# AllFusion® Plex

## Getting Started

r5.5 SP1

Computer Associates®

MAN03115041E
F15836-1E

# Contents

# Chapter 3: Develop Your First Application in 20 Minutes

# Chapter 7: Open Database Components

# Chapter 8: Windows Server Components

# Chapter 9: Java Components

Chapter

1

# Introduction

AllFusion® Plex is an Architected RAD tool that uses software patterns, models, and generators to accelerate the design and maintenance of business applications for the Java, J2EE, Windows, and iSeries platforms. Software patterns enable developers to employ emerging Internet technologies using one set of skills, one set of techniques, and one environment.

## Getting Started Quickly

To get started quickly, follow the basic installation steps described in the chapter "Installing AllFusion Plex." You will then be able to run through the introductory tutorial in the chapter "Develop Your First Application in 20 Minutes."

## Enabling Rapid Adoption and Integration

AllFusion Plex enables the adoption of new technology with minimal risk and significantly reduced cost. Immediately usable software patterns allow organizations to extend AllFusion Plex applications to dynamic web technologies such as wireless business and transactional XML-based applications using only a single set of skills.

The major benefits of the product include:

- **Increased productivity** by helping application architects to capitalize on existing investments. AllFusion Plex enables organizations to leverage knowledge and prior experience for reuse. Predefined patterns are customized to meet precise business needs. They provide specific model-based application building blocks that can be implemented and reused throughout applications to speed up the design and accuracy of the entire system.
- **Enhanced communication** by enabling the development of applications that focus on business requirements rather than technical specifications. Ongoing communication between business decision-makers and technical departments is enhanced through IT and business alignment.
- **Increased agility to meet market demands** by allowing organizations to easily integrate existing business data and applications with emerging Internet technologies to meet current and future business objectives.

## Distinctive Features

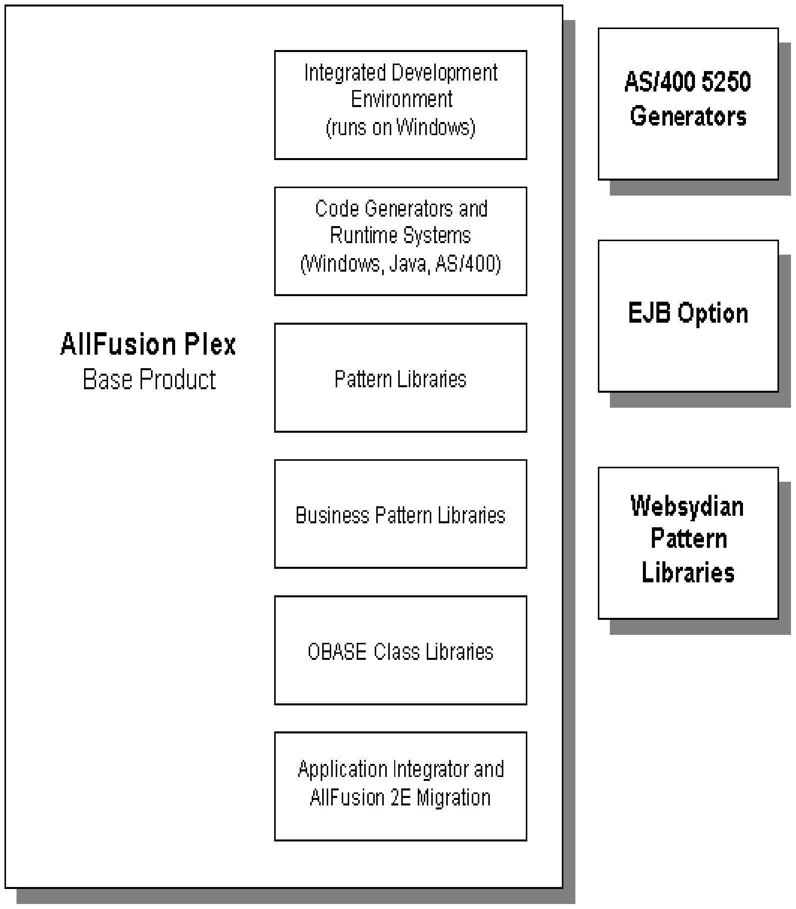
The distinctive features of the product include:

- **Patterns.** Patterns are reusable *building blocks* that provide solutions for many of the common challenges faced by application developers. AllFusion Plex ships with libraries containing hundreds of patterns. You can also build your own patterns and purchase patterns from third-party vendors.
- **Model-Based Development.** Applications built with AllFusion Plex are designed at the business level to ensure accuracy and consistency. From a single design, applications can be generated for a variety of platforms and databases. Application maintenance is performed at the business level, not the code level. This ensures that IT and business management are in sync, and that changes to business needs are reflected in applications faster.

- **100% Code Generation.** AllFusion Plex generates 100% of the code needed for business applications, including database objects as well as program code. The quality of the code generated has been consistently shown to reduce costs and coding errors. AllFusion Plex developers are business architects, not coders.
- **J2EE Support.** AllFusion Plex lets you deliver Java applications using Enterprise JavaBeans (EJBs) for the Java 2 Enterprise Edition (J2EE) platform. You can deploy these EJBs to a number of J2EE-compliant application servers, including BEA WebLogic, IBM WebSphere, and JBoss.
- **Java Client and Server Support.** Java applications created with AllFusion Plex support integration with UNIX, Sun Solaris, and Linux environments.
- **Business Integration.** Application integrators that encapsulate data from programs such as AllFusion® 2E are included with AllFusion Plex and can be easily applied. Native Java code from AllFusion Plex models can be extended to platforms such as UNIX or Linux. Software patterns can be incorporated, which extends applications to HTML, XML, wireless, and portal.

## Product Components

The following diagram shows the main components of AllFusion Plex.


AllFusion Plex
Base Product
Integrated Development Environment (runs on Windows)
Code Generators and Runtime Systems (Windows, Java, AS/400)
Pattern Libraries
Business Pattern Libraries
OBASE Class Libraries
Application Integrator and AllFusion 2E Migration
AS/400 5250 Generators
EJB Option
Websydian Pattern Libraries

AllFusion Plex includes:

- A Windows-based visual IDE, complete with GUI screen designers, a language-neutral business logic editor, diagrammer, and impact analysis tools.
- A multi-developer repository with built-in configuration management for storing design models across multiple versions, languages, and platforms.
- Code generators that automatically create 100% of the Java, C++, HTML, RPG, or SQL code required for implementing applications for the Web, wireless, Windows, Java, and iSeries platforms. AllFusion Plex generates the entire application, including HTML and GUI clients, 5250 host screens, server programs, and database objects.

AllFusion Plex minimizes the need for platform expertise, which insulates developers from technology changes and lets them focus on solving business problems rather than technical problems.

Applications generated by AllFusion Plex support a rich set of industry standards and technologies including XML, J2EE, J2SE, WAP, Web Services, SQL Server, Oracle, Ingres, and Linux.

## Supported Platforms

AllFusion Plex creates native, n-tier code for the following platforms:

- Java Server—Java servers with JDBC data access are supported across multiple platforms including Linux, Solaris, and OS/400. EJB Option for J2EE support.
- Windows Server—Windows servers are fully Back Office-compliant with Win32 C++ clients. They provide support for Oracle (through OCI) and SQL Server (through ODBC). They also include extensive COM integration support.

- Rich GUI Client—Rich GUI client supports both Win32 and Java (Swing) platforms and ActiveX and JavaBean GUI components. It supports access to all server platforms, plus stand-alone client data access through ODBC or JDBC to Ingres and other industry standard databases.
- iSeries Server—iSeries Servers are supported with RPG IV or RPG III code generation and DDS access to DB2/400.
- iSeries 5250—iSeries 5250 with RPG IV or RPG III code generation and DDS access to DB2/400. It includes drag-and-drop 5250 screen designer.
- HTML Web Client—HTML web clients through AllFusion Plex pattern libraries.
- XML document-based Web Service—XML document-based Web Services support through AllFusion Plex pattern libraries.

## For More Information

After reading this *Getting Started*, you can refer to the numerous resources available to you for additional information. In particular, there are two tutorials that take you step-by-step through the creation of a simple application:

- Tutorial for Windows
- Tutorial for AS/400 5250

AllFusion Plex offers detailed online help that provides comprehensive information and detailed instructions about how to use AllFusion Plex.

Chapter

2

# Installing AllFusion Plex

Review the readme file (Readme.html) for the minimum hardware and software requirements for developing applications with AllFusion Plex.

## Installation Checklist

Installation requirements for AllFusion Plex vary depending on your target development platforms and include third-party products that are not supplied with the product. For detailed version information and other system requirements, see the readme.

**Note:** Compatibilities are subject to change. For the latest information on AllFusion Plex compatibility with third-party products, check the support web site http://ca.com/supportconnect/, and select AllFusion Plex from the Product List drop-down menu.

Use the following table to determine which components you need to install and where to find the corresponding installation instructions.

| Component | Description | See |
|---|---|---|
| **Base Product** | | |
| AllFusion Plex base product | Includes the AllFusion Plex development environment, help files, sample models and tutorial models. | Installing the Base Product in this chapter |
| CA License Key | License keys are required to use AllFusion Plex and its optional components. The optional components are: ■ EJB Option ■ AS/400 5250 generator ■ Websydian pattern libraries | Entering the CA License Key in this chapter |
| **Pattern and Class Libraries** | | |
| Pattern libraries | The latest pattern libraries, recommended for most application development with AllFusion Plex. | "Pattern Libraries" chapter in this guide |
| Class libraries (OBASE) | The first set of pattern libraries developed for AllFusion Plex; not recommended for new development projects, except for the AS/400 5250 platform. | classlibs.hlp help file |
| **Tools and Utilities** | | |
| Application Integrator | Used for reverse engineering information from a variety of data sources into AllFusion Plex. | Installing the Application Integrator in the "Application Integrator" chapter in this guide |

| Component | Description | See |
|---|---|---|
| AllFusion 2E data migration | Used for importing schema from AllFusion 2E data models into AllFusion Plex. | AllFusion 2E Data Migration Requirements in the "AllFusion 2E Data Migration" chapter in this guide |
| Microsoft Word Viewer | Used for viewing and printing generated Windows reports. | Specifying your Print Program in the online help's *User Guide* (see the chapter "Designing and Printing Reports") |
| Adobe Acrobat | Used for viewing the online versions of the tutorial guides. | Adobe Acrobat Reader |
| **Platform-Specific Components** | | |
| Windows client | ■ Microsoft Visual Studio 2003 .NET.<br>The stand-alone version of Visual Studio 2003 .NET C++ is not recommended because it does not support code optimization features used in AllFusion Plex C++ builds.<br>■ A deployment platform of Windows XP or Windows 2000 is recommended. | Microsoft Visual C++ Compiler |
| Windows server (NT/BackOffice generator) | ■ AllFusion Plex Windows Application Server.<br>■ Microsoft Visual Studio 2003 .NET.<br>■ DBMS (Microsoft SQL Server, Oracle).<br>■ Windows 2000 or Windows Server 2003. For additional information on Windows Server | Installing the Windows Server Components in the "Windows Server Components" chapter in this guide |

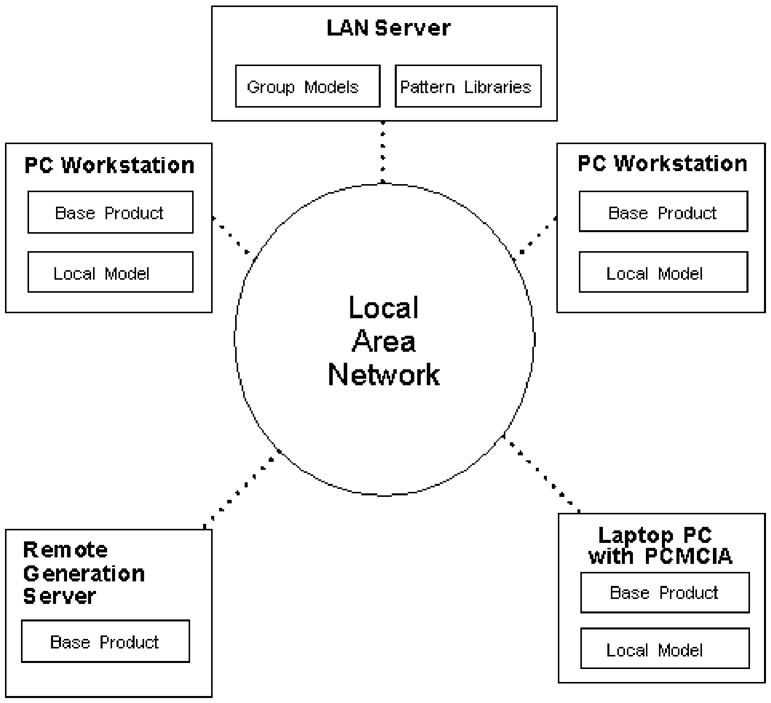
| Component | Description | See |
|---|---|---|
| | 2003 support, see the readme. | |
| iSeries (AS/400) (AS/400 5250 and AS/400 client/server generators) | ■ AllFusion Plex AS/400 libraries.<br>■ OS/400, or i5/OS. For support version information, see the readme.<br>■ RPG IV or RPG III compiler. | "AS/400 Components" chapter in this guide |
| Java client | ■ AllFusion Plex Java components.<br>■ Sun J2SE SDK 5.0.<br>Other versions of the JDK may work with AllFusion Plex, but they have not been fully tested.<br>■ NMAKE.EXE (Microsoft Program Maintenance Utility).<br>Optional. Used for local Java builds. For download information, see readme. | Installing AllFusion Plex for Java Components in the "Java Components" chapter in this guide |
| Java server | ■ AllFusion Plex Java components.<br>■ Sun J2SE SDK (or equivalent). See Java client for version information.<br>■ TCP/IP network connection.<br>■ Third-party DBMS such as Ingres, Oracle, or other third-party DBMS and JDBC driver.<br>For AS/400 Java server development:<br>■ PLEXJVA550 library.<br>■ IBM Client Access (or equivalent).<br>■ IBM AS/400 Toolbox for Java. | Installing AllFusion Plex for Java Components in the "Java Components" chapter in this guide |

| Component | Description | See |
|---|---|---|
| EJB Option | ▪ J2EE Application Server such as WebLogic, JBOSS, or WebSphere. The J2EE SDK includes an Application Server (the Reference Implementation) that is useful for testing purposes.<br>▪ Advantage Joe 4.0 for automated EJB deployment from AllFusion Plex. A copy of Advantage Joe 4.0 is included with the license for the EJB Option.<br>▪ Java 2 Enterprise Edition SDK 1.4. Note that this is also a requirement for Advantage Joe. We recommend following the installation steps for Advantage Joe 4.0 to download and install the J2EE SDK.<br>▪ J2SE SDK. This is same as the requirement for Java development.<br>▪ JDBC driver and DBMS. This is the same as the existing requirement for AllFusion Plex Java development. | "EJB Option" in the online help Java Platform Guide |
| Open Database (ODBC) | Requires an ODBC driver and DBMS. | "ODBC Components" chapter in this guide |

For detailed instructions about upgrading from earlier versions, see the *Release Summary*.

## Typical Configuration

You can configure AllFusion Plex in various ways. The following diagram shows a typical placement of AllFusion Plex components in a LAN environment:



Your group models and pattern libraries reside on the LAN server. The base product and the local models reside on the developer workstations. They may also reside on a laptop with a connection to the network. You only need to connect to the network when you are working on the group model.

You have the option of sending your local model to be generated and built by another computer on the network (a *remote generation server*). This lets you continue working on the local model while the generate-and-build process is running. Instructions for setting up a remote generation server are provided in the online help.

## Installing the Base Product

To install the AllFusion Plex base product on a development workstation, follow these steps:

1. Insert the AllFusion Plex product CD into a CD-ROM drive. The AllFusion Plex Installation CD window appears.
2. Click the Read button to read AllFusion Plex documentation, including the readme, and review the installation and upgrade information. (The readme file contains additional, late-breaking information not covered in this guide.)
3. Click the Install button to begin the installation.

   Initiate one of the following installations from the displayed menu:

| **Button** | **Function** |
|---|---|
| AllFusion Plex | Installs the AllFusion Plex base product. |
| AllFusion Plex Windows Deployment Server Components | Installs only the AllFusion Plex Windows Application Server components necessary to deploy your application on a Windows server. This option excludes build tools required for development. To install the build tools, run a custom installation of the base product. |
| Websydian Web Libraries | Installs third-party libraries to enable your AllFusion Plex applications for the Web. |
| Word Viewer | Installs a version of Word Viewer, which you can use to view and print AllFusion Plex run-time reports.<br><br>**Note:** If you have Microsoft Office installed, there is no need to install Word Viewer. |
| Java JDK 5.0 Update 3 | Downloads Java Development Kit (JDK) 5.0, which is the Sun Java Development Kit. It is a mandatory component for Java Development within AllFusion Plex. It is not required for Windows C++ or AS/400 RPG development. |

4. The InstallShield Wizard window appears.
5. Click Next.

6. At the License Agreement window, read the License Agreement and click Accept if you accept the terms of the agreement.

   **Note:** You must scroll to the end of the License Agreement to activate the Accept button.

7. Enter your name and the name of your company.

8. Specify a destination location. You can either accept the default, C:\Program Files\CA\AllFusion Plex r5.5, or you can click Browse to install to a different directory.

   **Notes:**

   - The installation program prevents you from installing this version into the same directory as the previous version of AllFusion Plex. Instead, you should install into a different directory or uninstall the previous version.
   - In addition to the specified destination directory, AllFusion Plex also installs a copy of Sun's JRE in the Program Files\CA\SharedComponents\Jre directory.

9. Specify which type of setup you want to install:

   - A *Typical* configuration installs all the program files, including help files, tutorial files, sample files, and pattern libraries. With a Typical install, the OBASE class libraries are not installed by default; to install them, select Custom Install.
   - A *Compact* configuration saves disk space by installing only the program files and the pattern libraries, but not the help files, tutorial files, or sample files.
   - A *Custom* configuration lets you specify which product components to install.

10. If you selected Custom, you can now select the components to install. The default settings for the Custom installation are the same as for a Typical installation, but you can also select Class Libraries, Application Integrator, and AllFusion Plex Windows Application Server.

11. Select which language you want to use as the default for generated Windows clients. You can change this setting later. For details, see the online help topic Specifying the National Language Library.
12. Click Back to return to any dialog in which you want to change the settings, or click Next to begin the installation.
13. Review your settings before copying files.
14. Click Finish when the installation is complete.
15. If prompted, select whether to restart your computer now or later and click OK.

**Note:** The AllFusion Plex installation creates an error log file, PlexInstall.log, and places it at the root of the C drive. The Application Server Deployment installation creates an error log file, PlexInstall.log.

## Silent Installation

To install the product in silent mode, use the /s command line switch on the setup.exe command. For example, from the command line enter:

```
d:\Plex550setup\setup.exe /s
```

where *d:*\ is the drive letter for your CD-ROM. This performs a full installation of the AllFusion Plex product to the default installation directory.

## System Path Modifications

The AllFusion Plex base product installation automatically makes changes to your system path. Specifically, the AllFusion Plex Bin directory (typically, C:\Program Files\CA\AllFusion\AllFusion Plex 5.5\Bin) is added to the system path. You can view or change the system path by double-clicking the System icon in the Control Panel and altering the environment settings.

## Adobe Acrobat Reader

To view the online versions of the tutorials, you need to install Adobe Acrobat Reader. To install Acrobat Reader, follow these steps:

1. On the AllFusion Plex Installation CD window, click Documents, Install Adobe Acrobat.
2. Follow the instructions displayed on the dialog.

## Java Development Kit

To develop applications using Java within AllFusion Plex, you must install the J2SE Development Kit (version 5.0). Use the following procedure as a guide for the installation process:

1. On the AllFusion Plex Installation CD window, click Install to display more installation options.
2. Click the Download JDK 5.0 Update 3 button.
3. Follow the instructions on the web site to download and install JDK 5.0 Update 3.

**Note:** If you install the JDK component after installing AllFusion Plex, you must perform any Java-dependent AllFusion Plex configuration changes manually.

## Entering the CA License Key

To complete the licensing of AllFusion Plex, place the information in the Execution Key from the ALP Key Certificate in the ca.olf file on the machine running AllFusion Plex. If you have access to the World Wide Web, you can get your updated ca.olf file by going to http://ca.com/supportconnect/. Otherwise, follow the directions in this section.

Within the ca_lic folder (usually located in Program Files\CA\SharedComponents) search for a file named ca.olf. If the ca.olf file does not exist, use a text editor to create a file named ca.olf, copy all of the information from the Execution Key into the file, and save it in the ca_lic directory.

