering work by reducing the number of tags and scripts in InTouch. It is also possible to dynamically set graphics references to instances at run time. Attributes in IAS can also be browsed from the SmartSymbols Editor and IAS object instances are generated when a smart symbol is replicated.

When migrating from an InTouch-only application to IAS, consider the advantages of migrating the InTouch graphics to SmartSymbols. You can do this in WindowMaker by selecting the graphic elements that you want to make into a SmartSymbol, making them into an InTouch cell, and then generating a SmartSymbol from the cell. The SmartSymbol Manager allows tag references in a SmartSymbol to be linked to attributes in an IAS object template. It also allows an object template instance to be automatically generated from a SmartSymbol and for the object attribute references to be switched at run time using the IOSetRemoteReference script function.

By converting InTouch tags to IAS ApplicationObjects, with associated attributes, and InTouch graphic elements into SmartSymbols, you can fully leverage Wonderware's ArchestrA technology today, and as FactorySuite A^2 evolves.

8. References

The following documents provide additional information on how to implement the Industrial Application Server. They can be located on the Wonderware Tech Support Site at http://www.wonderware.com/support/

- *FactorySuite A^2 Deployment Guide*
- *InTouch User's Guide*
- *ArchestrA IDE User's Guide*
- *InTouch 9.0 New Features User's Guide*

9. Summary

The Wonderware Industrial Application Server leverages the InTouch HMI as the visualization component of the supervisory system. As a result, there are several options for existing InTouch customer to integrate their current applications with the IAS and take advantage of the power this ArchestrA architecture provides.