AllFusion® Plex

Release Summary

r6

This documentation (the “Documentation”) and related computer software program (the “Software”) (hereinafter collectively referred to as the “Product”) is for the end user’s informational purposes only and is subject to change or withdrawal by CA at any time.

This Product may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Product is confidential and proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of the Documentation for their own internal use, and may make one copy of the Software as reasonably required for back-up and disaster recovery purposes, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the provisions of the license for the Software are permitted to have access to such copies.

The right to print copies of the Documentation and to make a copy of the Software is limited to the period during which the license for the Product remains in full force and effect. Should the license terminate for any reason, it shall be the user’s responsibility to certify in writing to CA that all copies and partial copies of the Product have been returned to CA or destroyed.

EXCEPT AS OTHERWISE STATED IN THE APPLICABLE LICENSE AGREEMENT, TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS PRODUCT “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS PRODUCT, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

The use of this Product and any product referenced in the Documentation is governed by the end user’s applicable license agreement.

The manufacturer of this Product is CA.

This Product is provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7013(c)(1)(ii), as applicable, or their successors.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Copyright © 2007 CA. All rights reserved.

Contents

Appendix A: Upgrading from Earlier Releases

Chapter 1: New Features

This chapter documents the new features for AllFusion® Plex r6.0.

New Features in AllFusion Plex r6.0

The new features in AllFusion Plex r6.0 include:

- C# Server generation and .NET runtime
- AllFusion Plex Application Management Console
- ANT and MSBUILD build environment
- Code Library packaging and deployment

C# Server Generation and .NET Runtime

This release of AllFusion Plex includes a new C# generator that enables AllFusion Plex server applications to be generated in C# and deployed on the Microsoft .NET Framework.

This generator can be regarded as a logical replacement for the WinNTC (C++) generator. In common with the Java generator, CA plans that future releases of AllFusion Plex will not typically require C# applications to be rebuilt as part of the release upgrade process. This is a major advantage when compared with AllFusion Plex C++ development.

For an introduction to AllFusion Plex C# support, see the sample model and associated documentation in the folder "\Samples\Dot NET Support and Code Libraries". For more information, see the .NET Platform Guide (Obnet.chm) in the AllFusion Plex online help.

Pattern and Class Library Support

To generate a function in C#, you can simply set its FNC language SYS triple to C#. Alternatively you can utilize new variants in the STORAGE, DATE, and OBASE libraries to automatically set the language across your model. Functions with panels or reports cannot be generated in C#. However Java and WinC clients can connect to C# server functions.

The new CSAPI pattern library encapsulates commonly required C# source code objects.

AllFusion Plex .NET Runtime

The AllFusion Plex .NET Runtime is a .NET Assembly (Plex.Obrun.dll) that must be deployed with C# applications generated by AllFusion Plex. The Runtime is a framework that encapsulates common services required by these applications. The AllFusion Plex .NET Runtime provides the following functionality:

- Implementation of each generated action diagram statement
- Remote call infrastructure enabling remote calls to Java functions, RPG functions or C# functions on other servers
- Dynamic application partitioning of calls between servers of all types
- Type mapping for cross-platform calls
- Special field value support (null and empty values)
- Support for object fields (FLD type SYS = Object) for easy integration with third-party classes.
- Data access via OLE DB including multiple database connections

At the same time, the existence of the AllFusion Plex .NET Runtime does not prevent generated applications making direct calls into the Microsoft .NET Framework via action diagram API Call statements.

This architecture reduces the size of generated applications by avoiding redundant code generation. In addition, it enables CA to make fixes and enhancements without requiring customers to regenerate their code.