If the ca.olf file does exist, open it using a text editor of your choice and make the following edits:

1. Replace all lines beginning with ID_ with the ID_ lines indicated in the Execution Key.
2. Go to the bottom of the file and immediately following any existing FEATURE lines add the FEATURE line from the Execution Key.

   **Note:** Do not remove any existing FEATURE lines.
3. Save the edited ca.olf file in the ca_lic folder.

**Note:** The FEATURE line can wrap to a second line on the certificate, but it must be entered on a single line with no carriage return in the ca.olf file.

## Frequently Asked Questions on CA Licensing (ALP)

**Q:** What is ALP?

**A:** ALP is the Automated License Program, the newest phase of CA licensing and order fulfillment. Rather than requiring clients to use a client-side application to identify hardware eligible to run CA software and request/create the appropriate license file, ALP products are shipped with a printed certificate representing their license file (based on the product and hardware information recorded in our license database), and are also given the opportunity to get their license keys electronically using our support web site.

**Q:** How can I get my ALP license keys online?

**A:** You can get ALP keys from http://ca.com/supportconnect. To log in to the secure area of the web site, you need a valid user ID and password.

To get ALP keys, perform the following steps:

1. Log in using your user ID and password.
2. Click the License Keys link under Downloads on the left pane.
3. Click the ALP keys link under What's New to view your ALP keys.

You can also install your ALP keys directly to your local machine by clicking the Install button.

**Q:** Why does CA licensing sometimes install into the Program Files folder and not C:\CA_LIC as it always has?

**A:** The latest CA licensing installation conforms to the CA installation standard by installing in the C:\Program Files\CA\SharedComponents\CA_LIC folder if no previous C:\CA_LIC licensing folder is found on a machine. If, however, a C:\CA_LIC folder is found on the machine, the new licensing will continue installing into that folder. Licensing packages that follow the installation standard are versions 1.52 and higher.

The following issue may occur when downgrading licensing:

If a 1.52 or higher version of licensing is installed on a machine with no previous licensing installed on it, then an earlier version of licensing is installed (downgrade of licensing), the earlier version automatically installs into the C:\CA_LIC folder, which creates two CA_LIC folders (C:\CA_LIC and C:\Program Files\CA\SharedComponents\CA_LIC). To fix the problem, simply install the latest posted version of licensing from the Total License Care (TLC) support site. This merges the two folders into C:\Program Files\CA\SharedComponents\CA_LIC. Over time, C:\Program Files\CA\SharedComponents\CA_LIC becomes the folder where licensing is managed.

**Q:** How do I use the mergeolf utility?

**A:** Syntax: MERGEOLF -n <new_olf> -c <current_olf> -o <output_olf> -b <backup_olf> -d

Where:

| | |
|---|---|
| -n: | Is the name of the new OLF file to merge |
| -c: | Is the name of the current OLF file to merge<br>Default: ca.olf |
| -o: | Is the name of the new OLF file to create<br>Default: ca.olf |
| -b: | Is the name of the backup of the current OLF file<br>Default: ca.bak |
| -d: | Enables debugging (mergeolf.log) |

Example:

```
MERGEOLF -n ca.no1 -c c:\ca_lic\ca.olf -o c:\ca_lic\ca.olf -b
c:\ca_lic\ca.old
```

**Q:** What is the correct format of an ALP key?

**A:** Following is an example of an ALP Execution Key:

```
ID_1 3991FFEF "Customer Name"      Technical Contact Name (As
recorded in our database)
ID_2 E4A19DB8 "Company Name"      Company Name
ID_3 C2F2E617 "00000000"           CA Site ID Number
ID_4 9550AD9 "email@domain.com"    Technical Contact Email (As
recorded in our database)
SERVER CAI 7152
DAEMON CAI_lic_d/usr/bin
FEATURE 0 CAI_lic_d 1.000 20-APR-2000 0 0CA4D081978DD1BD5C76
"(MODEL) (COMPONENT)" ANY # 0000000-000
```

The actual encrypted Execution Key is line 7, which is referred to as the FEATURE line, the ID and Server and Daemon lines are header lines that should only appear once in the file, at the beginning. Additional instances of these lines can invalidate your license file.

If you are having problems with your ALP key, verify that:

- There are no carriage returns interrupting the FEATURE lines in the ca.olf file.
- There are no blank lines in between the FEATURE lines in the ca.olf file.
- The ID lines of the newly generated ALP Execution Key replace any pre-existing ID lines in the olf.ca file.
- The name of the OLF is ca.olf and does not have an extra extension in the name; for example, ca.olf.txt or ca.olf.olf (This is most common on Windows systems when hide known file extensions is enabled in the explorer).

**Q:** When I download certain CA data files and utilities from the support site, they come with an extension of .caz. What is a .caz file?

**A:** Files downloaded with a .caz extension are compressed with CAZIPXP, available at http://supportconnectw.ca.com/public/ca_common_docs/ca_utility_supp.asp. To uncompress the file, use the command: **cazipxp.exe -u <*filename.caz*>**, which extracts the compressed files to the local directory.

**Q:** I have installed my ALP license key and I am receiving the following error:

```
- Computer Associates Licensing -- The license found is
inadequate for the Hardware Unit rating of this machine. You
might have upgraded the machine and if so, please rerun the
appropriate license program to properly license your product
```

**A:** Typically, this is due to an invalid or corrupt lic98.cap file: open the lic98.cap file with a text editor, the file should be comprised of a listing of detected model types in the format of: **SYS (detectedmodel) (tier) (checksum)**

- If the file is improperly formatted, you can copy the lic98.cap from your installation source or download the latest data files at our support web site
- If the lic98.cap file does not exist, you may not have a current ALP license run-time installed. Contact TLC for assistance
- If the lic98.cap exists and appears to be valid, contact TLC for assistance.

## Modifying, Repairing, or Removing the Base Product

After installing the AllFusion Plex base product, you can run the installation again to perform the following operations:

- Modify the base product
- Repair the base product
- Remove the base product

To modify, repair, or remove the AllFusion Plex base product:

1. Click Start, then choose Settings, Control Panel.
2. Double-click the Add/Remove Programs icon on the Control Panel.
3. Select AllFusion Plex from the list of products.
4. Click Add/Remove Programs.
5. Select Modify, Repair, or Remove, as required.
6. Click Next and follow the instructions on displayed the dialog.

## Installing Other Components

The following sections explain how to install other components.

### The Windows Application Server

To install the Application Server tool when you install the AllFusion Plex base product:

1. Click Custom when asked to specify which type of setup to install.
2. Check AllFusion Plex Windows Application Server in addition to what is already checked.

To install the Application Server after you install the AllFusion Plex base product:

1. Click Start, then choose Settings, Control Panel.
2. Double-click the Add/Remove Programs icon on the Control Panel.
3. Select AllFusion Plex in the list of products.
4. Click Change/Remove.
5. Select Modify and click Next on the InstallShield Wizard window.
6. Click AllFusion Plex Windows Application Server to check it.
7. Click Next to start the installation.

## Application Integrator

To install the Application Integrator tool when you install the AllFusion Plex base product:

1. When asked to specify which type of setup to install, click Custom.
2. Check Application Integrator in addition to what is already checked.

To install the Application Integrator tool after you install the AllFusion Plex base product:

1. Click Start, then choose Settings, Control Panel.
2. On the Control Panel, double-click the Add/Remove Programs icon.
3. In the list of products, select AllFusion Plex.
4. Click Change/Remove.

5. On the InstallShield Wizard window, select Modify and click Next.
6. Click Application Integrator to check it.
7. Click Next to start the installation.

## Microsoft Visual Studio .NET

Microsoft Visual Studio 2003 .NET C++ compiler must be installed to generate and build WinC and WinNTC objects in AllFusion Plex.

AllFusion Plex only requires a typical installation for Microsoft's Visual Studio. See the Visual C++ documentation for instructions regarding the different configurations.

Note the following:

- The AS/400 5250 and Java generators do not require the Microsoft Visual C++ compiler. Local Java builds require Microsoft's NMAKE utility, which is part of Visual C++ and also freely distributed with the .NET Framework SDK.
- If you are building Windows Server (WinNTC) functions, you must install Visual Studio on the machine on which you are building the server functions.
- The Microsoft Visual C++ stand-alone product does not support code optimization features that are used for AllFusion Plex C++ builds. For this reason, the Professional Edition (or Architect Edition) is recommended.
- This release of AllFusion Plex does not support Visual Studio .NET 2005.

# Chapter 3 Develop Your First Application in 20 Minutes

AllFusion Plex is an Architected RAD tool that uses patterns to accelerate the design, creation, and maintenance of software applications. This chapter provides an introduction to some of the innovative features of AllFusion Plex.

In this chapter, you define a very simple project management application. The pattern libraries in AllFusion Plex are used to streamline the process. The Panel Designer is also used to make some changes to a dialog. Before the end of this chapter, you will have the first part of the application up and running.

## Concepts

The following section explains the concepts used in the application environment.

### Reusable Business Patterns

AllFusion Plex developers model applications by reusing *patterns*. A pattern describes a solution to a recurring problem in software systems. Patterns are abstract descriptions that can be reused in many contexts. An example of a pattern is Structure.

The Structure pattern provides the database schema, user interface designs, and programs that can be used to implement hierarchical data structures such as a bill of materials, an organization chart, or a chart of accounts. AllFusion Plex includes libraries of such patterns, and additional libraries are available from third-party vendors.

Several of the patterns from the AllFusion Plex pattern libraries are used to build the sample application.

## Workgroup Development

AllFusion Plex provides a model-based workgroup environment in which teams of software developers can collaborate on the design and construction of applications. At the heart of this environment is a repository whose facilities include:

- Multiple developer support that enables developers to work offline, and then update their changes to the central repository
- Built-in configuration management that enables a model to store a single application design in multiple versions, platforms, and national languages
- The integration of designs stored across multiple models

For simplicity, the sample in this chapter focuses on a single-user environment. The single-user model, called a *local model*, has already been created for you.

## Code Generation

Based on the designs held in the repository, AllFusion Plex automatically generates 100% of the code to implement applications and database objects across a variety of platforms. Currently supported implementation options include:

- Multi-tiered eBusiness applications with Java or HTML clients and Java, Windows, or iSeries servers
- Multi-tiered client/server applications with Win32 clients and Java, Windows, or iSeries servers
- AS/400 5250 character based-terminal applications

Computer Associates constantly improves and expands the available generators to keep pace with customer demand and advances in technology. Generators insulate you from the underlying technology and its implementation details—AllFusion Plex users can take advantage of new platforms simply by regenerating their existing designs.

The sample in this chapter creates a Windows-ODBC application because this is the simplest platform to set up and configure. The development process is very similar regardless of which platform you are developing for.

## Terms

In this chapter, you create an *entity* called Project. The Project entity uses *fields* to store information, such as its start and end dates. You inherit *functions* to enable end users to create, modify, and delete projects. Entities, fields, and functions are all types of AllFusion Plex *objects*.

A *local model* is the file that stores the design of the application you are building. In your local model, you create and define objects.

The *object browser* displays all of the objects in a model. In it, you can see the pattern library objects available, as well as the objects that you define in your model.

## Object Types

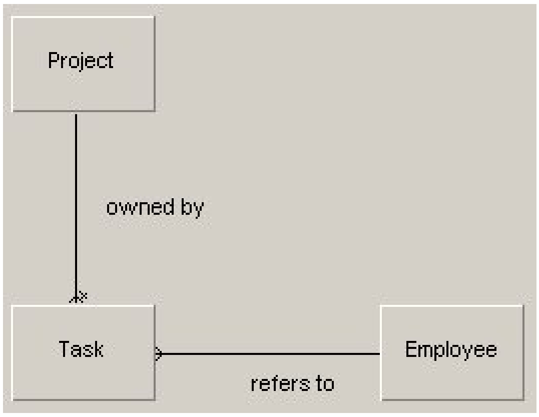
AllFusion Plex uses many types of objects. The following list shows the abbreviation and the icon that AllFusion Plex uses to identify each object type that you encounter while building the application in this chapter.

| Object Type | Abbreviation | Symbol | Description |
|---|---|---|---|
| Entity | ENT | | Entities are the tangible objects, such as customers and orders, and intangible objects, such as projects and job titles, about which you want to record information. |
| Field | FLD | | Fields are the pieces of information that you want to record about an entity, such as employee names, item prices, and order statuses. |
| Function | FNC | | Functions are processes that define the functionality of an application. A function is roughly equivalent to a program or method. |
| Panel | PNL | | Panels are the windows and dialogs that make up the user interface of the application. |
| Table | TBL | | Tables are the physical structures in which data is stored in applications that use relational databases. |
| View | VW | | Views represent all or part of the data in a database table. A view itself does not contain any data—you can think of it as a window through which you look at the data in the table. |
| Diagram | DGM | | Diagrams visually represent a subset of the objects in a model. They show objects and the relationships between the objects. |
| Message | MSG | | Messages hold text. The text can be used in error messages, as the caption for windows and dialogs, and so on. |

# Project Management Application

In this chapter, you create a simple project management application. For our purposes, a project consists of a group of tasks, and each task has one employee assigned to it. The model contains three entities: Project, Task, and Employee. To save time, the Employee entity is already defined for you.

Here is an entity-relationship diagram of the application created in this chapter. You will draw a diagram like this later.



If you have worked with entity-relationship diagrams before, you can see that:

- A project can have more than one task, but a task can only belong to one project.
- A task is owned by a project, which means that if you delete a project, you want all of its tasks deleted, too.
- Employees are assigned to tasks—one employee can be assigned to more than one task, but each task can only have one employee assigned to it.
- An employee is not dependent on any particular task, which means that if you delete a task, you do not necessarily want to delete the employee record too.

You can see all of this without having to look at any code. This diagram shows useful information, which AllFusion Plex uses to generate the application.

For the purpose of this sample application, this model is very basic. If you have developed real project management systems before, you may find parts of the model that you would design differently.

Even though it is a simple example, the end product is far from simple. In this chapter, you create both a wizard and a property sheet to work with the project data. If you have built these in other design environments, you know that you would not be able to reuse much code from the wizard when creating the property sheet. However, AllFusion Plex enables you to create a property sheet using the information you added to create the wizard, plus a few new lines of code.

## Opening the Sample Model

First you need to start AllFusion Plex and open the supplied sample model:

1. From the Start menu, choose Programs, then Computer Associates, AllFusion, AllFusion Plex, AllFusion Plex again; or just click the AllFusion Plex icon on your desktop.
2. From the File menu, choose Open.
3. In the Open File dialog, select TutorialWin.mdl, which is located in the Tutorial subfolder beneath the folder where you installed AllFusion Plex.
4. Click Open.

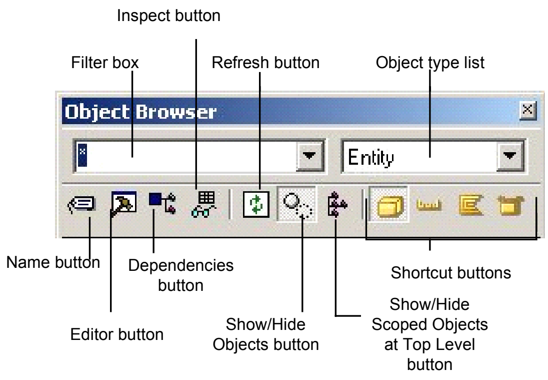
By default, when the model opens, a window called the Object Browser appears and the name of the model is displayed in the title bar of the main application window.

Save your work! If you need to stop working on this tutorial before reaching the end of a chapter, you can save your progress by clicking the Save toolbar button. AllFusion Plex also prompts you to save your changes when you close certain editors after making changes in them.

## Viewing the Object Browser

To show or hide the Object Browser, click the Show/Hide Browsers toolbar button (located on the right end of the toolbar) to show or hide the Object Browser.

The following graphic shows the object browser as it appears in the application:



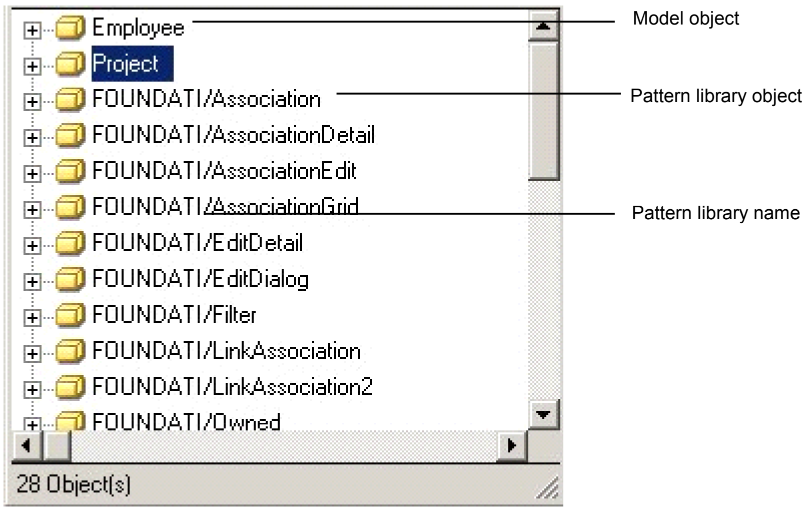
You can keep the Object Browser open while you work, and use it as a palette from which to drag the objects you need. When it overlaps other AllFusion Plex windows, it always appears on top.

By default, the Object Browser shows you objects of one type at a time. You can see other objects if they are *scoped* by the main object type. For more information, see More About Scope in this chapter. Notice that the Object Browser has shortcut buttons for displaying entities, fields, and functions. Setting the Object Browser to display functions only displays unscoped functions; functions that are scoped by another object (such as an entity or a view) are displayed when the Object Browser is focused on that object type.

You can also select an object type to view from the object list.

You can click the Show/Hide Library Objects toolbar button to show library objects.

You can tell that an object is in a pattern library because it has the library name in front of the object name by default. For instance, the second entity in this graphic is FOUNDATI/Association, which means that the entity Association comes from the FOUNDATION pattern library.



Employee
Project
FOUNDATI/Association
FOUNDATI/AssociationDetail
FOUNDATI/AssociationEdit
FOUNDATI/AssociationGrid
FOUNDATI/EditDetail
FOUNDATI/EditDialog
FOUNDATI/Filter
FOUNDATI/LinkAssociation
FOUNDATI/LinkAssociation2
FOUNDATI/Owned

28 Object(s)

## Defining the Project Entity

In AllFusion Plex, you define objects and specify relationships between them with *triples*. As the name implies, a triple has three parts: a source object, a verb, and a target object.

For example, in this chapter, you use the following triple to define a unique identifier for the Project entity:

Project **known by FLD** Project ID

In this example, Project (an entity) is the source object of the triple, **known by** (or **known by FLD**) is the verb, and Project ID (a field) is the target object.

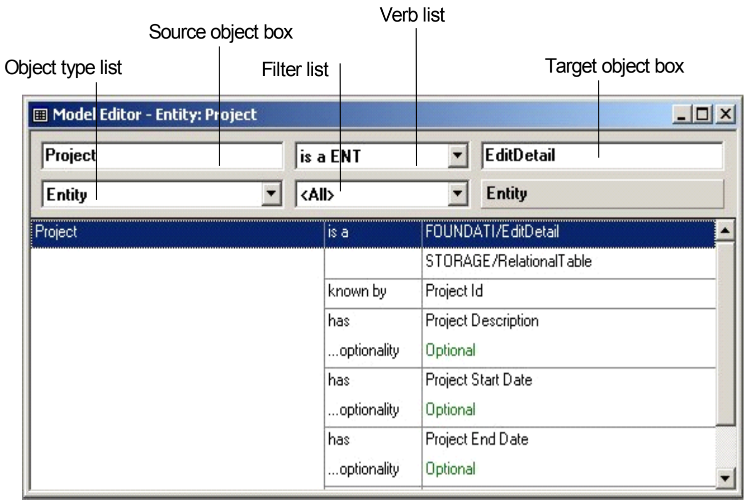
In the AllFusion Plex documentation, verbs are always printed in bold and usually with their source and target object types (ENT **known by** FLD, for example).

Triples provide a simple, easily understood language for entering information into AllFusion Plex. Here are some more examples of triples that you will enter in this chapter:

- Project **has FLD** Project Description
- Project Description **is a FLD** LongDescription
- Project **described by DGM** Project Diagram

In this chapter, triples are entered to define fields for the Project entity: an identifier, a description, and start and end dates for a project.

You use the Model Editor to view, add, edit, and delete triples. The following is the Model Editor as it will appear when you complete this chapter.



| | | |
|---|---|---|
| Project | is a | FOUNDATI/EditDetail |
| | | STORAGE/RelationalTable |
| | known by | Project Id |
| | has | Project Description |
| | ...optionality | Optional |
| | has | Project Start Date |
| | ...optionality | Optional |
| | has | Project End Date |
| | ...optionality | Optional |

## Specifying Attributes for the Project Entity

In the next series of steps you add the following triples to the model:

Project **known by** Project ID
Project has Project Description
Project has Project Start Date
Project has Project End Date

1. Open the Model Editor by choosing Model Editor from the Tools menu, or clicking the New Model Editor toolbar button.

   You can see some triples already entered in the Model Editor.