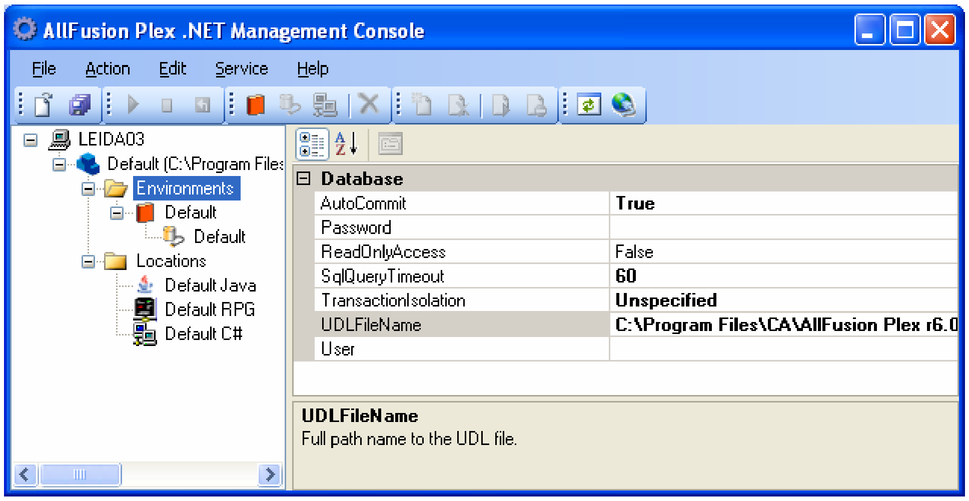
XML and .NET Integration Services

The AllFusion Plex .NET runtime provides a rich set of APIs and utilities to enable the integration of C# functions generated by AllFusion Plex with hand-coded .NET code. AllFusion Plex C# functions expose their parameter interfaces as .NET DataSets. This provides a programmer-friendly way for .NET developers to call C# functions generated by AllFusion Plex. No equivalent of an AllFusion Plex COM Connector is required.

A utility called *var2xml.exe* provides a way to automate the creation of XML files that document these parameter interfaces.

AllFusion Plex Application Management Console

The AllFusion Plex Application Management Console is a Windows application that enables the management of AllFusion Plex C# server applications and the associated .NET Runtime.



Database	
AutoCommit	True
Password	
ReadOnlyAccess	False
SqlQueryTimeout	60
TransactionIsolation	Unspecified
UDLFileName	C:\Program Files\CA\AllFusion Plex r6.0
User	

UDLFileName
Full path name to the UDL file.

The AllFusion Plex .NET Runtime can be run either as a command-line (or *console*) application or as a Windows Service.

Application Management Console—Basic Edition

When run as a Windows Service, the Application Management Console provides additional functionality such as remote server administration and application management data.

This release of AllFusion Plex includes a Basic Edition of the Windows Service. The Basic Edition is restricted to 10 concurrent users. This restriction does not apply when the AllFusion Plex .NET Runtime is run as a console application.

Note: In the future, CA plans to introduce a separately-licensed version of the Windows Service that will not have the 10 user limit.

The additional functionality provided by the Windows Service as compared to the console version includes the following:

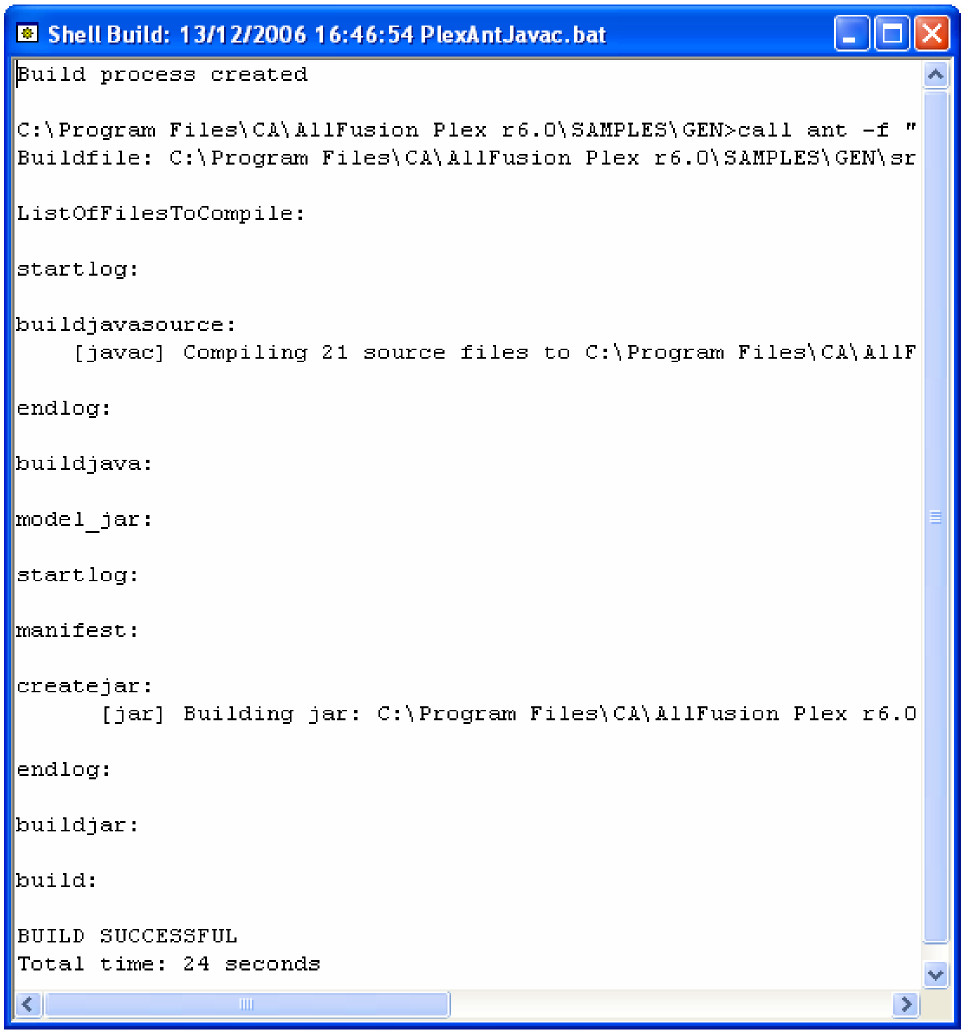
- Allows you to use standard Windows security for starting and running the AllFusion Plex .NET Runtime application.
- Configure the service to run multiple AllFusion Plex .NET Applications. Each application runs under its own *listener* instance. You can scale your application by easily adding more listener instances as the load on your server application increases. You can also have different applications or modules configured to run on different listener instances on the same system.
- Remotely configure and administer Plex .NET Runtime instances installed on any server inside your Windows network. Connect to any machine that has the Plex .NET Runtime Service installed on it, not just the local machine.
- Obtain information on the status of your deployed and running AllFusion Plex .NET applications anywhere in the network. Obtain information on the clients connected, the type of server requests they are making, the database activity performed, and other n-tier connections they might have made.

ANT and MSBuild Build Environment

The AllFusion Plex Java and C# generators utilize industry-standard build tools, Apache ANT for Java and MSBuild for C#. Apache ANT replaces NMAKE as was used in previous release of AllFusion Plex. MSBuild is a new build tool included in the Microsoft .NET Framework 2.0. Both Apache ANT and the Microsoft .NET Framework 2.0 are installed automatically with AllFusion Plex.

AllFusion Plex now generates Java (and C#) source files into directories reflecting their package structure defined in the model.

At build time, compiler messages are output to a new Shell Build window within the AllFusion Plex development environment. Multiple Shell Build windows can be kept open at one time, each identified with a timestamp in the title bar.



```
Build process created

C:\Program Files\CA\AllFusion Plex r6.0\SAMPLES\GEN>call ant -f "
Buildfile: C:\Program Files\CA\AllFusion Plex r6.0\SAMPLES\GEN\sr

ListOfFilesToCompile:

startlog:

buildjavasource:
    [javac] Compiling 21 source files to C:\Program Files\CA\AllF

endlog:

buildjava:

model_jar:

startlog:

manifest:

createjar:
      [jar] Building jar: C:\Program Files\CA\AllFusion Plex r6.0

endlog:

buildjar:

build:

BUILD SUCCESSFUL
Total time: 24 seconds
```

The new build environment has the following advantages as compared with previous releases:

- Vastly improved compile times for Java code.
- Automatic timestamp checking avoids unnecessary re-compilation of already compiled objects.
- Ability to customize and extend the build environment by editing the XML-based build scripts.
- Increased compatibility with 3rd party tools such as Eclipse and Visual Studio.

Default Code Libraries

For convenient unit testing, the build process automatically adds all Java or C# source files in the generation directory into a default code library (JAR file or .NET assembly DLL). The default code library is named after the local model but you can change its name with a setting in the Java or C# Build Options.

CA recommends using the new Code Library object type to model and implement the JAR files or .NET assembly DLLs that you actually deploy. For more information, see the next section.

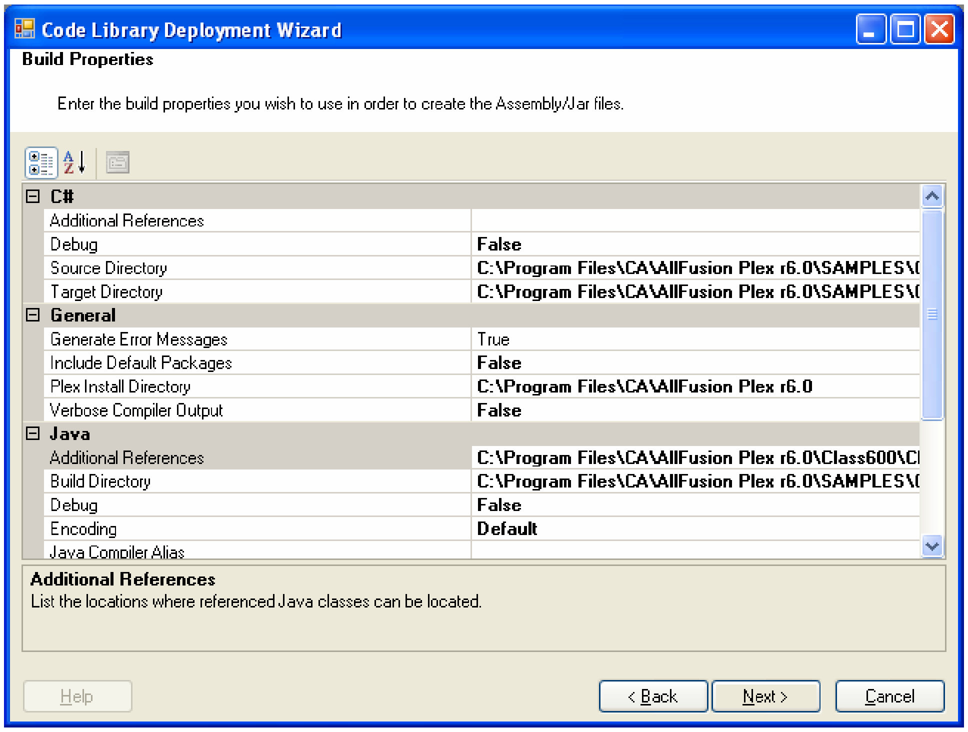
Code Library Application Packaging and Deployment