2. From the object type list, select Entity.

3. In the source object box, enter **Project**. Even though you have not specified anything about projects yet, you can type its name and AllFusion Plex adds the entity to your model.

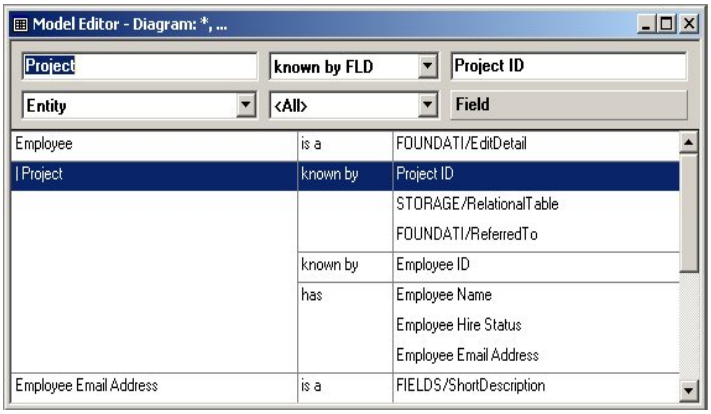
4. From the verb list, select **known by** FLD.

   **Note:** Instead of scrolling in the list, you can click on the verb list, and then press the first letter of the verb you are looking for until the verb appears. To select the **known by** FLD verb, press the K key once.

5. In the target object box, enter **Project ID**.

6. Press Enter.

   The Model Editor should now look like this:



Project | known by FLD | Project ID
Entity | <All> | Field

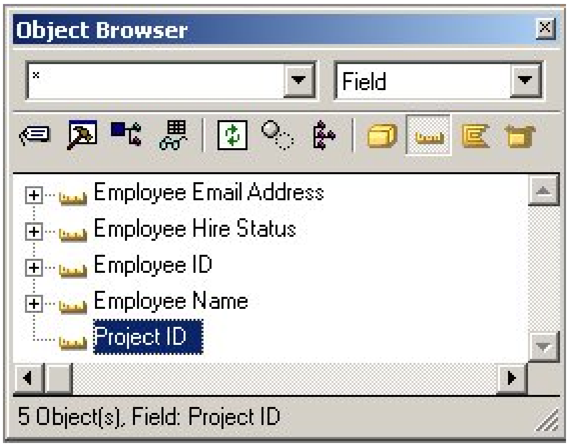
| | | |
|---|---|---|
| Employee | is a | FOUNDATI/EditDetail |
| Project | known by | Project ID |
| | | STORAGE/RelationalTable |
| | | FOUNDATI/ReferredTo |
| | known by | Employee ID |
| | has | Employee Name |
| | | Employee Hire Status |
| | | Employee Email Address |
| Employee Email Address | is a | FIELDS/ShortDescription |

   You just created the triple Project **known by** Project ID. This triple defines a primary key for the Project entity.

7. If the Object Browser is not open, click the Show/Hide Browsers toolbar button to open it.

8. Click the Fields toolbar button on the Object Browser to display field objects.

Notice the Project ID field you just added. All objects you define in a local model appear at the top of the Object Browser. Pattern library objects appear below the local objects.



Employee Email Address
Employee Hire Status
Employee ID
Employee Name
Project ID

5 Object(s), Field: Project ID

Notice that there are other fields already defined: Employee Email Address, Employee Hire Status, and so on. The Employee entity and its fields have already been defined in the model you are working with, to save time.

9. Project should still appear in the source object box in the Model Editor. Select **has** FLD from the verb list.

10. Enter **Project Description** in the target object box.

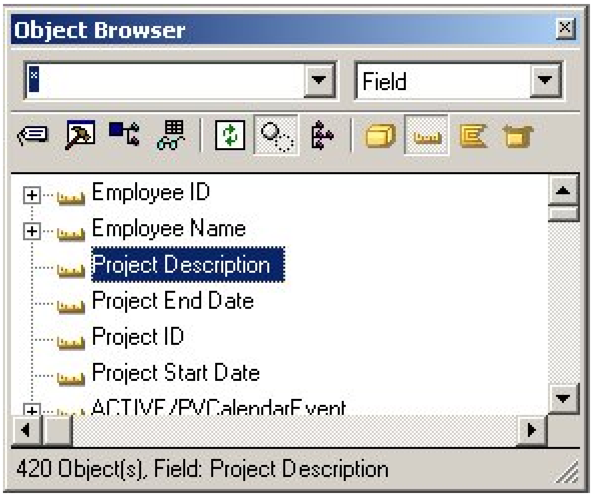
11. Press Enter.

You have created the triple Project **has** Project Description, which defines the field Project Description for the Project entity. You use this field to store a description of the project. This triple, ENT **has** FLD, creates a *non-key attribute*. The values in non-key attributes do not need to be unique to each entity. So, for instance, you may have more than one project that has the same text in the description field.

12. Repeat Steps 9 to 11 to create the following triples:

    Project **has** Project Start Date
    Project **has** Project End Date

13. Click the Refresh toolbar button on the Object Browser. The Object Browser shows the new fields:



Employee ID
Employee Name
Project Description
Project End Date
Project ID
Project Start Date
ACTIVE/PVCalendarEvent

420 Object(s), Field: Project Description

## Defining Field Properties

In the next series of steps you use inheritance to define the properties of fields. Inheritance in AllFusion Plex is defined in a simple, yet powerful way—using the **is a** verb. Here are the triples you enter in the next series of steps.

Project ID **is a** FIELDS/Identifier
Project Description **is a** FIELDS/LongDescription
Project Start Date **is a** DATE/CheckedDateISO
Project End Date **is a** DATE/CheckDateISO

The FIELDS and DATE prefixes indicate that the objects concerned belong to the FIELDS and DATE pattern libraries, respectively.

## More About Field Inheritance

The fields you defined in the previous section have two different data types: character and date. These represent different kinds of fields. The Project ID field holds code that uniquely identifies a project, the description holds text, and the start and end dates hold dates. Currently, your model only indicates that those fields exist, and that they belong to the Project entity, but has no information about what type of data they store.

Inheritance is the mechanism that enables an object to adopt the properties of another more general (or abstract) object. By inheriting from pattern library fields, you enable your application to:

- Validate data entered in the fields (which ensures, for example, that an end user does not accidentally enter February 31)
- Display data on the screen appropriately (such as displaying dates according to your Windows settings)
- Store data appropriately in the database (creating a text field in the database for the Project Description field, and date fields for the Project Start Date and Project End Date fields)

To define the properties of Project's fields:

1. Make sure that the Object Browser is focused on fields, and that library objects are showing (click the Show/Hide Library Objects button to display library objects and the Fields button to display field objects).
2. Select the Project ID field in the Object Browser by clicking the name (not the icon to the left of the name) and drag the field from the Object Browser to the source object box of the Model Editor.

   This changes the Model Editor source object type to Field, and changes the verb list so that only verbs appropriate for fields are contained in it.

   **Note:** The cursor changes to a closed parcel icon when you drag an object. It changes to an open parcel icon when it is over a location where you can drop the object:

The following table shows the closed and open parcel icons:

| Closed Parcel Icon | Open Parcel Icon |
|---|---|
| | |

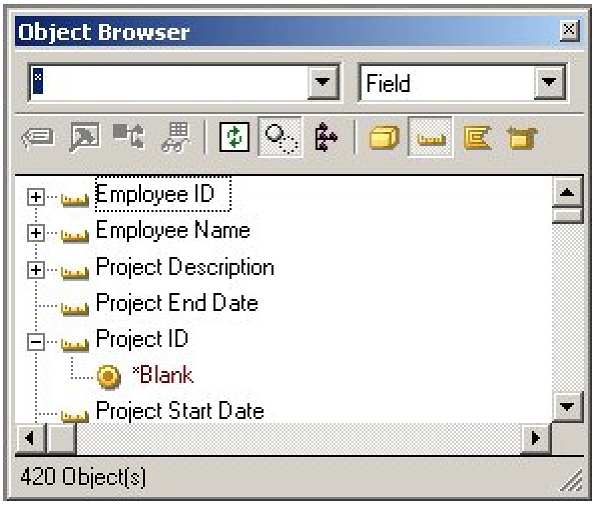
3. From the verb list, select **is a** FLD.

4. From the Object Browser, drag the library object FIELDS/Identifier field to the target object box, and press Enter.

   **Note:** You have to scroll down the Object Browser to find the FIELDS/Identifier field. You can use the filter box at the top of the Object Browser to only show some of the library items. In this case, you could type ***Identifier*** to display only FIELDS/Identifier. Remember to set the filter back to * when you are done.

5. You just created the triple Project ID **is a** FIELDS/Identifier. Click the Refresh button on the Object Browser. Notice that the Project ID field has a plus sign to the left, indicating that it now has scoped objects.

6. Click the plus sign to expand the field:



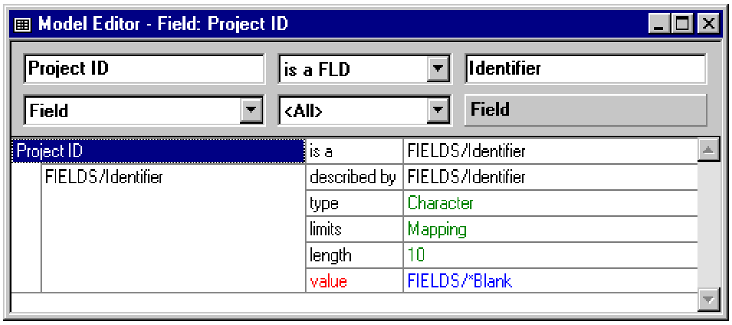
Values are another type of AllFusion Plex object. You can see that Project ID now has the value *Blank, but you cannot tell much else about what it inherited from FIELDS/Identifier.

7. Drag the Project ID field from the Object Browser to the body of the Model Editor. The body is the bottom part of the editor, where the full triples are displayed.

   When you drag one or more objects to the body of the Model Editor, the display changes to show you only the triples that define those objects. This is called *focusing* the Model Editor. When you drag the Project ID field to the Model Editor, it focuses on this field, showing the triple Project ID **is a** FIELDS/Identifier. This still does not give you much information.

8. To see more about what an object inherits from its ancestor objects, click the Two Levels toolbar button.

   The Model Editor shows another level of detail.



| | | |
|---|---|---|
| Project ID | is a | FIELDS/Identifier |
| FIELDS/Identifier | described by | FIELDS/Identifier |
| | type | Character |
| | limits | Mapping |
| | length | 10 |
| | value | FIELDS/*Blank |

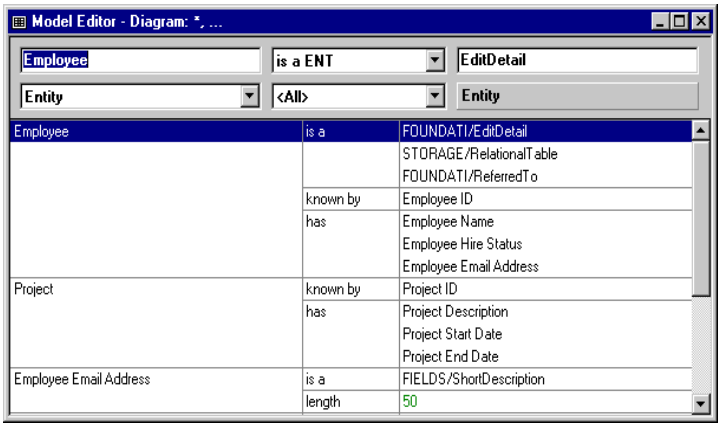
Now you can see that Project ID has inherited a data type of character and length of 10, along with the value *Blank (which you saw in the Object Browser in Step 6).

**Note:** In Step 4, you dragged the library object FIELDS/Identifier from the Object Browser to the target object box in the Model Editor. You can enter the name of the object into the target object box (without the library name) to accomplish the same thing. In Step 4, you would have entered Identifier.

***Important!*** *If you entered a wrong object name, you could create a new object with the wrong name. If this happens, find the erroneous object in the Object Browser and delete it, checking the Ripple Delete check box on the Delete dialog.*

9. Click the One Level toolbar button to set the Model Editor to show a single level of information.

10. Reset the Model Editor display by clicking the Clear Focus button. This displays all of the triples in the model, again:



Model Editor - Diagram: *, ...

|  |  |  |
|---|---|---|
| Employee | is a ENT | EditDetail |
| Entity | <All> | Entity |

|  |  |  |
|---|---|---|
| Employee | is a | FOUNDATI/EditDetail |
|  |  | STORAGE/RelationalTable |
|  |  | FOUNDATI/ReferredTo |
|  | known by | Employee ID |
|  | has | Employee Name |
|  |  | Employee Hire Status |
|  |  | Employee Email Address |
| Project | known by | Project ID |
|  | has | Project Description |
|  |  | Project Start Date |
|  |  | Project End Date |
| Employee Email Address | is a | FIELDS/ShortDescription |
|  | length | 50 |

If your Model Editor displays many more triples than shown in the previous graphic, then you have your model set to display library objects. If this is the case, click the Show/Hide Library Objects toolbar button.

11. Drag the Project Description field from the body of the Model Editor to the source object box. This field is in the third column of the Project **has** Project Description triple.

12. Enter **LongDescription** in the target object box, and press Enter.

    Notice that the Model Editor displays the triple as:

    Project Description **is a** FIELDS/LongDescription

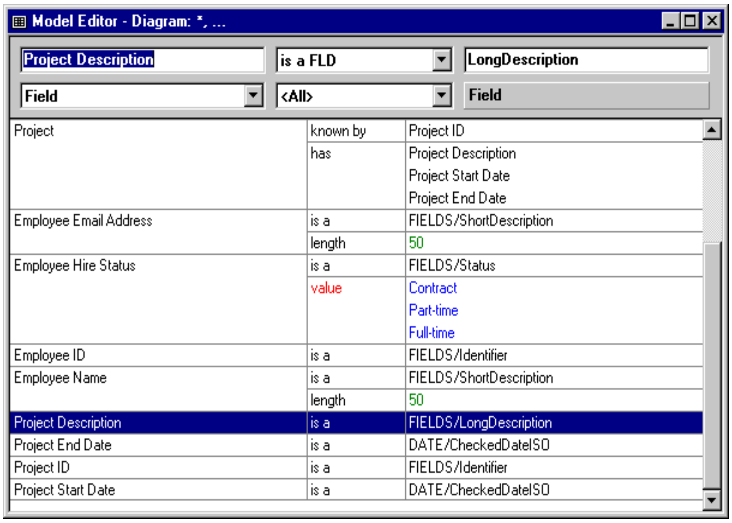
    This indicates that you correctly spelled the name of the pattern library field.

    **Note:** If you create a new object for a model, and it happens to share the name of a library object, you must rename your object or delete it (if you did not intend to create it).

13. Repeat either Steps 2 to 4 or Steps 11 to 12 to create the following triples:

    Project Start Date **is a** DATE/CheckedDateISO
    Project End Date **is a** DATE/CheckedDateISO

14. Click the Refresh toolbar button (on the main toolbar, not on the Object Browser). Your Model Editor should look like this:



| | | |
|---|---|---|
| Project Description | is a FLD | LongDescription |
| Field | <All> | Field |

| | | |
|---|---|---|
| Project | known by | Project ID |
| | has | Project Description |
| | | Project Start Date |
| | | Project End Date |
| Employee Email Address | is a | FIELDS/ShortDescription |
| | length | 50 |
| Employee Hire Status | is a | FIELDS/Status |
| | value | Contract |
| | | Part-time |
| | | Full-time |
| Employee ID | is a | FIELDS/Identifier |
| Employee Name | is a | FIELDS/ShortDescription |
| | length | 50 |
| Project Description | is a | FIELDS/LongDescription |
| Project End Date | is a | DATE/CheckedDateISO |
| Project ID | is a | FIELDS/Identifier |
| Project Start Date | is a | DATE/CheckedDateISO |

Inheriting from DATE/CheckedDateISO gives the fields functionality to ensure that end users enter valid dates.

15. Use the process explained in Steps 7 and 8 to look at the characteristics that these fields inherit from the pattern library fields.

### Using Continuation Triples to Make Fields Optional

Hopefully, you are getting comfortable with understanding and entering triples. We now introduce an extension to the triple syntax—the *continuation triple*. A continuation triple is like other triples except that the source object is itself a triple. In the next series of steps, you enter the following continuation triples to specify that certain fields are optional:

Project **has** Project Description
**...optionality** Optional

Project **has** Project Start Date
**...optionality** Optional

Project **has** Project End Date
**...optionality** Optional

You will enter other continuation triples later in this chapter.

## More About Optionality

When your end users enter data, they typically enter data in every field. You can specify that some fields are mandatory, while others are optional. If you do not specify optionality, the default is that they are mandatory.

Primary key fields must always be mandatory, so you should leave the Project ID field as it is. However, your end users may use a Project ID that is descriptive enough that they do not need to enter any data in the Project Description field. In addition, when they create projects, they may not have a start or end date yet.

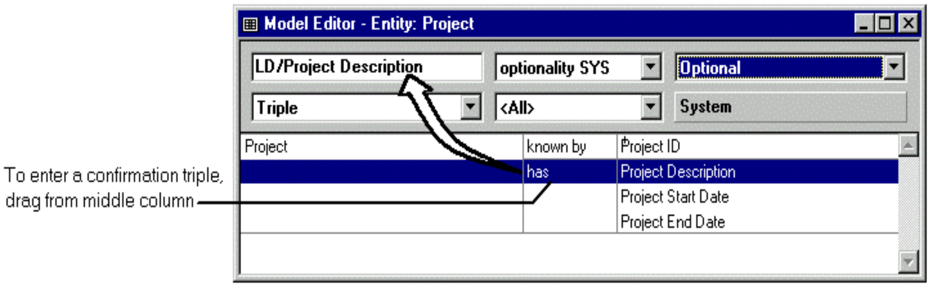
If end users leave a mandatory field blank and then try to close the dialogs, a message dialog prompts them to enter data for the blank mandatory field, and does not let them close the dialog until they do. Since this processing is defined as part of the pattern library and not hard-coded into AllFusion Plex, you can adapt it as required (but this concept is not explained in this chapter).

To make fields optional:

1. Click the Entities toolbar button to set the Object Browser to display entities.
2. Select the Project entity in the Object Browser and click the Inspect toolbar button.

   This focuses the Model Editor on the Project entity, showing only the triples that define that entity.

3. Click in the center of the triple Project **has** Project Description to select it.
4. Drag it to the source object (top left) box in the Model Editor:



5. Select **optionality** SYS from the verb (top middle) list, and Optional from the target object (top right) list.

6. Press Enter.

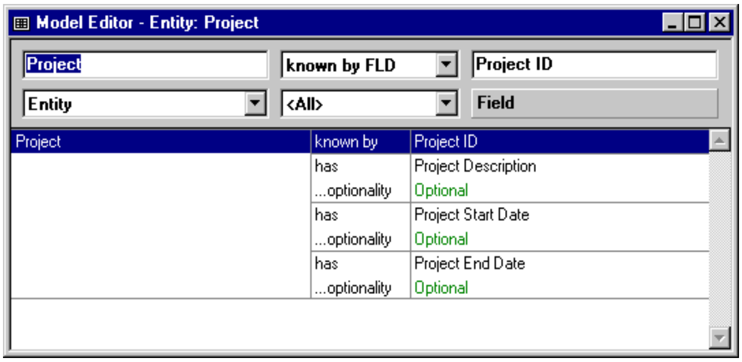
   You have entered the continuation triple:

   Project **has** Project Description
   **...optionality triples** Optional

   **Note:** If you do not see this view, click the One Level toolbar button to set the Model Editor to show a single level of information.

7. In the same manner, make the Project Start Date and Project End Date fields optional.

8. Refresh the Model Editor:



| | | |
|---|---|---|
| Project | known by FLD | Project ID |
| Entity | <All> | Field |

| Project | known by | Project ID |
|---|---|---|
| | has | Project Description |
| | ...optionality | Optional |
| | has | Project Start Date |
| | ...optionality | Optional |
| | has | Project End Date |
| | ...optionality | Optional |

## Using Inheritance to Define the Project Entity

You have now defined the fields in which the Project entity stores data, and specified the pattern library fields that those fields inherit from. In the next step, you give the Project entity a user interface and functionality to interact with a database.

You again use inheritance to add this functionality. In fact, you do it with only two triples:

Project **is a** FOUNDATI/EditDetail
Project **is a** STORAGE/RelationalTable

The first inheritance triple gives your entity the ability to display and process a user interface. The second inheritance triple provides the functionality to read Project records from, and write Project records to, a relational database.