Both Java and C# organize classes into packages (called *namespaces* in .NET). Packages are typically organized into hierarchies with packages nested one inside another.

For deployment purposes, these packages are bundled into JAR files in the case of Java or assembly DLLs in the case of C#. Typically, one JAR file or .NET Assembly contains many classes with each class roughly corresponding to an AllFusion Plex function. Note that this approach is very different from AllFusion Plex C++ (WinC and WinNTC) deployment where each AllFusion Plex function produces its own DLL.

One notable difference between Java and C# is that Java class files can be executed without being part of a JAR file whereas .NET always packages classes into DLLs. The .NET Framework has no equivalent of a standalone Java .class file.

AllFusion Plex uses the term *Code Library* to represent these deployment units. With the new Code Library object type in AllFusion Plex r6, you can use triples in the Model Editor to specify which packages are to be included in the code library and to provide information for the associated manifest such as copyright statements and version numbers. The new Code Library Wizard on the Tools menu can then be used to build the JAR or DLL files themselves.



Build Properties

Enter the build properties you wish to use in order to create the Assembly/Jar files.

C#	
Additional References	
Debug	False
Source Directory	C:\Program Files\CA\AllFusion Plex r6.0\SAMPLES\
Target Directory	C:\Program Files\CA\AllFusion Plex r6.0\SAMPLES\
General	
Generate Error Messages	True
Include Default Packages	False
Plex Install Directory	C:\Program Files\CA\AllFusion Plex r6.0
Verbose Compiler Output	False
Java	
Additional References	C:\Program Files\CA\AllFusion Plex r6.0\Class600\C
Build Directory	C:\Program Files\CA\AllFusion Plex r6.0\SAMPLES\
Debug	False
Encoding	Default
Java Compiler Alias	

Additional References
List the locations where referenced Java classes can be located.

Chapter 2: Changes to Existing Features

This chapter documents changes made to existing features and also includes a list of features removed from AllFusion Plex r6.0. In addition, see the online help for a list of fixes included in AllFusion Plex r6.0.

The changes for the existing features can be classified under the following categories:

- Changes in System i support
- Changes in Java support
- Changes in C++ (WinC and WinNTC) support
- Updated Generate and Build Options dialog
- Other changes and enhancements
- Features removed in AllFusion Plex r6.0

Changes in System i Support

The following section discusses the enhancements to the AllFusion Plex System i support.

Rebranding

The name *System i* is now used in the AllFusion Plex product and documentation to reflect IBM's branding of the computer system formerly known as iSeries and AS/400. Similarly, the OS/400 operating system is now referred to as i5/OS.

Long Password Support for System i Servers

The AllFusion Plex System i dispatchers now support the use of long and mixed-case passwords up to 128 characters long as determined by the QPWDLVL system value.

Java, C++, and C# functions can all connect to AllFusion Plex System i servers using long passwords.

Previous releases of AllFusion Plex only supported passwords up to 10 characters in length.

New Password Management Sample Model

The AllFusion Plex run time provides a facility for the end-users to reset their own expired passwords if the System i Dispatcher (YOBSYTCPDP or YOBSYTCP) is running under a user profile with *SECADM authority.

Some users prefer not to run the dispatcher with *SECADM authority. In this scenario, you need to provide your own mechanisms for managing passwords. A new sample model shipped with the product illustrates the technique of changing passwords using IBM iSeries Access APIs (for C++). The sample model acts as an example of how you can build this functionality into your own applications, provided you have the IBM iSeries Access licenses.

Changes in Java support

The major new features for Java support in AllFusion Plex are ANT build support and JAR file modeling and generation through the new Code Library object. For more information, see the chapter “New Features.” This section summarizes other notable improvements to the AllFusion Plex Java support.