The step-by-step instructions that follow help you better understand the consequences of entering these two triples.

## Patterns and Inheritance

Inheritance is the primary means of reuse when you design an application with AllFusion Plex. Inheriting from prebuilt and pretested pattern libraries provides tremendous productivity and quality benefits.

*Patterns* are groups of AllFusion Plex objects that are designed to be reused and customized. In this chapter, you inherit from the included pattern libraries. Additional pattern libraries are available from third-party vendors.

**Patterns compared to Templates.** The templates or frameworks supplied by other tools provide some of the benefits of patterns. Typically, a template provides a means of copying a predefined solution. Inheritance is not the same as copying. When an object inherits from an AllFusion Plex pattern, the relationship is dynamic—changing a pattern automatically changes all the objects that inherit from the pattern.

Also note that patterns are created with AllFusion Plex rather than being hardcoded into the tool. You can customize a pattern library by creating your own patterns that inherit from the supplied patterns, or you can create your own patterns from scratch.

**Patterns and Components.** Patterns and components (such as COM and EJB components) are complementary technologies. AllFusion Plex patterns can be used to combine and implement groups of components. Components are usually compared to the building blocks of a house. Extending this metaphor, you can think of patterns as the blueprint or plan for the house.

Like patterns, components are intended to maximize software reuse. Components cannot be changed, where patterns have an internal structure you can modify in certain places. Another distinction is that patterns are purely a design construct where components form a part of the running application.

## Adding Functionality to the Project Entity

To add functionality to the Project entity:

1. If the Object Browser is not displaying entities, click the Entities toolbar button.

   Notice that there is no plus sign to the left of the Project entity. This tells you that there are no objects scoped to it.

   To add these triples, you must first set the source object type in the Model Editor to Entity. You could change the object type directly. But, when you drag an object from the Object Browser, it sets the object type *and* shows all of the triples for that object.

2. Drag Project from the Object Browser to the source object box of the Model Editor.

   This is similar to using the Inspect toolbar button; it changes the Model Editor so that it only shows triples related to the Project entity, changes the object type (assuming it was not already set to Entity), and puts Project in the source object box.

3. From the verb list, select **is a** ENT.

   **Note:** The **is a** verb that you use in this step is different than the one that you used to specify in inheritance for Project's fields. There are several verbs that have the same name, but have a different source and target object. AllFusion Plex only lets you select the verb that matches the target object (in this step, the **is a** ENT verb). For more information about the types of **is a** verbs, search for "is a" in the online help index.

4. Enter **EditDetail** in the target object box, and press Enter.

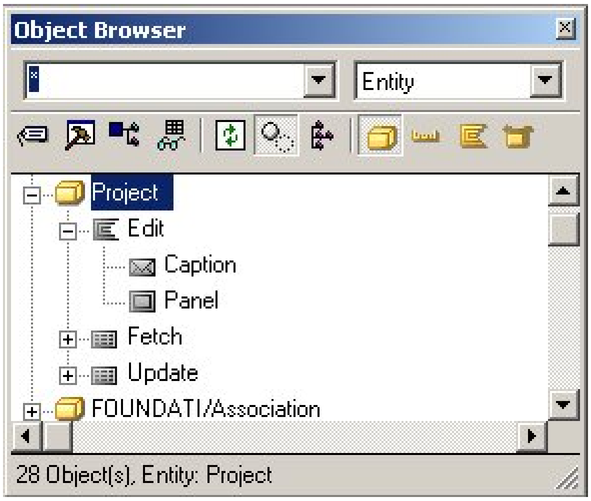
   You just created the triple Project **is a** FOUNDATI/EditDetail. This indicates that Project inherits the structure and functionality of the EditDetail pattern in the FOUNDATION pattern library. You can find the FOUNDATION/EditDetail pattern in the Object Browser by making sure that library objects are displayed (by clicking the Show/Hide Library Objects toolbar button) and scrolling down.

   For more information about the EditDetail pattern, select it in the Object Browser and press SHIFT+F1.

5. In the Object Browser, click the Refresh toolbar button. By inheriting from EditDetail, the Project entity now has some scoped objects (it has a plus sign next to it).

   For additional information about scoped objects, see More About Scope later in this chapter.

6. Click the plus sign to the left of the entity icon to expand Project, and then click the plus sign next to the Edit function to expand it:



Project inherited one function, Edit, with a scoped panel and a caption, and two views, Fetch and Update. These objects give Project a user interface, and enable it to store data to and retrieve data from a database.

Specifically:

- The Edit function displays the panel scoped by it.
- The Caption and Panel objects, which are scoped by the Edit function, store the layout of the Edit Projects panel. To see what this panel looks like, see the generated application illustration in Running the Project.Edit Function later in this chapter.
- The Fetch and Update views scope functions that read and write database records.

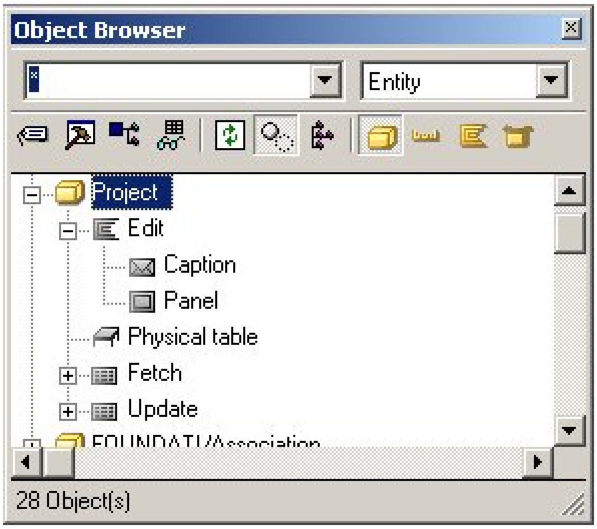
Next, indicate how the Project entity stores information. Your application uses a relational database, so you need processing that creates and maintains database tables. Set Project to inherit from STORAGE/RelationalTable for that functionality.

7. The Model Editor should still have Project in the source object box, and is a ENT showing in the verb list. Drag STORAGE/RelationalTable from the Object Browser to the target object box and press Enter to create the following triple:

   Project **is a** STORAGE/RelationalTable

8. Click the Refresh button to see all of the triples that you have defined for the Project entity in the Model Editor.

9. Click the Refresh button on the Object Browser:



Notice that the Project entity has inherited an object called Physical Table from STORAGE/RelationalTable. This defines the table that is created in the database when you generate the application. The fields that you defined for the Project entity are stored in this table.

## More About Scope

A *scoped object* is an object that belongs to another object. A scoped object cannot exist independently of the object by which it is scoped. For example, a panel that is scoped by a function is deleted when that function is deleted.

Some types of objects are unscoped. This means that they exist independently of all other objects in the model. For example, entities are typically unscoped. Conversely, a table is always scoped to an entity. To create a table object, you must specify the entity to which it belongs.

When you refer to a scoped object, you usually need to use its full name to avoid ambiguities. For example, there is an entity called Project that scopes a function called Edit. The function's full name is Project.Edit which distinguishes it from other Edit functions such as Employee.Edit that also exist in this model.

The previous steps demonstrated how scoped objects are usually created when you inherit from a pattern. You can also create scoped objects manually.

## Generating and Building

You have now defined fields for the Project entity, and specified the properties of those fields. You have also defined functionality for the Project entity, providing a basic user interface and the ability to write to and read from a database. You are now ready to generate and build the Project entity. This is the process in which AllFusion Plex turns your model into source code (generating), and then turns your source code into compiled objects (building).

After you have generated and built the objects in your model, you will be able to run the program to see what you have created.

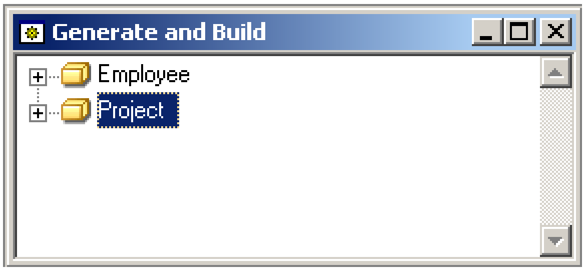
To generate and build the Project entity:

1. From the Tools menu, choose Generate and Build, or click the New Gen and Build toolbar button.

2. The Generate and Build window appears. The Message Log pops up when you open the window. From the Options menu, choose Quiet Mode, and then minimize the window. This keeps the Message Log from popping up every time it has a new message.

3. If the Generate and Build window shows library objects, click the Show/Hide Library Objects toolbar button to hide them.

   The Generate and Build window now shows the Project and Employee entities (Employee was already added to your model before you started):

4. Select the Project entity.
5. Click the Generate and Build toolbar button.

   AllFusion Plex expands Project and highlights all of the generatable objects under it. Not all of the scoped objects are selected.

6. A Confirm Generate dialog appears, indicating the number of objects that are generated. Click Yes.

   AllFusion Plex generates those objects, and then summarizes the generation process.

   **Note:** Three warnings appear in the Message Log during the generation and build processes. This is expected and not a problem. You can expect **one** warning message (BLD9083) to occur during the generation stage (for more information, see Checking for Errors in this chapter).

7. When the generation process is complete, the Cancel button becomes an OK button. Click OK to close the Generation Status dialog.

8. AllFusion Plex prompts you to compile and build the objects. Click Yes both times.

AllFusion Plex starts Microsoft Visual Studio to compile the generated C++ source files. Depending on your settings, Visual Studio may start minimized. Note that if you have not started Visual Studio at least once since installing it, it will not open automatically (the first time it opens, it creates certain registry entries that AllFusion Plex requires).

At the same time, the database table is sent to the ODBC data source being used for this sample application. To make setup easier, AllFusion Plex created this data source automatically during installation together with the underlying Microsoft Jet database.

**Note:** You can expect to see two of the three warning messages during the build process (for more information, see Checking for Errors later in this chapter).

You can see that your C++ build is done when the label on the Visual Studio taskbar button changes from the name of the model to Microsoft Development Environment [design]. If you have opened the Visual Studio window so that you could watch the build, you can also see that it is finished when the cursor returns to the top of the build summary.

## Native Platform Implementation

AllFusion Plex provides a complete environment for generating and compiling across all supported platforms. For Windows implementations, you can see how it seamlessly integrates with Microsoft Visual Studio.

AllFusion Plex generates native C++ that is based on Microsoft Foundation Classes (MFC), a robust, industry standard C++ class library. Each function in the model is implemented as a Windows DLL.

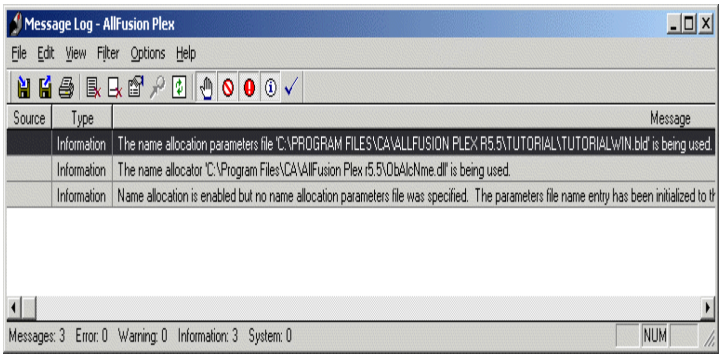
If we chose to implement our application in Java, AllFusion Plex would generate 100% pure Java code based on Sun's Java 2 platform. Similarly, we could also implement the application on the IBM iSeries by generating native RPG and DDS code.

## Checking for Errors

Three warnings appear in the Message Log during the generation and build processes. This is expected and not a problem. If you have additional warnings or error messages this may indicate a problem. The most likely cause is that you have not entered all the necessary triples into the model—go back and check against the instructions.

Errors are also likely if AllFusion Plex did not install properly.

- One warning message (BLD9083) indicates that the Project Description field has a length greater than 255. This is not a problem for the database we are using.
- Two warning messages (BLD9081) indicate that no source for various views has been submitted to the ODBC data source. This is expected because we have set up AllFusion Plex not to implement views in the Jet database being used for this sample application.


Message Log - AllFusion Plex
File Edit View Filter Options Help

| Source | Type | Message |
|---|---|---|
| | Information | The name allocation parameters file 'C:\PROGRAM FILES\CA\ALLFUSION PLEX R5.5\TUTORIAL\TUTORIALWIN.bld' is being used. |
| | Information | The name allocator 'C:\Program Files\CA\AllFusion Plex r5.5\ObAlcNme.dll' is being used. |
| | Information | Name allocation is enabled but no name allocation parameters file was specified. The parameters file name entry has been initialized to th |

Messages: 3 Error: 0 Warning: 0 Information: 3 System: 0 NUM

### Checking for C++ Build Errors

To check for C++ build errors, follow these steps:

1. Click the WinC Build Summary toolbar button to view the build summary, to make sure there were not any errors.

   **Note:** The WinC Build Summary toolbar button may not work if the path to your generation directory contains embedded spaces.

   You are not prompted to review the build summary any more in this sample application, but if your application does not run correctly, you should review to see if there were errors.
2. Click the Save toolbar button to save your model.

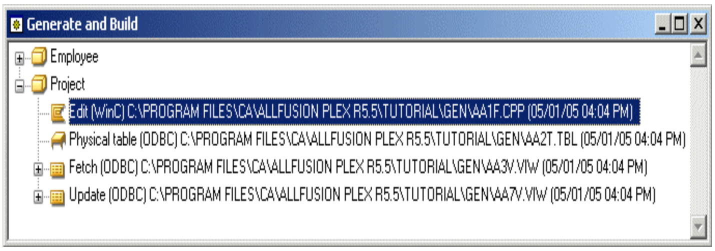
## Using Your Generated Application

By entering a few triples, you have created a fully functional application—an instance of the EditDetail pattern. You can use your generated application to create, edit, and delete projects. You can create a description for each project, and indicate a start and end date.

## Running the Project.Edit Function

To run the Project.Edit function, follow these steps:

1. In the Generate and Build window, select the Edit function under the Project entity:

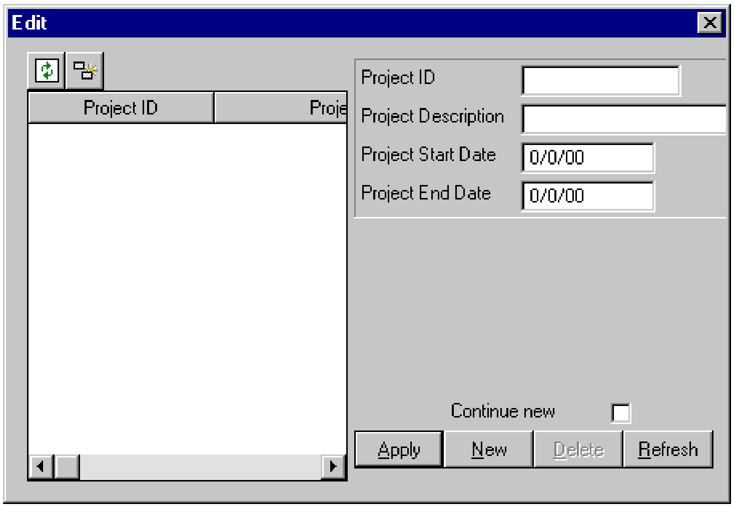

Generate and Build

Employee
Project
Edit (WinC) C:\PROGRAM FILES\CA\ALLFUSION PLEX R5.5\TUTORIAL\GEN\AA1F.CPP (05/01/05 04:04 PM)
Physical table (ODBC) C:\PROGRAM FILES\CA\ALLFUSION PLEX R5.5\TUTORIAL\GEN\AA2T.TBL (05/01/05 04:04 PM)
Fetch (ODBC) C:\PROGRAM FILES\CA\ALLFUSION PLEX R5.5\TUTORIAL\GEN\AA3V.VIW (05/01/05 04:04 PM)
Update (ODBC) C:\PROGRAM FILES\CA\ALLFUSION PLEX R5.5\TUTORIAL\GEN\AA7V.VIW (05/01/05 04:04 PM)

2. Click the Run toolbar button.

3. When you are prompted to choose the data source for your application, double-click Plex Tutorial.dsn.

   Your generated application starts. It should look like this:



Project Start Date: 0/0/00
Project End Date: 0/0/00

Leave this dialog open. The next steps show you how to add projects to your database.

## Default Panel Layouts

Based on the design details you entered earlier, AllFusion Plex has automatically added the correct fields to the panel and provided a default layout. You adjust the default layout later. The general appearance and functionality of this panel, with the grid on the left and the fields on the right, is determined by the pattern library from which it is inherited. You can override almost any aspect of this appearance and behavior.

### Adding a New Project

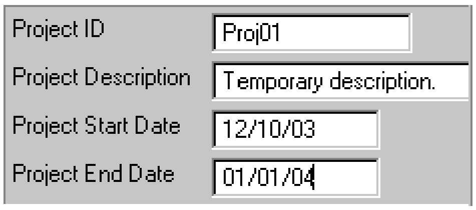
To add a new project:

1. You started your application in the previous procedure. Now, enter the following values.

   **Note:** The format required for the dates vary depending on your regional settings in the Windows Control Panel.

| Value | Enter |
|---|---|
| *Project ID* | **Proj01** |
| *Project Description* | **Temporary description.** |
| *Project Start Date* | **12/10/2003** |
| *Project End Date* | **01/01/2004** |

   The area on the right of the dialog should look like this after you have entered these values:



| | |
|---|---|
| Project ID | Proj01 |
| Project Description | Temporary description. |
| Project Start Date | 12/10/03 |
| Project End Date | 01/01/04 |

For now, you only enter **Temporary description.** in the Project Description field because the field extends past the right edge of the dialog, and if you type anything longer, you will not be able to see what you are typing. Since Project Description was defined as an optional attribute, you can leave it blank if you prefer.

After you have finished entering data, you modify the panel layout so that you can see more of the description, and so that it does not run off the right edge.

2. Click the Apply button. The values are posted to the database, and the grid region on the left of the window shows the new project.

3. Click the Copy toolbar button to create a new project populated with Proj01's data. Add the following two projects to the database:

| Value | Enter |
|---|---|
| *Project ID* | **Proj02** |
| *Project Description* | **Temporary description.** |
| *Project Start Date* | **03/01/2001** |
| *Project End Date* | **04/04/2001** |

| Value | Enter |
|---|---|
| *Project ID* | **Proj03** |
| *Project Description* | **Temporary description.** |
| *Project Start Date* | **05/05/2004** |
| *Project End Date* | **06/05/2004** |

4. Click the close window button at the upper-right corner of the dialog to exit the application.

   **Note:** None of the panels that are inherited from the pattern libraries have Cancel buttons on them. If you click the Close Window button, any pending actions are discarded. If you clicked Apply, though, those changes are completed before the window closes.

## 100% Code Generation

You have entered 13 triples into the model. From those triples, AllFusion Plex generated over 9,000 lines of C++ and SQL code to create a fully functional application. You got one hundred percent code generation!

Feel free to experiment. Examine referential integrity checking inherited from the pattern library. For example, you cannot enter the same Project ID for multiple records.

## Preserving Data

By default, each time you build your application, AllFusion Plex rebuilds all of the objects you select in the Generate and Build window, including the tables in your database. Because rebuilding a database table erases all data in the table, if you leave your local model set up as it is, you will lose all of the data you just entered the next time you build.

You can prevent losing this data by entering a TBL **implement** SYS No triple for the table. This keeps the table from being rebuilt the next time you build the entity to which it is scoped (in this case, Project). Only set this triple after you have built the table at least once.

To keep the Project entity table from being regenerated:

1. In the Object Browser, select Project.Physical Table, and drag it to the source object box in the Model Editor.
2. From the verb list, select **implement** SYS.

3. From the target object list, select the value No.
4. Press Enter.
5. Save your model.

Keep in mind that if you make any changes to an entity that affect its table, such as adding fields to it, you must set the **implement** SYS value back to Yes, and regenerate the table. Any data in it is lost. If you want to preserve data entered after rebuilding, make sure you reset the **implement** SYS value to No.

## Introducing the Panel Designer

You may have noticed when you ran the Project.Edit function that the dialog did not look quite right. The Project Description field ran off the right edge, and there was not enough room to enter more than a couple of words of description.

The Panel Designer changes the panel displayed by Project.Edit. You change the size of the Project Description field and give it a multiline edit control (so that you can enter multiple lines of text). Some spin controls are added to the date fields too.

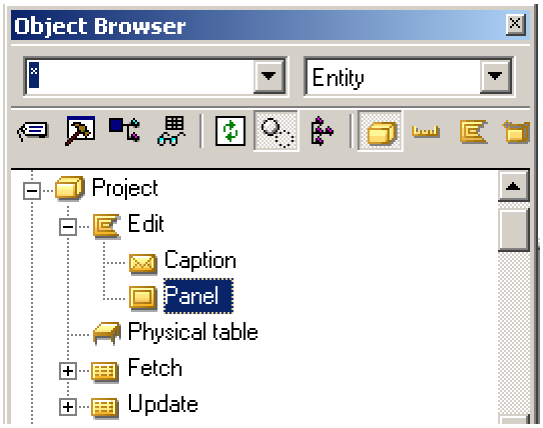
### Visual Development

You have seen that, unlike visual development tools, the starting point for an AllFusion Plex application design is a design model, not screen layouts. This is because design and modeling are the keys to building large-scale applications successfully. However, just like a visual programming tool, AllFusion Plex provides an easy-to-use editor for designing graphical user interfaces (GUIs). It includes a rich set of native GUI controls plus the ability to use third-party components (ActiveX controls in Windows and JavaBeans in Java). There is another mode of editor that enables the character-based screens of an AS/400 5250 application to be designed in the same way.

## Opening the Panel Designer

To open the Panel Designer:

1. If the Object Browser is not displaying entities, click the Entities toolbar button.
   Remember that this button is not on the toolbar. It is on the Object Browser. Some buttons are on both the Object Browser and the toolbar.
2. If you do not see the Project entity, click the Refresh toolbar button.
3. Expand the Project entity, expand the Edit function, and select Panel:



4. Click the Editor toolbar button. The Panel Designer appears. You can click the Editor button when you select any object. AllFusion Plex always opens the appropriate editor.

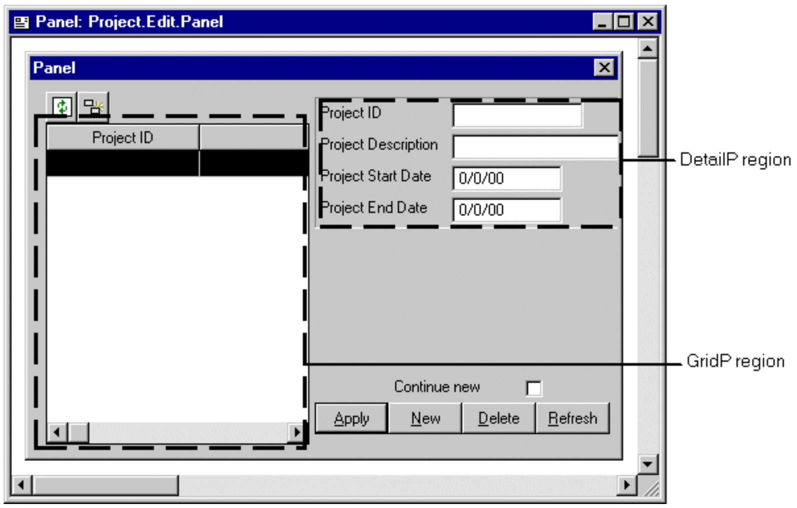
   The Panel Designer is made up of three windows:

   - Design Window
   - Panel Palette
   - Panel Property Sheet

## Design Window

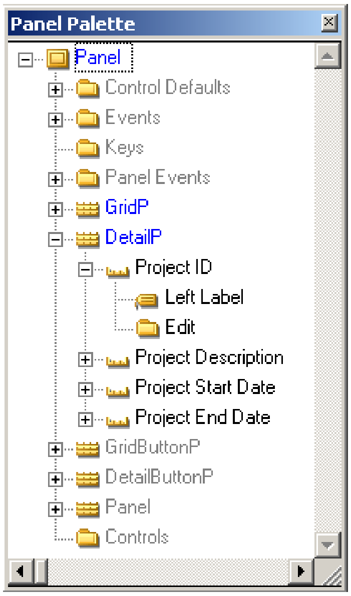
The Design Window is the main window in the Panel Designer, and shows you what your panel looks like. When you change a panel’s properties, you see how the changes affect what the panel looks like here. You can select, move, and resize buttons, fields, and other user interface elements using this window. When you make visual changes to the panel and its elements using the other windows, the changes appear here.

The elements of a panel are grouped into regions (each region has a name). The following graphic shows the grid and detail regions in the panel displayed by Project.Edit:



## Panel Palette

The Panel Palette shows all the elements of the panel, including fields, labels, and buttons, that are grouped under folders (🗀) and regions (☰). You can expand the folders and regions to see what they contain. Most of the visible elements are contained in regions. Notice that you can see the five regions of the panel (GridP, DetailP, GridButtonP, DetailButtonP, and Panel) in the Panel Palette (left).



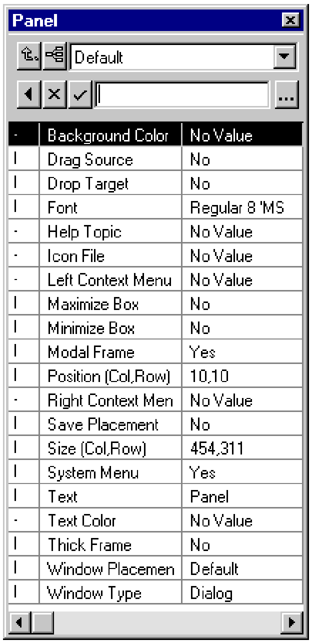
When a region is expanded, as the DetailP region is in this example, you can see the elements contained in that region. In the case of the DetailP region, this includes the Project ID, Project Description, Project Start Date, and Project End Date fields.

When represented on panels, fields typically contain more than one part, usually including at least one label and a control. Notice on the previous graphic that the Project ID field shows both a Left Label and an Edit control. The label indicates whether it is a Left Label, a Top Label, or a Right Label. The type of label indicates if it appears to the left or right of the control (left/right label), or as a column heading (top label). The control is the part that end users interact with. The settings for the control indicate if it is displayed as an edit box, a list box, a combo box, and so on.

## Property Sheet

Every element on a panel has a set of properties associated with it, such as its size, color, or position. The Property Sheet shows you the properties of the currently selected element. If no element is selected, the Property Sheet shows you the properties of the panel as a whole.

To see what you can change about an element, select the element in the Design Window or the Panel Palette, and check its properties on the Property Sheet. The properties displayed depend on the type of element selected.



| | | |
|---|---|---|
| · | Background Color | No Value |
| I | Drag Source | No |
| I | Drop Target | No |
| I | Font | Regular 8 'MS |
| · | Help Topic | No Value |
| · | Icon File | No Value |
| · | Left Context Menu | No Value |
| I | Maximize Box | No |
| I | Minimize Box | No |
| I | Modal Frame | Yes |
| I | Position (Col,Row) | 10,10 |
| · | Right Context Men | No Value |
| I | Save Placement | No |
| I | Size (Col,Row) | 454,311 |
| I | System Menu | Yes |
| I | Text | Panel |
| · | Text Color | No Value |
| I | Thick Frame | No |
| I | Window Placemen | Default |
| I | Window Type | Dialog |

For example, you can specify if a button should be included in the tab sequence by setting its Tab Stop property. However, a field label would never be included in a tab sequence, so if you select a label, the Property Sheet does not display this property.

## Defining a Multiline Edit Control

In this next set of steps, you modify the panel displayed by Project.Edit so that it displays more than one line of text and so that it does not extend past the edge of the dialog.

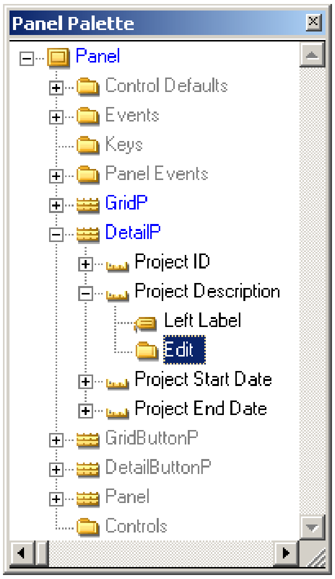
To modify Project.Edit.Panel:

1. Arrange the Panel Designer windows so they do not overlap, and you can work with the contents of each.

   You must select an element on a panel to make a change to it. To select only the element, and not the region it is in, use the Panel Palette.

2. If it is not already expanded, expand Panel to display the regions on the Edit panel.

3. Expand the DetailP region, and then expand the Project Description field.



4. Select the Edit control.

   The values in the Property Sheet indicate the properties you can change for the edit box. Also, notice in the Design Window that only the edit box next to the label Project Description is selected.

5. In the Design Window, drag the edit control below the label, so that its left edge lines up more or less with the j in the word Project. Do not be concerned that it covers up the Project Start Date field—this is fixed later.

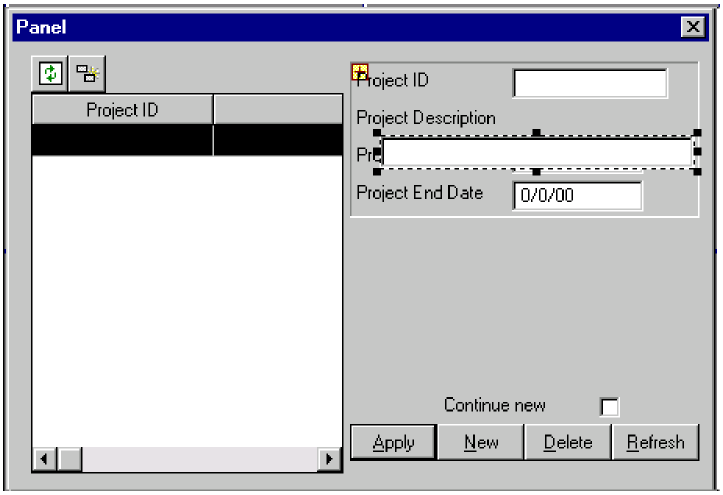
   In the Property Sheet, the Position (Col, Row) property changes as you move the element. If you want to make fine tuning changes, you can specify the exact pixel location in the Property Sheet.

   The field still extends past the right edge of the panel.

6. In the Property Sheet, click the Size (Col, Row) property.

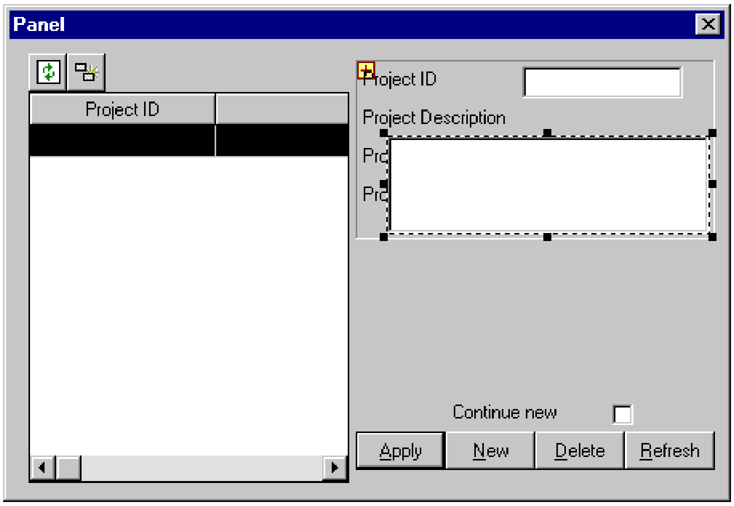
7. ✓ In the edit area at the top of the Property Sheet, change the value to 200, 20, and then click the check mark button.

   The edit control resizes so that it fits within the panel, and the beveled box that surrounds the fields resizes so that it is just slightly wider than the fields.



8. Drag the handle on the bottom center of the edit box to make the box taller. Make it about three times as tall.

   The panel should look something like this:



   You have repositioned and resized the Project Description edit box. Next, set two properties that enable end users to enter more than one line of text in the edit box.

9. In the Property Sheet, double-click the Multiline property to change its value from No to Yes.

10. Select the Scrolling property in the Property Sheet.

    **Note:** If the Scrolling property is not available, ensure that the value of the Multiline property changed from No to Yes.

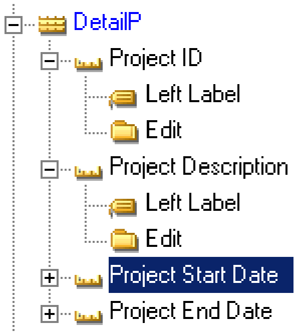
    Notice that the input area at the top of the Property Sheet changed so you can select a value from a list, but you cannot type in a value.

11. Select Vertical from the list.

    Setting the Scrolling property to Vertical causes the text in the multiline edit box to wrap to the next line when the cursor reaches the right edge. If you did not set this option, text would continue on the first line and extend beyond the edge of the field, rather than wrapping to the next.

When you changed the size and shape of the Project Description edit box, it covered both the Project Start Date and the Project End date fields. Next, you move those two fields down.

12. In the Panel Palette, select the Project Start Date field in the DetailP region, without expanding it.

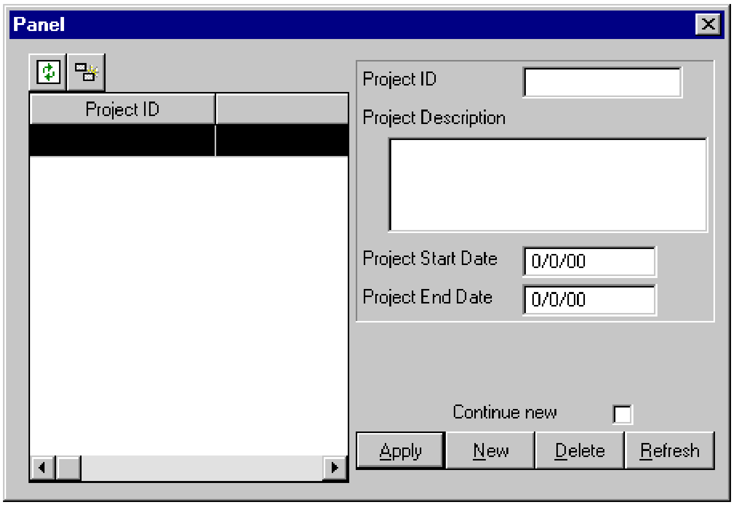


DetailP
Project ID
Left Label
Edit
Project Description
Left Label
Edit
Project Start Date
Project End Date

13. Holding the Ctrl key down, select the Project End Date field. This selects both of the fields; you can see this in the Design Window.

14. In the Design Window, drag the fields so they are below the bottom of the Project Description multiline edit box.

In the Design Window, you can see that the beveled box around the fields resizes to fit around all of the fields:



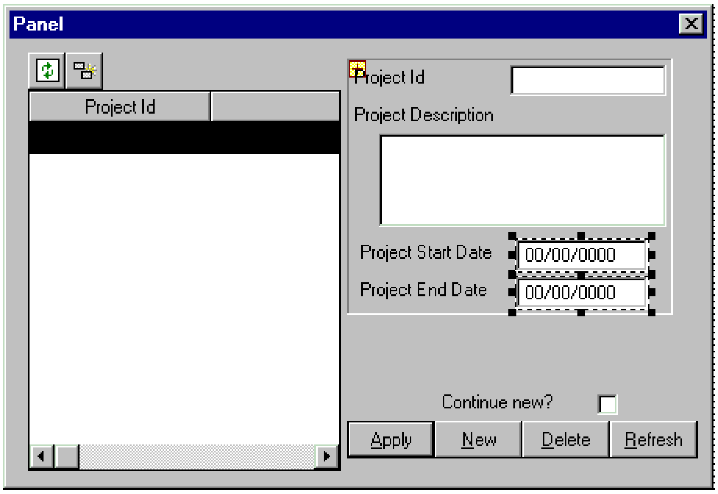
Panel
Project ID
Project ID
Project Description
Project Start Date 0/0/00
Project End Date 0/0/00
Continue new

## Adding Spin Controls

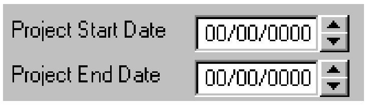
In the following steps, you modify the two date fields so that they include spin controls.

1. On the Design window, hold down Ctrl and click one of the edit controls of the date fields. Holding down Ctrl causes the control to be selected instead of the region in which it is contained.

2. Hold down Ctrl *and* Shift at the same time and click the other date control. Both date controls should now be selected and the Design window should look like this:



3. In the Property Sheet, locate the Spin Control property and set it to Yes.



Notice how you can select multiple elements on a panel and change properties for all of them.

4. From the File menu, choose Save.

5. Close the Design Window. This closes the whole Panel Designer.

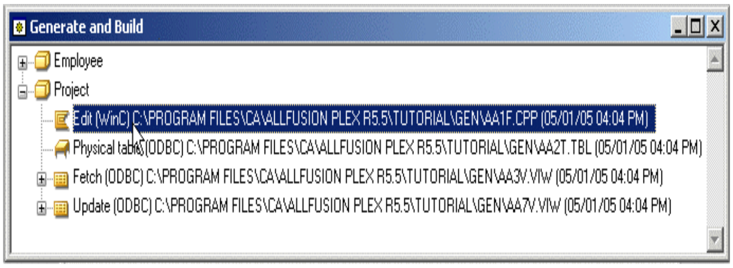
## Regenerating the Function

Now that you have modified the panel, you must regenerate and rebuild the function that displays it before you can see the changes.

To regenerate and rebuild the Project.Edit:

1. If you still have the Generate and Build window open, switch to it. Otherwise, click the New Gen & Build window toolbar button.

2. Select Project.Edit:



Employee
Project
Edit (WinC) C:\PROGRAM FILES\CA\ALLFUSION PLEX R5.5\TUTORIAL\GEN\AA1F.CPP (05/01/05 04:04 PM)
Physical table (ODBC) C:\PROGRAM FILES\CA\ALLFUSION PLEX R5.5\TUTORIAL\GEN\AA2T.TBL (05/01/05 04:04 PM)
Fetch (ODBC) C:\PROGRAM FILES\CA\ALLFUSION PLEX R5.5\TUTORIAL\GEN\AA3V.VIW (05/01/05 04:04 PM)
Update (ODBC) C:\PROGRAM FILES\CA\ALLFUSION PLEX R5.5\TUTORIAL\GEN\AA7V.VIW (05/01/05 04:04 PM)

   To rebuild a panel, you regenerate and rebuild the function that scopes it. Because you have only modified the Project.Edit.Panel object, you only need to regenerate and rebuild Project.Edit. If you had made changes that affected the scoped table or other objects, you would need to regenerate and rebuild those other objects too.

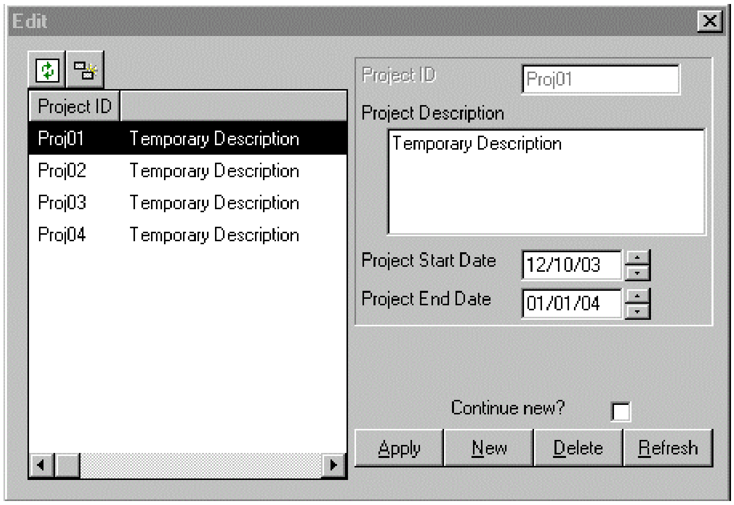
3. Click the Generate and Build toolbar button.

4. Click Yes when prompted to generate, and again when prompted to build.

5. When the generate and build process has completed, choose Project.Edit in the Generate and Build window and click the Run toolbar button.

6. When prompted to choose a data source, choose Plex Tutorial.dsn.

   The modified dialog appears, showing the data you already entered:



| Project ID | |
|---|---|
| Proj01 | Temporary Description |
| Proj02 | Temporary Description |
| Proj03 | Temporary Description |
| Proj04 | Temporary Description |

Project ID: Proj01
Project Description: Temporary Description
Project Start Date: 12/10/03
Project End Date: 01/01/04
Continue new? ☐

7. Select the text in the Project Description field for Proj01, and replace it with a longer description. For example:

   **Create a database application for the McCready account. Contact is Jim Hauser, 415-555-3146.**

8. Click Apply.

   Notice that the description is changed in both the grid and detail regions.