AllFusion Plex Java Runtime JDK Compatibility

The AllFusion Plex Java Runtime (Obrun.jar) is now compiled for compatibility with JDK 1.4 instead of JDK 1.2 as at the previous release.

Exec SQL Improvements

Exec SQL action diagram statements are now implemented as reusable prepared statements. This improves runtime performance by enabling the database to efficiently cache and reuse database queries.

FLD type SYS = Object Support

Java-generated functions can now use fields with FLD type SYS = Object (object fields). This feature is intended for users with Java programming knowledge and enables easier integration with third party Java classes. It can be regarded as comparable with server-side COM Component Import support for C++. It is more flexible than COM Import in that Java programmers can integrate any Java class usage into an AllFusion Plex model (not only formal components) without the overhead of importing the full class definition into AllFusion Plex.

This feature is also supported for C#. For more information, see Chapter 6 of the online User Guide.

Removal of Dependencies on .Obin and .Obout Classes

An AllFusion Plex Java function no longer has a dependency on the .Obin and .Obout classes of the functions that it calls. This enables a number of efficiency and performance improvements at generation, build and runtime. For example, when generating a Java function, AllFusion Plex no longer generates the .Obin and .Obout classes for any called Java functions.

New Java Database Connection APIs

The following source code objects have been added to the JAVAAPI library:

- JAVAPI/DisconnectDBConnection
- JAVAAPI/DisconnectALLDBConnection

In addition, the AllFusion Plex Java Runtime provides a getJDBCConnection method to return a handle to the native JDBC connection being used by the AllFusion Plex application.

Java Client: Input Validation of Edit Masks

Character fields in Java clients now support input validation of edit masks. To control this feature, edit controls in the Panel Designer have a new property called Edit Mask Valid For property. This property only applies to Java clients. The following are the valid values for the Edit Mask Valid For property:

- Input—Specifies that the edit mask is used to validate input.
- Output—Specifies that the edit mask is used for display only.
- Both—Specifies that the edit mask applies to both input and output.

The default value for this property can be controlled at model-level by the continuation triples on the FLD edited by LBL triple.

Java Client: Automatic Batch File Creation

When you run a Java client function from the Generate and Build window, AllFusion Plex automatically creates a batch (.bat) file that you can use to launch the function independent of AllFusion Plex. You can view or edit the contents of the batch file with the View Source command on the Generate and Build window.

Java Client: Infragistics JSuite Demo Controls

A single JAR file called pvAll.jar is now included with AllFusion Plex r6.0 instead of the complete Infragistic JSuite demo product. The pvAll.jar file contains all JSuite demo Java beans. You can download the full version or demo version from the Infragistics web site.

Changes in C++ (WinC and WinNTC) support

The following section provides the changes related to C++ support.

Packages Names in Generated Function Calls

To better support dynamic application partitioning between different platforms, AllFusion Plex now uses the fully qualified implementation name of a called function in the generated C++ code. The fully qualified implementation name includes the package to which the function belongs. For example, *MyPackage.AA1F" or "MyScopingPackage.MyPackage.AA1F*.

Previously, AllFusion Plex included the package only if the called function had a language of Java at generation time. If the called function had a language of RPGIV (for example), this would prevent the call being dynamically switched to Java or C# at run time (because Java and C# both require the package name to execute the call).

If no package definition is provided by triples in the model, then a default package name is used. For C++, RPG and Java, the default package name is specified in the Java Generation Options. For C#, it is specified in the C# Generation Options.

Typically, there will be no upgrade impact associated with this change unless you have previously used function-level deployment time partitioning. For more information, see the topic Deployment Time Partitioning in the online help's User Guide.

Visual Studio 2005 Support for C++

AllFusion Plex r6.0 requires Visual Studio 2005 SP1 for compiling C++ functions. The Standard, Professional, and Architect editions of Visual Studio 2005 SP1 are all compatible with AllFusion Plex.

Note: The process for deploying the MFC and Visual C++ runtime DLLs has changed. For more information, see the online help topic AllFusion Plex-Generated Files Required by Windows Clients. C++ applications created by AllFusion Plex r6.0 are unmanaged C++ applications that do not require the Microsoft .NET Framework.

Updated C++ Fixed Decimal Libraries

The C++ programming language does not provide native support for fixed precision data types (known as fixed decimal fields in AllFusion Plex). Instead, AllFusion Plex provides support for C++ fixed decimals through a third code library. For AllFusion Plex r6.0, CA has switched to a different third party library. As a result of this change an additional DLL (ob600decNumber.dll) must be deployed with AllFusion Plex C++ applications. No changes in functionality should result from this change.

NT/BackOffice Rebranding

The term *BackOffice* generator is no longer used in the AllFusion Plex product documentation because Microsoft has discontinued the BackOffice Logo program. The term *Windows C++ Server* generator is typically used to describe this platform instead.

If you are developing new applications for the Windows Server platform, CA recommends that you use the C# or Java generators.

Updated Generate and Build Options

Various changes have been made to the Generate and Build Options dialog reflecting the addition of new features and the removal of other features. The changes to the Generate and Build Options dialog are summarized as follows:

- Build (.BLD) files created at earlier releases are not fully compatible with AllFusion Plex r6.0. In general, CA recommends that you create new local models (and thus, new BLD files) as part of the release upgrade process. If you use an old BLD file, then you may need to review your settings to avoid problems. If an old BLD file is being used, the Build Directories section will display references to AS/400 instead of System i.
- To activate or deactivate a System i or WINNT build system, use the new check box on the System i Configuration or NT Configuration dialog box. This new check box is called *Make this the active build system.* If no remote WINNT system is active, (which is the default) then the Local System is used instead and WinNTC objects are built locally.
- Database Build options is the new name for ODBC Build options. This reflects that these options are used for all local builds of database objects including JDBC and .NET database objects.
- The Resource Directories setting in the Generate and Build Options now appears in the Generation Options section (instead of 32-bit C++ Builds section). This is because this option affects generation and applies to both Java and C++.

Path and Directory Validation

Paths and directories entered in the Generate and Build Options dialog are now validated, including entries with multiple paths separated by semi-colons (;). If a specified file or directory is not accessible, a warning message is displayed in the Message Log but the invalid value is left in place. Note the following:

- The validation checks only that the file or directory exists, not whether it is the correct value for the particular option concerned.
- Root directories such as C:\ or D:\ are not validated.

Other Changes and Enhancements

The following sections describe additional changes and enhancements. For a list of fixes see the online help (Plex.chm).

Parameter Mapping Dialog Improvements

The Default Mapping feature is now available for API Calls and Message statements instead of being restricted to function calls only.

The sizes of the Parameter Mapping dialog and the Select Field dialog have been increased to reduce the need to scroll when viewing long field names. As a result, the minimum recommended screen resolution for AllFusion Plex is now 1024 x 768.

New OBASE Variants

New or renamed variants for OBASE include the following:

- JAVA/System i RPG400
- JAVA/System i RPGIV
- Windows/.NET

OBASE Help Files Now in CHM Format

The help files for the OBASE family of class libraries are now provided in Microsoft HTML Help (CHM) format. This enables you to access these help files from the Contents, Search and Index tabs of the main AllFusion Plex help system.

New Java Tutorial in Getting Started

Chapter 3 of the *Getting Started* (Plex_Getting_Started_ENU.pdf) includes an introductory tutorial to the product. For AllFusion Plex r6.0, this tutorial is now based on the Java generator instead of the WinC (C++) generator.

Summary of Verb Table Changes

There are two new object types in AllFusion Plex r6.0:

- Code Library (CDL)

  For more information, see the chapter "New Features."

- Page (PGE)—Represents a web page. Web page generation is not supported for this release and the associated verbs are not documented in this Release Summary.

The following table summarizes the new verbs and system values.

Source	Verb	Target	Description of the Target
CDL	language	SYS (C#, Java)	Defines the type of the code library to produce.
CDL	NET type	SYS (Module, Assembly)	Defines the type of .NET application block that this Code Library produces.
CDL	file name	NME	Specifies the name of the Code Library object.
CDL	impl name	NME	Specifies the internal name of the Code Library to inject into the manifest.
CDL	includes	CDL	Defines other scoped Code Libraries to include in the source Code Library.
CDL	comprises	PKG	Defines the packages that are stored in the Code Library.
CDL	company name	LBL	Contains the company name to inject into the manifest data.
CDL	copyright information	LBL	Contains copyright information to inject into the manifest data.
CDL	trademark information	LBL	Contains trademark information to inject into the manifest data.
CDL	default alias	LBL	Contains the default alias name to inject into the manifest data.
CDL	product	LBL	Contains the product name to inject into the manifest data.
CDL	sign file	NME	Specifies the name of the sign file to associate with a Code Library implemented as an assembly.
CDL	assembly information	SRC	Contains additional information to inject into the manifest generated for the Code Library.

Source	Verb	Target	Description of the Target
CDL	major version	NBR	Specifies the first element of the formal Code Library version number.
CDL	minor version	NBR	Specifies the second element of the formal Code Library version number.
CDL	build number	NBR	Specifies the third element of the formal Code Library version number.
CDL	revision number	NBR	Specifies the fourth element of the formal Code Library version number.
FNC	Language	SYS (C#)	The C# system value is new in this release.
LST	contains	CDL	Enables Code library objects to be placed on Lists.
SBJ	code library	CDL	Enables Code library objects to be placed in Subject Areas.