9. Select Proj02 and repeat this process to change its description to:

   **Create an email client for internal support staff.**

   and change the Proj03 description to:

   **Convert the ODBC version of the McCready application to an AS/400 version.**

10. Close the application.

## Review

In this chapter, you:

- Learned how to use the Model Editor to add triples
- Inherited from pattern library objects to define fields, and data access and user interface functions
- Generated and built the Project entity’s database table and the inherited functions, and added some sample data
- Used the Panel Designer to modify the inherited panel layout
- Regenerated and rebuilt the affected objects to see your changes

This chapter introduced the following patterns (and their scoped objects):

| Pattern | Description |
|---|---|
| FIELDS/Identifier | A 10-character data field. |
| FIELDS/LongDescription | A variable-length character field. |
| DATE/CheckedDateISO | A date field, where the date is stored in the ISO standard for dates. Verifies that dates entered are valid. |
| FOUNDATION/EditDetail | Lists database records and lets you edit, add, change, and delete records in a single dialog. |

This chapter introduced the following triples:

| Triple | Description |
|---|---|
| ENT **known by** FLD | Defines the primary key of an entity |
| ENT **has** FLD ...**optionality** SYS | Defines a data attribute of an entity and whether it is mandatory or optional to provide a value for a field |
| ENT **is a** ENT | The source entity inherits the properties of the target entity |
| FLD **is a** FLD | The source field inherits the properties of the target field |
| TBL **implement** SYS | Specifies whether to generate and build a database table |

Chapter 4

# Group Model Licensing

Group model licensing is an optional feature that enables Independent Software Vendors (ISVs) to protect their group models from unauthorized use. Computer Associates and its business partners will have informed you which of their group models are licensed during the sales transaction.

The first time you reference or open a licensed group model, the licensing dialog appears identifying the group model as licensed. The licensed group models require an Authorization Code.

If the licensed group model author enables the trial license feature, you will receive a trial license for a temporary period. You must request and enter your Authorization Code before this period ends. If enabled, the trial license feature enables you to immediately use the group model while you wait for your Authorization Code. If this feature is not enabled, you cannot use the model until you receive your Authorization Code. After you receive your Authorization Code, enter it in the Licensing dialog.

You can get your Authorization Code using the email or fax facility in the AllFusion Plex Licensing dialog.

The information about whom you should get your Authorization Code from is stored in the licensed group model. If you use the email or fax facility, this information automatically appears. Choosing About Group Model from the Help menu displays the same information.

To navigate to a licensed group model's licensing dialog, see Accessing the AllFusion Plex Licensing dialog. Use these instructions, where applicable, to maintain the license of a licensed group model.

## Local Model Licensing

In addition to the group model license, the vendor can opt to secure any local models that use a licensed group model. This means that a license check is performed each time a local model is opened that uses objects from the licensed group model. You must have a valid authorization code on your workstation to use such local models.

## Accessing the Group Model Licensing Dialog

Use the Group Model Licensing dialog to enter your Authorization Code for a licensed group model.

To access the Group Model Licensing dialog:

1. Start AllFusion Plex.
2. Log in to your host group model.
3. On the Group Model window, select the licensed group model and choose About Group Model from the Help menu.
4. Click License.

If a licensed group model is unauthorized, this dialog automatically appears the first time the model is opened.

## Transferring a Group Model License

Group model licenses are highly secure and specific to a particular PC. You cannot use the same authorization code on more than one PC. Instead you can transfer a group model license to another workstation. You can either transfer the license directly across the network or indirectly using a floppy disk.

### Direct Transfer

Use this method to transfer a license to an unlicensed version of the group model that exists on a different PC on your network.

To perform a direct transfer:

1. On the target PC, attempt to access the target copy of the group model (the unlicensed version) at least once, thus creating the license directory. Be sure it is closed before performing the license transfer.

   ***Important!*** *Never transfer a license to an open copy of AllFusion Plex; it disables the license.*

2. On the target PC, share the license directory of the target copy of AllFusion Plex (Windows\Oblicense) with full permissions, to let the license files to be copied.
3. Map the directory path for the target copy of AllFusion Plex license directory.

   **Note:** UNC path names are not supported; for example, \\MyServer\Windows\Oblicense. You must map to the target PC using a drive letter, for example, K:\Windows\Oblicense.

4. On the source workstation, log in to your host group model.
5. On the Group Model window, select the licensed group model and choose About Group Model from the Help menu.
6. Click License, then click Direct License Transfer.

7. In the Direct License Transfer dialog, enter the path to the target PC's Windows\Oblicense directory (remember the path must begin with a drive letter).
8. Click OK to transfer the license.

A message box displays the results of the transfer.

## Floppy Disk License Transfer

Use the floppy disk method to transfer the license between two PCs that cannot communicate over the network.

**Note:** When transferring the license to a different PC on your network, Computer Associates recommends using the direct transfer method.

Using this method requires advance planning. Before you can begin the transfer process, the trial license on the target PC (unlicensed version) must be expired. This is because the target PC needs to be registered before you transfer the license, and that cannot be done until the trial license has expired.

For this transfer, you need an ordinary floppy disk with 180 bytes of free space.

The following scenario explains how to transfer a license from your office PC to your home PC, and back again to the office PC.

### Preparing for the Transfer

Before you can transfer your license, do the following:

1. Install AllFusion Plex on your office PC and enter the authorization code for the group model.
2. Install AllFusion Plex and the group model on your home PC and attempt to access the group model. Wait until the trial license expires before continuing.

## Transferring the License

Use the following procedure for transferring a group model license from your office PC to your home PC.

On your home PC:

1. Start AllFusion Plex and attempt to access your target group model. This creates a new set of license files in the Oblicense directory, which are necessary to register the home PC.
2. Insert a blank disk into your home PC and create two directories: A:\HOME and A:\OFFICE.

   **Note:** You can name these directories anything you want. For these instructions, we are using HOME and OFFICE directories.
3. Log in to your host model, select the license group model and choose About Group Model from the Help menu.
4. Click License, and then click Transfer in the Group Model Licensing dialog.
5. In the AllFusion Plex License Transfer dialog, click Register Target PC.
6. In the Register License Transfer dialog, enter A:\HOME, and then click OK.

Your home PC is now registered. A registration file is placed in the HOME directory. The Office directory is still empty.

On your office PC:

1. Insert the disk into your office PC. Start AllFusion Plex and access the AllFusion Plex License Transfer dialog.
2. In the AllFusion Plex License Transfer dialog, click Transfer License to Floppy.
3. In the Transfer License to Floppy dialog, enter A:\HOME, and then click OK.

   A message box displays the status of the transfer.

4. Click OK.

   The group model on your office PC is now unlicensed—the license is on the disk. To return the license to this machine the next day, you must register the office PC now.

5. While still in the AllFusion Plex License Transfer dialog, perform the registration procedure to register your office PC.

6. In the Register License Transfer dialog, enter A:\OFFICE, and then click OK.

Both directories on the disk now contain data. The HOME directory contains the license. The OFFICE directory contains the office PC's registration information. The registration information is required for returning the license to your office PC.

**Note:** At this point, you cannot register the office PC again. If you try to register it again the following message appears:

```
Source directory already contains registration file.
```

It is important to note that you cannot transfer a different license to the office PC at this time. If you do, you will be unable to transfer the original license back to the office PC. This is because the Site Code on the office PC changes when you transfer a different license to the PC. If you attempt to transfer the original license back to the office PC, it will not work with the new site code.

On your home PC:

1. Insert the disk in your home PC and access the Group Model Licensing dialog.

2. Click Transfer.

3. In the AllFusion Plex License Transfer dialog, click Transfer License to PC.

4. In the Transfer License to PC dialog, enter A:\HOME, and then click OK.

   The license on the disk is transferred to your home PC. The HOME directory is now empty and your office PC is not licensed.

5. To return the license to your office PC, transfer the license to the OFFICE directory.

# Chapter 5 Pattern Libraries

Pattern libraries are sets of reusable design objects on which you can base the applications you develop.

AllFusion Plex provides the following sets of pattern and class libraries:

- Pattern libraries
- Class libraries (OBASE family)

## Version and Level Names

The pattern libraries accompanying AllFusion Plex r5.5 SP1 are called V5.2 Patterns.

## Installing the Pattern Libraries

The pattern libraries are included in the Typical installation of the AllFusion Plex base product. They are not included with the Compact installation. For installation instructions, see the chapter “Installing AllFusion Plex” in this guide.

## Installing the Class Libraries

To install the class libraries you must perform a Custom installation (or modify your existing installation). For installation instructions, see the chapter "Installing AllFusion Plex" in this guide.

For information on using the class libraries, see the CLASSLIBS.HLP help file.

## Installing the Pattern Libraries on a Network

In a typical workgroup environment (see Typical Configuration in "Installing AllFusion Plex"), the pattern libraries are installed on a network drive where they are accessible by all developers. This ensures that all developers use the same version of the libraries, thus simplifying model administration.

You can install the pattern or class libraries on a network server:

1. On the server machine, run the setup program (see Installing the AllFusion Plex Base Product in "Installing AllFusion Plex").
2. Specify the directory where the libraries are to be installed.

   **Note:** You can use a UNC path (such as \\MyServer\MyShare\Libraries) as the location for the libraries. This means that all developers accessing the libraries on your network can also use the same path. The use of UNC paths is optional. If you need to access a library, and the path name specified at installation time is not recognized, AllFusion Plex prompts you to enter a valid path.
3. Select the Custom installation option.
4. Select the libraries that you want to install and clear the other options.

Chapter

6

# AS/400 Components

This chapter describes how to install and configure the iSeries (AS/400) components—specifically for the AS/400 client/server, AS/400 5250, and Java client—AS/400 server products.

**Note:** AllFusion Plex also supports the development of Java Server applications on the AS/400. For setup instructions, see the chapter "Java Components."

## What Is Shipped to You

The following AllFusion Plex for AS/400 Client/Server and AllFusion Plex for AS/400 5250 product libraries are located on the AllFusion Plex Product CD as binary files:

- PLEX550 library containing the remote configuration and run-time objects
- YTUTORIAL (AllFusion Plex Tutorial library)
- YTUTREFER (AllFusion Plex Reference Tutorial library)
- APPINTOBJ (AllFusion Plex Application Integrator library)

**Note:** If you purchased the AllFusion 2E Data Migration product, do not use these installation instructions. For installation instructions, see Restoring the Product Libraries in this chapter.

## Minimum AS/400 Development Requirements

The minimum requirements for the AS/400 development environment include:

- PLEX550 library containing the remote configuration and run-time objects.
- OS/400. For supported version information, see the readme.
- RPG/400 compiler and/or ILE RPG/400 compiler.

  Functions defined with the RPG400 (or SQLRPG400) system value are generated with RPGIII syntax and require the RPG/400 compiler.

  Functions defined with the RPGIV (or SQLRPGIV) system value are generated with RPGIV syntax and require the ILE RPG/400 compiler.
- DDS compiler.
- UIM compiler (for AS/400 5250 generator only).
- QSYS2 library for AS/400 CPIC communications in the system portion of your library list.
- 4 MB of disk space for the AllFusion Plex product (for application development and run-time execution).
- Sufficient space for generated and compiled AS/400 objects.
- A PC that meets the requirements of the PC Development Environment (AS/400 Client/Server applications only).

## Minimum iSeries Deployment Requirements

This section lists the requirements for an AS/400 running:

- An AS/400 5250 application
- An AS/400 client/server application
- A Java-RPG/400 application

The components you need are:

- 5250-type terminals, or a PC running a 5250 emulation package (AS/400 5250 applications only)
- OS/400
- The following libraries must be in your library list when you call your function:
  - A data library, where your physical and logical files reside
  - A generation library, where your generated programs reside
  - The PLEX550 library

    **Note:** You can combine your data library and generation library into one library.

## Library List Considerations

At run time, the PLEX550 library should be lower than your data and generation library in your library list.

## Transferring the Product Libraries from CD to AS/400

Before restoring the AllFusion Plex AS/400 product libraries, you must first transfer the files from the CD to your AS/400. These instructions assume you have TCP/IP installed, configured, and active on your AS/400; otherwise, you cannot continue. Either contact your AS/400 System Administrator about installing TCP/IP, or order the AllFusion Plex AS/400 product tape.

To transfer AS/400 (binary) files from the CD to your AS/400:

1. Get the IP address (for example, nnn.nnn.nnn.nnn), or the symbolic name (myas400.mycompany.com), of your destination AS/400 from your AS/400 System Administrator.
2. Log on to your PC.
3. Insert the AllFusion Plex CD into an accessible CD-ROM drive.
4. Open a command line window on your PC.
5. Change drives to the location of your CD-ROM drive.
6. Type **cd as400libs** and press Enter.
7. Type **ftp** and press Enter. This starts an FTP session.
8. Type **open nnn.nnn.nnn.nnn** (where nnn.nnn.nnn.nnn is your AS/400 TCP/IP address—you can substitute the symbolic name for the IP address) and press Enter.

   You are successfully connected if you are prompted for your user profile.

   **Note:** Once connected, the following message appears:
   ```
   Connection will close if idle for more than five minutes.
   ```

   If the message, Connection closed by remote host appears, the session has been idle for too long, and you must start over at Step 7.
9. Type your AS/400 user profile at the prompt and press Enter.

10. Type your AS/400 password (the cursor does not move) at the prompt and press Enter.

    The reply message, xxx logged on (where xxx is your user profile), appears.

11. Type **quote site namefmt 1** (there must be a blank space between namefmt and 1) and press Enter.

    The following reply message appears: Now using naming format "1".

    **Note:** You must specify 1 as the naming format.

12. Type **bin** and press Enter to set the binary format.

13. Type **cd qgpl.lib** (you can specify another existing library in place of QGPL; the .lib extension is required), and press Enter.

    The following reply message appears: Current library changed to QGPL.

14. Enter **put plex550.savf** and press Enter (the .savf extension is required). The time to transfer the file varies depending on your network speed.

    Along with other information, the following reply message appears: File transfer completed successfully.

    **Note:** The .savf extension must appear as shown in the example.

15. Repeat Step 14 for ytutorial.savf, ytutrefer.savf, and appintobj.savf.

16. Enter **quit** and press Enter. This will stop your FTP session.

17. Quit the MS-DOS session.

18. You can now restore the *SAVF files.

## Restoring the Product Libraries from Save Files (*SAVF)

To install the AllFusion Plex for AS/400 client/server or AllFusion Plex for AS/400 5250 components from save files:

1. Log on as QSECOFR, or with QSECOFR equivalent authority.

2. If you are upgrading an existing release of AllFusion Plex, first clear the product libraries by entering the following command strings:

   ```
   CLRLIB LIB(PLEX550)
   CLRLIB LIB(YTUTORIAL)
   CLRLIB LIB(YTUTREFER)
   CLRLIB LIB(APPINTOBJ)
   ```

3. Restore the AllFusion Plex product library as follows:

   ```
   RSTLIB SAVLIB(PLEX550) DEV(*SAVF) SAVF
   (QGPL/PLEX550) FRCOBJCVN(*YES *RQD)
   ```

4. Restore the Tutorial library by entering:

   ```
   RSTLIB SAVLIB(YTUTORIAL) DEV(*SAVF)
   SAVF(QGPL/YTUTORIAL)
   ```

5. Restore the Tutorial library by entering:

   ```
   RSTLIB SAVLIB(YTUTREFER) DEV(*SAVF)
   SAVF(QGPL/YTUTREFER)
   ```

6. Restore the Application Integrator library by entering:

   ```
   RSTLIB SAVLIB(APPINTOBJ) DEV(*SAVF)
   SAVF(QGPL/APPINTOBJ)
   ```

7. Add PLEX**550** to the top of your library list.

## PC-to-AS/400 Communications Software

For AS/400 development, AllFusion Plex requires your PC to be linked to an AS/400. The link between the PC and the AS/400 can be through native AS/400 TCP/IP support. See the readme file for the recommended PTFs related to the TCP/IP connectivity for details.

## Configuring the AllFusion Plex TCP/IP Environment

The following guidelines explain how to configure AllFusion Plex TCP/IP protocol services according to the needs of both your developers and end users. TCP/IP does not use, or require, the QCMN subsystem. Therefore, these instructions are based on QCMN not being present.

To use TCP/IP, you must have:

- OS/400

  For additional information on configuring TCP/IP for your AS/400, see the IBM publication, *TCP/IP Configuration and Reference* (SC41-34209-00). These instructions presume that TCP/IP is installed and active on your AS/400.

- TCP/IP configured and started on your AS/400
- TCP/IP version of the AS/400 AllFusion Plex Dispatcher started on the AS/400

For AllFusion Plex clients to connect using TCP/IP you must have:

- The TCP/IP AllFusion Plex Dispatcher started on the AS/400
- A port number assigned when starting the TCP/IP dispatcher program

## Selecting Which TCP/IP Dispatcher Program to Use

AllFusion Plex includes two different AS/400 dispatcher programs called YOBSYTCPDP and YOBSYTCP.

YOBSYTCPDP is recommended because it offers improved security. Specifically, this dispatcher uses a single socket for all communications between the client and the server so that only the specified port needs to be open through a firewall.

One limitation of the YOBSYTCPDP dispatcher is that it does not display the actual user profile for the Job User parameter of the spawned job name. Instead, it displays the user profile that the dispatcher job is running under (and so it is the same regardless of the actual user profile). This is a limitation of the spawn API used by the YOBSYTCPDP dispatcher that does not provide a USER parameter. In contrast, the YOBSYTCP dispatcher uses the SBMJOB command that does provide a USER parameter.

## Starting the TCP/IP Dispatcher Program

This section provides examples on how you can start the TCP/IP dispatcher program. Before you execute the examples, review the following important considerations:

- An alternative version of the YOBSYTCP dispatcher is provided in the PLEX library that uses RTGDTA(*JOBD) when submitting jobs. By default, this alternative version is named YOBSYTCP_R. To use the alternative version of the dispatcher it should be renamed.  For example, the following commands rename the original dispatcher to YOBSYTCP_X and YOBSYTCP_R to YOBSYTCP.

  ```
  RNMOBJ OBJ(PLEX550/YOBSYTCP) OBJTYPE(*PGM) NEWOBJ(YOBSYTCP_X)
  RNMOBJ OBJ(PLEX550/YOBSYTCP_R) OBJTYPE(*PGM) NEWOBJ(YOBSYTCP)
  ```

- If you are using OS/400 V5R3, you need to perform the following steps before installing your dispatcher. The names of the V5R3 objects start with the character V. You need to rename these objects to Y as shown in the example that follows. Before renaming these V objects, rename or move the existing Y objects (we do not recommend deletion of these objects).

  For example:

  ```
  RNMOBJ OBJ(PLEX550/YOBSYTCP) OBJTYPE(*PGM) NEWOBJ(XOBSYTCP)

  RNMOBJ OBJ(PLEX550/YOBSYTCP_R) OBJTYPE(*PGM)
  NEWOBJ(XOBSYTCP_R)

  RNMOBJ OBJ(PLEX550/YOBSYTCPDP) OBJTYPE(*PGM)
  NEWOBJ(XOBSYTCPDP)

  RNMOBJ OBJ(PLEX550/VOBSYTCP) OBJTYPE(*PGM) NEWOBJ(YOBSYTCP)

  RNMOBJ OBJ(PLEX550/VOBSYTCP_R) OBJTYPE(*PGM)
  NEWOBJ(YOBSYTCP_R)

  RNMOBJ OBJ(PLEX550/VOBSYTCPDP) OBJTYPE(*PGM)
  NEWOBJ(YOBSYTCPDP)

  RNMOBJ OBJ(PLEX550/VOBSYTCPDP) OBJTYPE(*PGM)
  NEWOBJ(YOBSYTCPDP)
  ```

- The YOBLISTEN job runs until it is ended manually or the system IPLs. Set the SCDTIME parameter to a time after the IPL, and after the STRTCP command was issued and has completed TCP/IP startup.

- It is not necessary to set the USER parameter to QSECOFR, but the user parameter you supply must have the following authorities:

  - *ALLOBJ authority is not required, however, *USE authority for each AS/400 user profile that initiates a client connection request is required for validating AS/400 user profiles and passwords.  If *ALLOBJ is not used, connecting clients cannot reset their own expired passwords.

  - *USE authority for objects QSYS/QSYGETPH and QSYS/QSYRLSPH.

- The parameter, 55000, is the assigned port. However, it can be any unused port number that is not in the range of port numbers previously registered with the Internet Assigned Number Authority (IANA). These registered ports are typically within the range of 0 to 1023.

  We recommend using port numbers between 3000 and 60000 to avoid conflicts with lower numbers as they become registered.

**Example 1** Write your own CL program to start PLEX550/YOBSYTCPDP or PLEX550/YOBSYTCP, and call the CL program as part of your IPL.

**Example 2** Create a Job Schedule Entry (ADDJOBSCDE).

The following is an example you could use if your IPL happened nightly:

```
ADDJOBSCDE JOB(YOBLISTEN) CMD(CALL )) FRQ(*WEEKLY)
PGM(PLEX550/YOBSYTCPDP) PARM('55000'
SCDDATE(*NONE) SCDDAY(*ALL) SCDTIME('hh:mm:ss')
JOBD(PLEX550/PLEX) JOBQ(QGPL/QINTER) USER(QSECOFR)
TEXT(Plex TCPIP C/S Dispatcher')
```

**Note:** The PARM value in the previous statements must include the single quotes or the call will fail.

After submitting the ADDJOBSCDE command, the Dispatcher will not start until the scheduled time after the next IPL.