Features Removed in AllFusion Plex r6.0

The following features have been removed in AllFusion Plex r6.0:

- Clear Local Java Build Summary menu item
- Remote Generation
- Remote Java Builds—Java builds are now carried out on the local system.
- Generate JAR Batch File menu item—This functionality is replaced by the Code Library object type.
- Generate JavaHelp Map File menu item
- Build Selected Only menu item
- Joe EJB Deployment tool—AllFusion Plex can still generate EJBs which can be deployed according to your J2EE application server's documentation.

Appendix A: Upgrading from Earlier Releases

Multiple releases of AllFusion Plex and its generated applications can be installed on the same machine. For example, AllFusion Plex r5.5 and AllFusion Plex r6.0 can be installed and run on the same machine.

Before upgrading, review the readme (Readme.html) for system requirements and late-breaking upgrade notes. Also, check the AllFusion Plex home page on the CA Technical Support for additional information or fixes that are made available after documenting this guide.

Contact Technical Support

For online technical assistance and a complete list of locations, primary service hours, and telephone numbers, contact Technical Support at http://ca.com/support.

Upgrading from AllFusion Plex r5.5

This section explains upgrade requirements related to upgrading from r5.5 (including r5.5 SP1).

Windows C++ Functions Upgrade Requirements

Windows functions (WinC and WinNTC) created with AllFusion Plex releases earlier than r6.0 must be recompiled to be compatible with the AllFusion Plex r6.0 C++ run-time system.

A single Windows application cannot contain DLLs created with different releases of AllFusion Plex. AllFusion Plex includes a run-time version check whenever a generated DLL calls another. A run-time error occurs if you attempt to make a call between DLLs created with different releases. In your development environment, you need to rebuild all of your application DLLs whenever you move to a new release of AllFusion Plex.

For more information about run-time applications, see Running Applications Created with Different Versions of AllFusion Plex in the online help.

For more information about the new features in AllFusion Plex r6.0, see the chapter New Features in this guide.

COM Import Upgrade Considerations

If you have used the COM Component Import feature, CA recommends that you reimport and regenerate the imported COM packages as part of the release upgrade process. This is due to fixes and improvements to the COM import and wrapper generation processes at this release. You may need to revise the wrapper attributes of the package before it compiles. Note the following:

- The COM Component Import wizard provides an option *Do not overwrite existing package.* However, clearing this option is not appropriate for upgrading previously imported COM packages.
- An alternative is to delete the COM package object before running COM Component Import. However, this will invalidate existing action diagram statements that reference the COM package.

To avoid these limitations, the following upgrade procedure is recommended:

1. Create a new group model and attach the COMPONENT library.
2. Extract a new local model and use the COM Component Wizard to import a new COM package for component you need to upgrade.
3. In Object Browser, right-click the COM package and select XML Import from the Tools menu. Export the package to a named XML file.
4. Open a local model containing the COM package that you need to upgrade.
5. From the Tools menu, select Import and then XML Import. Select the Clear option and then import the XML file you created earlier.

The above procedure will upgrade the COM package while preserving existing action diagram code.

Java Functions Upgrade Requirements

This section discusses the upgrade requirements for Java functions.

Upgrading from AllFusion Plex r5.5 SP1

You do not need to regenerate existing Java functions if you are upgrading from r5.5 SP1 (Build 5.5.93).

The AllFusion Plex r6.0 Java run time (obrun.jar) is backwards compatible with earlier releases. You can use the new runtime with functions created at r5.5 and previous releases. This enables you to take advantage of fixes and improvements in the new runtime without necessarily upgrading to the full AllFusion Plex r6.0 SP1 product.

Upgrading from AllFusion Plex r5.5

If you regenerate a Java function at AllFusion Plex r6.0, then you must also regenerate all other functions in the call graph of that Function. This is due to changes in the parameter formats in AllFusion Plex r5.5 SP1, and r6.0. CA recommends a full regeneration and recompilation of all Java functions when upgrading from AllFusion Plex r5.5 (Build 5.5.63) or previous releases. This reduces the number of generated classes and identifies any source code objects that need to be modified for compatibility with the new format of generated Java code.

Note: This Java regeneration and rebuild requirement is only required when upgrading from AllFusion Plex r5.5 and earlier. CA does not anticipate that future releases of the Java generator will have this requirement. No such requirement exists if you are upgrading from r5.5 SP1.

If you want to use the AllFusion Plex r6.0 Java run time (obrun.jar) with an AllFusion Plex r5.5 Java application, you need to have these settings in your ob600xxx.properties file:

```
Version=600

SPVersion=0
```

Java Source Code when upgrading from AllFusion Plex r5.5

The Java source code you entered into source code objects in your model may need to be modified. This is because of the changes in the structure of the Java classes created by the generator at AllFusion Plex r5.5 SP1. The source code objects supplied in CA's pattern libraries (such as the JAVAAPI library) are already modified. Re-extract from the shipped JAVAAPI library before regenerating and compiling with AllFusion Plex r6.0.