To start the Dispatcher before the next IPL:

1. From the command line, enter:

   ```
   WRKJOBSCDE
   ```

2. Enter 10 next to the job, and press Enter.

The job will start immediately (typically you would only do this once on the day you set up the Dispatcher).

## Starting Additional TCP/IP Dispatchers

You can start additional AllFusion Plex TCP/IP Dispatchers on additional ports at any time for testing purposes. For instance, when first configuring AllFusion Plex for TCP/IP to the AS/400, you may want to start the Dispatcher manually before adding the job to your startup routine.

To manually start the AllFusion Plex TCP/IP Dispatcher on a port:

1. From the command line, enter:

   ```
   SBMJOB CMD(CALL PGM(PLEX550/YOBSYTCPDP) PARM('55000'))
   JOB(YOBLISTEN) JOBD(PLEX550/PLEX)
   JOBQ(QGPL/QINTER)
   ```

   The parameter passed into the YOBSYTCPDP (or YOBSYTCP) program must be an available, unused port number.

2. To verify that your port has started, from the command line, enter:

   **WRKTCPSTS *CNN** (check the Local Port column for the port number you assigned)

**Note:** Use QINTER as the job queue so that incoming AllFusion Plex clients get interactive response times. The job queue that starts the AllFusion Plex TCP/IP Dispatcher is used for submitting individual client jobs as they are requested. Using a batch job queue may cause your AllFusion Plex client jobs to sit in your batch job queues, subsequently causing a time-out and hanging behaviors on the client side.

## Restoring the Product Libraries Under a Different Name

You can choose to restore the product library under a different name instead of the default PLEX550. If so, you will need to manually update several references to the library name as described in the following sections.

### Updating the Job Description

After restoring the library, enter the following command to update its JOBD:

```
CHGJOBD JOBD(library-name/PLEX)
INLLIBL(QTEMP library-name YTUTORIAL YTUTREFER APPINTOBJ
QGPL)
```

### Modifying Files on the PC

The following instructions explain how to change certain default settings in AllFusion Plex to use the name of the AllFusion Plex AS/400 product library that you specified (PLEX550 in this example) instead of the default name (PLEX).

Using a text editor, such as Wordpad, modify the following files on the PC to reflect the unique AS/400 library names that you used for the new version:

**Note:** These instructions use C:\Plex550 as the new product directory name on the PC, and PLEX550 as the new AS/400 library name.

- C:\Plex550\Plex.bld

  Under the section titled [remote configuration], change the line

  ```
  jobdesclib=PLEX550
  ```

  to:

  jobdesclib=*library-name*

- C:\Plex550\Plex.INI

  Under the section titled [Remote], change the line:

  ```
  Library=PLEX550
  ```

  to:

  ```
  Library=library-name
  ```

- C:\Plex550\Bin\Obsyrt.ini.INI

  Under the section titled [Remote], change the line:

  ```
  Library=PLEX550
  ```

  to:

  ```
  Library=library-name
  ```

Using the same change for Obsyrt.ini, modify:

- \Bin\Ob550rc.INI
- \Bin\Ob550rcd.INI

## System Administrator Guidelines for Multiple Versions

The following guidelines tell you how to configure AllFusion Plex services for multiple versions according to the needs of your developers and end users.

### Configuring with Prestart Jobs

To decrease the amount of time OS/400 takes to start and initialize an AS/400 server job, you have the option of using AS/400 prestarted jobs. These jobs are prestarted for communication requests when the subsystem starts.

## Installing Additional Prestarted Jobs for Multiple Versions

If you are installing an additional AllFusion Plex run-time library and you are using the AS/400 standard QCMN subsystem for your communication jobs, you need to add an additional job entry for AllFusion Plex server jobs.

To make these additions:

**Note:** This procedure uses PLEX550 as the additional AllFusion Plex library.

1. Sign on with QSECOFR authority.
2. Add PLEX550 to your library list.
3. From the command line, enter:

   ```
   ADDPJE SBSD(QCMN) PGM(PLEX550/YOBSYRPC)INLJOBS(3)
   JOB(YOB550) CLS(QGPL/QINTER)
   ```

4. Verify that prestarted jobs are configured correctly by displaying the active jobs in the QCMN subsystem.

   When QCMN begins, six prestarted jobs are started. Three of these jobs are for the PLEX library and three are for the PLEX550 library. The three PLEX jobs are identified by the YOBSYRPC job names. The three PLEX550 jobs are identified by the PLEX550 job names.

For more information on prestarted jobs, see IBM's *Application System/400 Work Management Guide*.

**Note:** It is not necessary to add another Routing Entry. The Routing Entry that was added when PLEX550 was installed services all AllFusion Plex job requests. The different job names do not have an impact because the compared value is YOBSYRPC, which is the RPC program name in both libraries.

## Considerations and Background

Here are some important points to consider:

- To achieve the best possible response times, we recommend that all server jobs run with an interactive (QINTER) communications class.
- Ensure that the subsystem, such as QCMN, has enough memory and resources to give server jobs adequate resources for processing client requests.
- Configuration discussed here does not apply to configuring AllFusion 2E Data Migration.
- One AS/400 server job is started for each Windows client session.

Chapter 7

# Open Database Components

This chapter describes how to configure various components for the AllFusion Plex Open Database generator.

## Before You Begin

Using AllFusion Plex for Open Database, you can design and generate applications that are ODBC-enabled. It is possible to design an application without specifying the database management system (DBMS) in your design. When you generate or deploy your application, you choose a third-party DBMS product using the Microsoft Open Database Connectivity (ODBC) interface.

The Database and ODBC Drivers section in the online help identifies the DBMS products and ODBC drivers that have been tested with the Open Database generator.

To use the Open Database generator with AllFusion Plex, you need to configure:

- The environment
- A database (DBMS product)
- An ODBC driver
- A data source

The data source comprises:

- The data you want to access
- The DBMS you will use
- The platform on which that DBMS resides
- Any network needed to access the information

You need to connect to a configured data source to build the tables and views you design in your application, and to access the data in your database.

For instructions on configuring the environment and the data source, see Connecting to ODBC in the online help.

## Configuring the Database and ODBC Driver

To use the AllFusion Plex for Open Database generator, you need to install and configure an ODBC-compliant Database Management System (DBMS) and an ODBC driver. Install these products on your PC or a network server that several developers or end users can access. Refer to specific vendor documentation for installation instructions.

## Databases

A DBMS lets you store, delete, modify, and retrieve data stored in the database. If the DBMS product you select has an ODBC-compliant driver, you will be able to use this product with the Open Database generator, provided that the database supports the features you designed in your application. Some of the DBMS products are very powerful and offer a wide range of features; others provide less support.

After the DBMS and the ODBC driver are installed, configure the data source through the Windows ODBC Administrator. For more information, see Configuring a Data Source in the online help.

Most third-party databases are installed with a basic set of configuration options. Some of the configuration steps required for your target database may include the following:

- Installing any necessary communications software that provides users with access to the database
- Configuring startup files
- Setting up security procedures
- Enrolling all users in the database
- Setting up appropriate authority levels
- Setting up maintenance features

If you have installed the database on a network server, you need to configure a network connection to the machine on which the database resides. Ensure that the ODBC driver that you use for access executes over the network you select.

Chapter 8

# Windows Server Components

This chapter describes how to configure the Windows Server components for AllFusion Plex.

## Windows Server Development Requirements

The hardware and software requirements and recommendations for the Windows Server are listed in the following sections.

### Hardware

The hardware requirements for the Windows Server are:

- Intel Pentium 1 GHz PC (or higher)
- 1 GB RAM (minimum)
- 20 GB hard drive. At least 1 GB of free space recommended
- CD-ROM drive

### Software

The software requirements for the Windows Server are:

- Windows 2000 or Windows 2003 (for additional information on Windows Server 2003 support, see the readme)
- Microsoft Visual Studio .NET 2003  (Professional Edition or Architect Edition)
- AllFusion Plex Windows Application Server
- Microsoft SQL Server or Oracle (for detailed version information, see the readme)

For additional information, see PC Development Environment.

## Installing the Windows Server Components

To develop Windows server applications you must install the development version of the AllFusion Plex Application Server.

### Windows Client Requirements

You need to run the AllFusion Plex installation program on each developer's machine to install the client components. A typical installation of AllFusion Plex includes the necessary components.

In addition, the AllFusion Plex Application Server must be installed on the server machine.

## Installing the Application Server

To install the Application Server on a Windows 2003 or Windows 2000 operating system, the following authorities are required:

- Local Administrator authority is required if AllFusion Plex has previously been installed anywhere in the domain.
- Domain Administrator authority is required if AllFusion Plex has not been previously installed. The server must be part of a domain.

To install the Application Server when you install the AllFusion Plex base product:

1. When asked to specify which type of setup to install, click Custom.
2. Check AllFusion Plex Windows Application Server in addition to what is already checked.

To install the Application Server after you install the AllFusion Plex base product:

1. Click Start, then choose Settings, Control Panel.
2. On the Control Panel, double-click the Add/Remove Programs icon.
3. In the list of products, select AllFusion Plex.
4. Click Change/Remove.
5. On the InstallShield Wizard window, select Modify and click Next.
6. Select the AllFusion Plex Windows Application Server check box.
7. Click Next to start the installation.

## About the Application Server

The *Application Server* is the name for the services and components that support AllFusion Plex applications on Windows servers.

The Application Server includes:

- Environment Manager
- Build Service and Build Service Manager
- Dispatch Service and Dispatch Service Manager
- Printing Service

The standard AllFusion Plex installation process installs these components in the AppServer\Bin directory.

### Job Status Database

To support the Build Service, the Application Server installation program automatically:

- Creates a Microsoft Access database for storing status information about AllFusion Plex builds. This job status database is located at \AppServer\Jobsts\Objobsts.mdb.
- Configures an ODBC data source for the job status database.

### Configuring the Build Service and the Dispatch Service

You have the option of configuring the startup information for both the installed services. The default installation sets the startup type to be manual. You can change this to automatic so that the services start at boot up.

For more information, see the online help (Ntservices.chm).

## Oracle Support

You can use the Windows Server (WinNTC) generator to create applications that access data in Oracle databases on Windows Servers. The run-time application uses the native Oracle Call Interface (OCI) for fast data access.

Before using AllFusion Plex, ensure that Oracle is properly installed on the server (see your Oracle documentation for details), and that a basic user (like the default user SCOTT) has been configured with the following privileges:

- Connect
- Resource
- Table space

## Installing the Windows Client Components—Microsoft RPC

Consider the following before installing the Windows Client Components—Microsoft RPC:

- AllFusion Plex clients connect to the Application Server using the Microsoft RPC run-time services.
- Using the AllFusion Plex Environment Manager, create a user profile with proper authorities on the Windows Server for each AllFusion Plex developer. For more information, see Creating User Environments in the online help.
- For information on setting up a database, authorizing users, and configuring a data source name, see your SQL Server documentation.

## Deploying Windows Server Applications

The AllFusion Plex installation has a Windows Server Deployment option. When selected, this installs only the components necessary to deploy your application (the Dispatch Service, Printing Service, and the Environment Manager).

Chapter 9

# Java Components

This chapter describes how to install and configure the Java components for AllFusion Plex. For additional help, see the Java Platform section of the online help.

## Java Development Requirements

AllFusion Plex supports the generation of Java applications on both the client and server. Java applications can be developed and deployed in the following configurations:

- Java Client to RPG (iSeries) Server
- Java Client to Java Server
- Windows Client to Java Server

AllFusion Plex generates 100% pure Java code that, in principle, can be run under any Java Virtual Machine on any platform. If you have any existing Java applications generated by AllFusion Plex, run-time backwards compatibility enables new versions of the AllFusion Plex Java run time to be deployed without regenerating existing applications.

AllFusion Plex Java clients are primarily tested on Windows. Java client applets are tested using the Sun Java plug-in. Note that AllFusion Plex Java clients require the Sun Swing classes to support GUI components.

The following table summarizes the installation requirements for AllFusion Plex Java development. Deployment requirements are documented in the Java Platform section of the online help.

| **Component** | **Description** |
|---|---|
| Java client | AllFusion Plex Java components<br><br>Sun J2SE SDK 5.0. Other versions of the JDK work with AllFusion Plex, but they have not been fully tested. Freely downloadable.<br><br>NMAKE.EXE (Microsoft Program Maintenance Utility) Required for local Java builds and available with Microsoft Visual Studio .NET 2003 and with the Microsoft SDK For Java (downloadable free from www.microsoft.com/java/sdk/) |
| Java server | For Java server development:<br>■ AllFusion Plex Java components<br>■ Sun JDK (or equivalent). See Java client, above, for version information.<br>■ TCP/IP network connection<br>■ Third-party DBMS and JDBC driver (or ODBC driver with JDBC-ODBC bridge)<br><br>For iSeries Java server development:<br>■ PLEXJVA550 library<br>■ IBM Client Access (or equivalent)<br>■ IBM iSeries Toolbox for Java |

## Installing Java Components

The AllFusion Plex Java components are installed automatically when you install AllFusion Plex. They include:

- ObJava directory
- Lib and Remote subdirectories in the ObJava directory
- ob550bld.bat and ob550dsp.bat files in the ObJava directory

To install and configure the Java server components, do the following:

1. Configure the server for run-time and remote building.
2. Create a remote Job Status database, if one is not already created.
3. Start the Build Manager and Dispatcher.

### Configuring the Server for Run-Time and Remote Building

Copy the ObJava directory from the AllFusion Plex directory on your client machine to the remote server.

The Java Build Manager for AllFusion Plex and the Java Dispatcher for AllFusion Plex can be administered in the following ways:

- Using a web server that processes a shipped servlet
- Using a batch file that runs the Java VM interpreter within the session

Using the batch file is the simplest option. The use of a web server is more appropriate for production applications.

For instructions on starting the Build Manager and the Dispatcher, see Starting the Java Build Manager and the Java Dispatcher later in this chapter.

For information on web servers, see Setting up a Web Server in the online help.

## Creating a Remote Job Status Database on Windows

The Job Status Database is a database that maintains build status information. Whether you use a web server or the batch file option to manage the remote build manager, a job status database must be created.

If you are using a Windows server, you can use the database that the Application Server Build Service provides. For details on what is installed, see Installing the Windows Server Components later in this guide.

**Note:** If you already have a Job Status database and data source, skip this section.

Creating a Job Status database involves the following steps:

1. Creating a table.
2. Configuring a data source.
3. Editing the bld.properties file.

### Creating a Table

AllFusion Plex automatically creates the JobList table in the target database if it does not already exist.

### Configure a Job Status Data Source

If you are using the JDBC-ODBC bridge on Windows, follow these steps:

1. From the Windows Control Panel, double-click the ODBC icon and click the System DSN tab.
2. Click Add and select the required driver.
3. Click Finish.

   The ODBC Setup window appears.

4. Enter the following:
   - Data Source Name: **JavaBuild**
   - Description: Plex for Java Build Status

   Enter any other details required for your database.
5. Click OK.

### Edit the bld.properties File

Edit the following parameters in the Default section of the ob550bld.properties file (located in the Plex\ObJava directory) on the server:

```
Default.Build.Driver=NameofBuildDriver
Default.Build.DataSource=DataSourceName
Default.Build.User=UserName
Default.Build.Password=Password
```

## Setting Up Java on the AS/400

To run AllFusion Plex Java functions on the AS/400, you need to install the PLEXJVA550 library and the AllFusion Plex Java components onto your AS/400.

These instructions assume you are using the AS/400 Toolbox for Java. The use of IBM Client Access (or equivalent) is also recommended for accessing the Integrated File System (IFS) on the AS/400.

**Note:** If you are developing Java client to RPG server applications, you can ignore this section. To connect to RPG server functions on an AS/400, Java clients use the AS/400 TCP/IP dispatcher that is included in the PLEX550 product library. The installation process is the same as for Windows clients and is described in the chapter “AS/400 Components.”

## Installing the PLEXJVA550 Library

The PLEXJVA550 library is supplied as a save file in the AS400 Libraries folder on the AllFusion Plex CD-ROM. This needs to be transferred to the AS/400 and restored in the same way as the other AllFusion Plex AS/400 libraries. For details, see Transferring the Product Libraries from CD to AS/400 and Restoring the Product Libraries from Save Files (*SAVF) earlier in this guide.

## Copying the Objava Directory to the AS/400

To copy the Objava directory to the AS/400, perform the following steps:

1. Install the AllFusion Plex Java components onto a Windows machine, as described in Installing Java Components section in this chapter.
2. Create a directory at the top-level of the AS/400 IFS. In these instructions, we assume the directory name is Plex.
3. Copy the ObJava directory and its contents from your Windows machine to the AS/400, to create a Plex/ObJava subdirectory.

## Configuring the Java Build Manager and Dispatcher

The following instructions assume you are using the JDBC driver from the AS/400 Toolbox for Java. The Build Manager and Dispatcher must be configured with the necessary JDBC settings to ensure that the AS/400 Toolbox for Java classes are available on the class path.

To configure the Java Build Manager and Dispatcher, perform the following steps:

1. Create a directory at the top-level of the IFS to serve as the location for the build process. For example, /JavaBuild.

2. Change the settings in the ob550bld.properties file. Descriptions of these settings can be found in the AllFusion Plex help. A typical example follows with the necessary modifications appearing in bold.

   **Note:** In the following example, each entry should be on a single line.

   ```
   Default.Build.Compiler=javac
   Default.Build.ClassPath=/plex/jt400.jar:/plex/Objava/lib/
              obrun.jar:/qibm/proddata/java400/lib/java.zip
   Default.Build.SourceDirectory=/JavaBuild
   Default.Build.MaxCompiles=2
   Default.Build.MaxBuilds=2
   Default.Build.Driver=com.ibm.as400.access.AS400JDBCDriver
   Default.Build.DataSource=jdbc:as400://MACHINE/LIBRARY
   Default.Build.User=USERID
   Default.Build.Password=PASSWORD
   ```

3. Change the settings in the ob550dsp.properties file. Descriptions of these settings can be found in the AllFusion Plex help. A typical example follows with the necessary modifications highlighted. Each entry should be on a single line.

   ```
   Environment.Default.Driver=
       com.ibm.as400.access.AS400JDBCDriver
   Environment.Default.DataSource=
       jdbc:as400://MACHINE/LIBRARY
   Environment.Default.User=USERID
   Environment.Default.Password=PASSWORD
   ```

## Optimizing Java Classes

You must optimize the system .class files for them to perform well on the AS/400. This only needs to be done once.

You use the CRTJVAPGM command on a single file, an entire JAR file, or an entire directory (using wildcards). There are four levels of optimization you can choose. Unless you have a reason to do otherwise, use full optimization (level 40). The command creates an optimized hidden file for each .class file that is loaded by the AS/400 Java Virtual Machine. For more information on this command, see the AS/400 online help.

Run CRTJVAPGM as a batch process on the following files:

- obrun.jar
- JT400.jar (you can copy this file from the default location to the Plex directory you created earlier to make the path shorter)

## Starting the Java Build Manager and the Java Dispatcher

Add PLEXJVA550 to your library list. Then, use the YSTRJVABLD and YSTRJVADSP commands to start the Build Manager and Dispatcher, respectively. These commands include parameters for the port number and locations of the run-time and properties files. Use different port numbers for the Build Manager and the Dispatcher. The path to JT400.jar file should be referenced in the Path To Additional Class parameter.

See YSTRJVABLD or YSTRJVADSP in the Index of the AllFusion Plex online help for more details on these commands.

A second method of starting the services is by using the RUNJVA command. To use this method, enter **RUNJVA** on an AS/400 command line and press F4. For the Class parameter, type in the name of the run-time class that requires service:

- ObRun.ObComms.ObService - to start the Dispatcher
- ObBldMgr.ObBuildManager - to start the Build Manager

**Note:** The class names are case-sensitive.

For the Parameters parameter, type in the port number and the path to the properties file, for example, 1998 and /Plex/Objava. Then at the Classpath prompt, type in the full path to the AllFusion Plex Java run time, for example, /Objava/lib/obrun.jar.

# Chapter 10 AllFusion 2E Data Migration

This chapter describes how to install the AllFusion 2E Data Migration product.

## What Is Shipped to You

The AllFusion 2E Data Migration product includes the following migration libraries located in the AS400 Libraries directory on the AllFusion Plex product CD-ROM:

- YOMF
- YOMFVENG
- YOMFDB

## AllFusion 2E Data Migration Requirements

For PC and AS/400 prerequisites, see:

- PC Development Environment
- Minimum AS/400 Development Requirements.

This section describes only the prerequisites that are specific to AllFusion 2E Data Migration.

### Memory

The minimum memory requirement for AllFusion 2E Data Migration is 1 GB RAM.

Since Data Migration can populate large AllFusion Plex models quickly, consider increasing the amount of RAM on the PC used for development, if you are importing a large AllFusion 2E model.

### Minimum AS/400 Prerequisites

The requirements include:

- OS/400
- A PC-to-AS/400 communications software package (for specific requirements, see Upgrading from Earlier Releases of AllFusion Plex in the *Release Summary*.)
- 20 MB of disk space for the AllFusion 2E Data Migration product

## Restoring the Product Libraries

Use the following instructions to restore the AllFusion Plex objects from the product CD to the AS/400.

1. Transfer the migration files, YOMF.savf, YOMFVENG.savf, and YOMFDB.savf, from the CD to your AS/400. See Transferring the Product Libraries from CD to AS/400 if you need instructions.
2. If you are using the product CD, follow the steps in Restoring the Product Libraries from Save Files (*SAVF).

   Restore the files, YOMF.savf, YOMFVENG.savf, and YOMFDB.savf.

3. If you are upgrading an existing release of AllFusion Plex, note that the product library names for Data Migration have changed. Though it is not necessary for this installation procedure, you may want to delete the old libraries using the following command strings:

   ```
   DLTLIB LIB(YIMPS2OBSY)
   DLTLIB LIB(YIMPS2VENG)
   DLTLIB LIB(YOBSCS)
   DLTLIB LIB(YOBSCSVENG)
   ```

   **Note:** If you have not completed the steps in PC-to-AS/400 Communications Software, enter the following command strings:

   ```
   CLRLIB LIB(PLEX550)
   CLRLIB LIB(YTUTORIAL)
   CLRLIB LIB(YTUTREFER)
   ```

4. Restore the AllFusion Plex run-time libraries:

   If you are using V3R6 or later, enter:

   ```
   RSTLIB SAVLIB(PLEX550) DEV(device) ENDOPT(*LEAVE)
   RSTLIB(PLEX550) FRCOBJCVN(*YES *RQD)
   ```

5. Restore the Tutorial library by entering:

   ```
   RSTLIB SAVLIB(YTUTORIAL) DEV(device) ENDOPT(*LEAVE)
   ```

6. Restore the Tutorial Reference library by entering:

   ```
   RSTLIB SAVLIB(YTUTREFER) DEV(device) ENDOPT(*REWIND)
   ```

7. Add PLEX550 to the top of your library list.

8. Restore the Migration main product library as follows:

   ```
   RSTLIB SAVLIB(YOMF) DEV(device) ENDOPT(*LEAVE)
   RSTLIB(PLEX550) FRCOBJCVN(*YES *RQD)
   ```

9. Restore the Migration user interface library as follows:

   ```
   RSTLIB SAVLIB(YOMFVENG) DEV(device) ENDOPT(*LEAVE)
   RSTLIB(PLEX550) FRCOBJCVN(*YES *RQD)
   ```

10. Restore the Migration Mapping database as follows:

    ```
    RSTLIB SAVLIB(YOMFDB) DEV(device) ENDOPT(*LEAVE)
    RSTLIB(PLEX550) FRCOBJCVN(*YES *RQD)
    ```

## Parallel AllFusion 2E Installations

If you are working with parallel AllFusion 2E installations, you must adapt the YIMPS2OBSY/QBATCH job description library list to fit your environment. Adapt your job description to reference the correct libraries if you have:

- Placed AllFusion 2E objects in additional libraries
- Consolidated AllFusion 2E objects into fewer libraries
- Renamed your AllFusion 2E libraries

Chapter

# 11 Application Integrator

This chapter describes how to install Application Integrator. *Application Integrator* allows reverse engineering information from a variety of sources into AllFusion Plex.

There are several data sources that you can import into the Application Integrator. Each one is a separately licensed product.

- Schema/ODBC—enables the integration of data schemas from any ODBC data source. This replaces the previous ODBC Import Utility.
- Schema/400—enables the integration of a DDS-defined data schema from the AS/400. This replaces the previous DB2/400 Import Utility.
- Schema/Biz—enables the integration of a logical database that has been defined in COOL:Biz.
- Application Integrator/400—enables the integration of OS/400 programs (hand-coded or generated by other tools).
- Application Integrator/2E—enables the integration of AllFusion 2E functions direct from an AllFusion 2E model.

More information on using Application Integrator is available in the online help.

## What Is Shipped to You

The AllFusion Plex product CD includes:

- The Application Integrator tool
- The APPINT AS/400 library, installed as a save file in Application Integrator directory

## System Requirements

The requirements specific to Application Integrator are:

- PC Development Environment. Large imports may require more than the recommended minimum memory.

  **Note:** For more information about PC Development Environment, see System Requirements section in the readme.

- Minimum AS/400 Development Requirements, which include:
  - OS/400

    **Note:** For supported version information, see the readme
  - For AS/400 connections, a TCP/IP connection

    **Note:** For more information about Minimum AS/400 Development Requirements, see chapter "AS/400 Components" in this guide

## Installing the Application Integrator

To install the Application Integrator tool when you install the AllFusion Plex base product:

1. When asked to specify which type of setup to install, click Custom.
2. Check Application Integrator in addition to what is already checked.

To install the Application Integrator tool after you install the AllFusion Plex base product:

1. Click Start, then choose Settings, Control Panel.
2. On the Control Panel, double-click the Add/Remove Programs icon.
3. In the list of products, select AllFusion Plex.
4. Click Change/Remove.
5. On the InstallShield Wizard window, select Modify and click Next.
6. Select the Application Integrator check box.
7. Click Next to start the installation.

## Setting Up the ODBC Data Source

At the end of the installation process, you may see a warning message indicating that no directory has been specified for ODBC file DSNs on your system. If a warning message appears, complete the following steps manually:

1. Locate the file Appint.dsn in the Application Integrator directory.
2. Copy this file to the default directory on your system where ODBC file DSNs are stored. This is typically C:\Program Files\Common Files\ODBC\Data Sources but can be changed using the ODBC Administrator.

**Note:** If you did not see a warning message, these steps are not necessary.

## Preparing to Import from the AS/400

If you are importing from an AS/400-based data source, you must first install the AllFusion Plex product library, PLEX550, and the Application Integrator product library, APPINT on the AS/400. If you are only using the Schema/ODBC or Schema/Biz modules you can ignore this section.

To restore the AllFusion Plex objects from the product CD to the AS/400:

1. Transfer the save files from the CD to your AS/400. Follow the steps in Transferring the Product Libraries from CD to AS/400 section in the chapter “AS/400 Components.”

   The appint.savf file is installed in the Application Integrator directory. The plex.savf is available in the AS400 Libraries folder on the product CD.

2. Follow the steps in Restoring the Product Libraries from Save Files (*SAVF) in the chapter “AS/400 Components” if you are using the product CD.
3. To restore the APPINT library enter:

```
RSTLIB SAVLIB(APPINT) DEV(device) ENDOPT(*REWIND)
FRCOBJCVN(*YES)
```

## Running Multiple Versions of AllFusion Plex

If you are running multiple versions of AllFusion Plex, edit the APPINT/APPINT job description to use the correct names for the PLEX and APPINT libraries.

Chapter 12

# Frequently Asked Questions

This section contains some frequently asked questions and their answers.

## Questions and Answers

**Q:** When should I back up my group model?

**A:** You should perform regular backups of your group models on a daily basis. A group model is the central design repository. It is extremely important. If you accidentally destroy a local model, all the work you have done since the last extract from the group model is lost. If the group model is accidentally destroyed, all the work for that group model is lost. It may be possible to recover all or some of the data from a local model (using the XML Import/Export feature) but this is not guaranteed in all cases. For more information, see Backups in the online help.

**Q:** How often should I update?

**A:** Computer Associates recommends that you update your changes to the group model at least once a week, but preferably more often. Frequent updating in a workgroup environment makes your changes available to other developers. More importantly, after changes are updated to the group model, they become part of the central repository and are more secure. Note that extended periods between updates affect the time it takes to do an update; the more work that AllFusion Plex has to do to get through the differences between your local model and the group model, the longer it takes.

***Important!*** *Do not wait several weeks between updates.*

**Q:** Why does my function, field, table, or view have two implementation names?

**A:** You and someone else updated the group model after having generated and built functions and panels or tables or views. Your implementation and file name triples and objects—and those of the other person—were added to the group model without one set replacing the other. If this happens, you must delete the duplicate names.

Note that a group model conflict occurred when the second person tried to update the group model. You can choose not to update at this point. To avoid such conflicts in advance, ensure that you frequently update changes, especially after making changes that impact other developers in your workgroup. For more information, see Avoiding and Resolving Conflicts in the online help.

**Q:** I tried to generate an object but I got an error stating that I must specify an implementation language. What went wrong?

**A:** If you inherited the object from a pattern library (such as OBASE) make sure that the configuration of your local model is correct. To do this, choose Configuration from the File menu and, in the Model/Library box, select each model in turn. For each one, make sure that the Variant option is not set to Base. Instead, select the appropriate variant for your current work.

**Q:** What is the difference between levels and versions in group models?

**A:** A version is a collection of levels in a specified sequence. Assigning work to various levels means that you can keep track of general fixes and customer specific fixes simultaneously. For example, in the application FunkyApp, after Release 1.0 two fixes were made for customer A and customer B, followed by a general release of fixes at Release 1.1. This resulted in four levels: Release 1.0, Customer A fixes, Customer B fixes, and Release 1.1. The following table shows the version and level combinations possible. For more information, see Working with Versions and Levels in the online help.

| Version | Levels Include | Description |
|---|---|---|
| Release 1.0 | Release 1.0 | The first generic release, of FunkyApp 1.0 for all customers |
| Customer A 1.0 | Release 1.0<br>Customer A fixes | A FunkyApp 1.0 release, specific for customer A |
| Customer B 1.0 | Release 1.0<br>Customer B fixes | A FunkyApp 1.0 release, specific for customer B |
| Release 1.1 | Release 1.0<br>Release 1.1 | Generic FunkyApp 1.1 release, for all customers except A and B |

| Version | Levels Include | Description |
|---|---|---|
| Customer A 1.1 | Release 1.0<br>Customer A fixes<br>Release 1.1 | A FunkyApp 1.1 release, specific for customer A |
| Customer B 1.1 | Release 1.0<br>Customer B fixes<br>Release 1.1 | A FunkyApp 1.1, release specific for customer B |

**Q:** I cannot use constants or literal values in action diagrams. Why?

**A:** It might be faster if you could enter literals anywhere in the code, as you can in most programming languages, so why does AllFusion Plex not permit this? The reason is maintainability. Too many programs in too many languages have literals that are coded into the programs with no explanation as to the significance of the letter or numeric value. As an enterprise development tool, AllFusion Plex was designed to enable applications to be easily maintained by someone other than the original developer. Hence, you are forced to label all literals all the time to make your programs easier to understand.

**Q:** What are all these +++ meta-operations in library functions?

**A:** Meta code is used in pattern libraries to interrogate the triples in the model and provide the expected functionality. For example, a field with a triple MyField default MyDefault does not provide any default values without supporting meta code in the action diagram to set that default value. To learn more about meta code, Meta-operations, click the Locate button to see related topics in the Contents tab in the online help. Work on customizing supplied library code by copying and altering the code of others.

**Q:** When should I abstract my code into a pattern?

**A:** Sometimes a pattern is obvious, but the extra effort to make a great, inheritable, customizable pattern is not worth the effort because you may only implement it twice. On the other hand, something that obviously should be a pattern may be difficult to abstract. Often, the answer is that after you implement it once—certainly when learning about abstracting something for the first time—it is then clearer and simpler to work out whether you should abstract it and how you can turn it into a pattern. To know when to abstract becomes easier with experience.

**Q:** I cannot see my field names in the action diagram debugger. Why?

**A:** In the C++ build options, select the Force Build Of Selected Objects; then rebuild your functions. If this does not work, delete the entire Obj subdirectory, and then rebuild. For more information, see Generating and Building for Action Diagram Debug in the online help.

**Q:** How can I simplify parameter mapping?

**A:** An important technique is the management of field domains. Field domains control how AllFusion Plex automatically maps parameters. When you perform mapping manually, field domains also restrict the list of available fields, which makes it easier to find the one you want. For more information, see Field Domains in the online help.

Another technique is to use default mapping in the Parameter Mapping dialog. Default mapping lets the Action Diagrammer automatically make its best guess when mapping parameters between function calls, saving you from mapping each parameter individually. For more information, see Setting up Default Mapping in the online help.

**Q:** What is the difference between pre, post, and edit points? Which should I use?

**A:** Edit points are left over from previous versions of AllFusion Plex. Collection points (the collective name for pre and post points) are an improved place to put your code. The difference is that edit points overwrite each other at the different levels down the inheritance tree (you have to be extremely careful with the nesting), whereas collection points are cooperative and just collect code together in sequence. You can always add code to the bottom of a collection point in any action diagram, regardless of whether it has been used before. Edit points are restricted because if they have been used, you can no longer use the edit point further down the inheritance tree. Therefore, we recommend that you do not use edit points.

So which should be used: post or pre points? Sometimes it is obvious. You want to put code before or after another piece of code in the edit, pre, or post point so you know where to put it. What if it does not matter if the code goes in the pre or post point? Then the code always goes into the post point. If someone inherits from your function and wants to add code, if your code is in a post point they can add code before it (in the pre point) or after it (remember that you can always add code to the end of a collection point, regardless if it has been used higher in the inheritance hierarchy). If you add your code habitually in the pre point, people can only add code after (never before) your code if they are further down the inheritance hierarchy. In conclusion, always use post points unless there is a reason not to. For more information, see Edit Points and Collection Points in the online help.

**Q:** Which post point do I use?

**A:** This is perhaps the most difficult issue within AllFusion Plex for the novice. Computer Associates recommends that you:

1. Make an educated guess.
2. Expand the code before and after the post point.
3. Consider whether this is the correct post point.
4. Test it and see.
5. If it does not work, try Step 3 again; then try Steps 1 to 5 again.

Knowing which post point to use is the key to writing AllFusion Plex functions quickly. Look at other functions and determine why the local modifications were added at the particular points that the developer chose. Remember that the class and pattern libraries are based on the use of edit and collection points, so they can be examined for clues. Sometimes it helps to open action diagrams from the libraries, for example examining UISTYLE/EditDetailGrid shows you how to integrate UISTYLE/Detail and UISTYLE/Grid, and how to use UISTYLE/UIBasicShell.

Examining other pieces of code, particularly the libraries, helps to increase your total knowledge of which post point to use and when to use it. This increases your ability as an AllFusion Plex developer.

**Q:** Why do I have to keep on dragging my line to the edit box at the top of the window?

**A:** You do not have to do this. You can press the Insert key on the keyboard to enable inline editing. Note that for multiline comments, you must press Ctrl+Enter to enter the line into the action diagram, but for a normal single line of code, just press Enter when you have finished editing.

**Q:** How do I delete a field or a variable from a function?

**A:** You can do this in the Model Editor by deleting triples. You cannot just press the Delete key on the Variable Palette. Go to the top line of the function or the Variable Palette (they both show the function name) and press F12. This calls a Model Editor focused on the function. Any fields and variables that may be deleted are listed in the triples. Note that inherited fields and variables are not listed. To delete an inherited field you must delete the triples from the function that specified them higher in the inheritance hierarchy. You cannot delete the fields and variables of library functions. Remember to press F5 (Refresh) to redisplay the variable palette to show that the fields and variables have been deleted.

**Q:** How can I document and add comments to my code?

**A:** Use the Comment and Seq constructs frequently. Take AllFusion of the AllFusion Plex long object names to produce meaningful names. Carefully consider the naming of subroutines and edit points, and use block narratives where appropriate.

**Q:** How can I indicate that a field is supposed to be a dual (call by reference) parameter?

**A:** Change the triple in the Model Editor from MyFunction input MyField to MyFunction dual MyField. You cannot do this from the Action Diagrammer.

**Q:** I changed the size of a Windows panel in the Panel Designer but at run time the size did not change. Why?

**A:** The Save Placement property of the panel is probably set to Yes and the previous size is probably still saved in the application .ini file. To see the change, delete the entry in the .ini file (or rebuild it using the Create Exe command) or set Save Placement to No.

**Q:** My button on the panel does not do anything, even though the code is there. Why?

**A:** The physical event in the panel has most likely not been mapped to the AllFusion Plex logical event. A *physical panel event* is the result of somebody clicking, dragging, pressing, double-clicking, or selecting something on the panel. For example, pressing a button or selecting a menu item or entering text are all regarded as physical events, because they occur visibly on your computer screen.

The other events in AllFusion Plex are *logical events*. Logical events only happen within AllFusion Plex. They are not visible to you. Instead AllFusion Plex lets you tie a physical event (clicking) to a logical event. The action diagram, and in particular the events handler, can respond only to logical events.

If you right-click the button (when it is selected) in the Panel Designer and choose Event Mappings, you can see the Event Mappings dialog. Click Pressed in the list box on the left (Physical event) and check that the list box on the right (Logical event) has the corresponding event selected in blue (for example, MyButtonPressed). If it is not highlighted, select it before clicking OK to close the dialog. Switch back to the action diagram—pressing F11 switches you quickly between a function and its panel—and press F5 to refresh. Find the event code (an Event: Event MyButtonPressed construct in the Events Handler construct), which should now be executed when you press the button.

If you want to check which physical events are mapped to which logical events, look in the Events folder in the Panel Palette in the Panel Designer. If a logical event has one or more physical events assigned to it, it can be expanded and the physical event or events are displayed.

**Q:** Why does the MDI parent not respond to common events triggered in the MDI children?

**A:** The Menu ID properties of the menu items are most likely not set. Open the MenuShell panel of MyMDI in the Panel Designer, and expand the Menu bar folder in the Panel Palette. Find the menu items that are intended to be duplicated across all of the children but responded to by the parent, and ensure that they have a MenuId property of a number. Do not use the numbers 1 to 500, 54016, or 54107.

**Q:** I have Menu IDs, but my MDI child panel does not trigger MDI panel events. Why?

**A:** All MDI child panels and the MDI parent panel inherit from the common MenuShell panel, so do not expect the child functions to pick up the Menu ID property numbers you have set until you recompile them. When you recompile the children, the individual panels will know the Menu ID numbers and be able to pass them to the parent. Remember to recompile the MDI parent because it needs to know the Menu ID numbers.

**Q:** How do I delete a menu item?

**A:** Press the Delete key. There is no selection to delete a menu item from the right button pop-up context menu.

**Q:** I cannot see the new values I have entered for my status fields. Why?

**A:** If you add values to a field in the Model Editor and then go into a panel, these new values are not shown on the field. In the Panel Designer, just select the field and choose Refresh Values from the pop-up menu. The values are refreshed to match the triples in the Model Editor.

**Q:** Is there a way to see the full text of a message in the Message Log?

**A:** Yes, right-click the message.

**Q:** I cannot see the changes I have made reflected in other windows. Why?

**A:** Click the Refresh button (F5) to see your changes. In some cases, you may need to close both a panel and its related action diagram (answering Yes when prompted to save changes), and then reopen them to see changes.

**Q:** I added a triple in the Model Editor but now I cannot find it. What happened?

**A:** There are a number of factors that determine which particular triples get displayed in the Model Editor. Make sure that you check each one. For more information, see Understanding the Model Editor Display in the online help. In addition note the following:

- The Model Editor does not display inherited (implicit) triples.
- Triples that you add to an inherited object may not be visible in an unfocused Model Editor window because there is no triple referencing the inherited object. In such cases, focus the Model Editor by dragging and dropping the inherited object from the Object Browser.

**Q:** Why does AllFusion Plex permit me to enter incomplete or illogical data?

**A:** While AllFusion Plex enforces correct syntax, it does not enforce consistency or completeness until you generate. Thus, the model could not contain the triple "Credit limit length Customer," but it could contain both "Credit limit length 6" and "Credit limit length 7." AllFusion Plex does this for the following reasons:

- Sometimes during design, not all of the properties of an object are known. Under these circumstances, a tool that does not let you record anything until everything is known can hurt rather than help.

- In a workgroup environment, it is sometimes required that several different values of a property be recorded simultaneously. So, two team members may disagree on the length of the credit limit field. AllFusion Plex enables both lengths to be recorded and the disagreement to be resolved later.
- In a workgroup environment, it is inefficient or simply impossible to enforce rules interactively. Not all the necessary triples may be present in your local model at a given time.

**Q:** There is no list of target object types available in the Model Editor. Why?

**A:** The target object type on the Entry Bar is not capable of input because its value is always determined by the verb that you select.