Problems in Java source code result in an unexpected type error at compile time. For example:

C:\GENJAVA\MyFunction_ObFnc.java:*line number*: unexpected type

```
required: variable

found   : value
```

To fix such problems use the assign method instead of the = operator.

Prior to r5.5 SP1, you could use the operator = to assign the return value of a method to an output parameter of your source code API. For example:

```
&(4:) = anyMethod(&(1:), &(2:), &(3:));
```

From r5.5 SP1 onwards, the code in the previous example needs to be modified to include the assign method instead of the = operator, as follows:

```
&(4:).assign(anyMethod(&(1:), &(2:), &(3:)));
```

Windows Application Server Environment Settings

The registry keys for the AllFusion Plex Windows Application Server environment settings have changed because of the rebranding from Advantage to AllFusion.

The AllFusion Plex r6.0 installation program automatically copies pre-existing environment settings to the new key values. However, if you do not see the copied settings in the AllFusion Plex Windows Application Environment Manager, execute the migration2.exe program (in the AppServer\Bin folder).

Note: No parameters are required to execute the migration program.

RPG Functions Upgrade Requirements

You do not need to regenerate or rebuild existing RPG functions when upgrading to AllFusion Plex r6.0.

Group and Local Model Upgrade

Both local and group models can be upgraded to AllFusion Plex 6.0. You do not need to update local models to the group model before you begin. However, it is good practice to update local models to the group model regularly. Therefore, CA recommends that you update the group model and make offline backups of all group and local models before you start the upgrade. Then create new local models after upgrading the group model to the new release.

If you save or log in to a local or group model using AllFusion Plex r6.0, you will not be able to access the model with a previous release. For this reason, it is necessary for all developers in your work group to upgrade to the new release of AllFusion Plex at the same time. Two developers cannot work on the same model simultaneously if they are using different releases of AllFusion Plex.

Build (.BLD) file compatibility

As discussed in the previous section, it is recommended that you create new local models, and therefore new build files as part of the upgrade process. If you use an old BLD file with AllFusion Plex r6.0, AllFusion Plex sends an error message (E-BLD-1777) each time you open the Generate and Build window. If you want to continue using the old BLD, review your Generate and Build Options to ensure they are compatible with AllFusion Plex 6.0.

Old build files will cause the Build Directories section of the Generate and Build Options to be displayed using the AS/400 brand name instead of System i.

To prevent the error message being displayed you can add the following entry to the BLD file:

```
[Options]

Release=600
```

Long File Names Not Truncated By Default

The Name Allocation option called Truncate Long File Names at Generation is no longer switched on by default. This means that the file names longer than 8 characters will no longer be truncated unless you explicitly select this option. This may have an upgrade impact if your model contains names that were truncated at previous releases.

Upgrading from AllFusion Plex r5.1

The following sections provides the upgrade requirements related to upgrading from r5.1

Inheritance Resolution Changes in AllFusion Plex r5.5

At r5.0 of AllFusion Plex (formerly known as Advantage Plex), the inheritance engine was changed to address limitations and bugs in inheritance behavior. Additional changes were also made to support the new Dependency Browser in r5.0. For some customers, these changes caused significant problems when the time came to upgrade to the new release. AllFusion Plex r5.1 included some fixes for these problems, but some significant problems remained.

AllFusion Plex r5.5 resolves the known bugs in the inheritance engine. No changes were made to the inheritance engine between r5.5 and r5.5 SP1. There are differences in inheritance engine behavior compared to earlier releases that will impact some customers. In the following sections three different cases are described.

Case 1: Multiple Inheritance Call Resolution

In some cases inherited function calls may be resolved differently in the action diagram with the result that a different call is created in the generated code. In a typical model, this only affects a very small percentage of functions (if any). These cases can be identified through testing of the generated application, comparisons of generated source, or comparisons of action diagram code. If you require further assistance in identifying such problems, contact CA Technical Support.

You can resolve these problems by adding the appropriate replacement triples to return the function to its required behavior. Ideally such triples can be entered in your *standards layer* model, thus minimizing the amount of changes required.

Background

AllFusion Plex r3.5 introduced support for multiple function inheritance by changing the behavior of the FNC is a FNC verb. As a result, even existing AllFusion Plex functions could inherit from more than one function without keying any extra triples. Another consequence of this was to allow two or more different inherited Calls triples to be Visible in the Object Properties corresponding to the same original action diagram call. Only one call is appropriate in any inheriting action diagram and AllFusion Plex determines which of the available functions should actually be called when it is loaded.

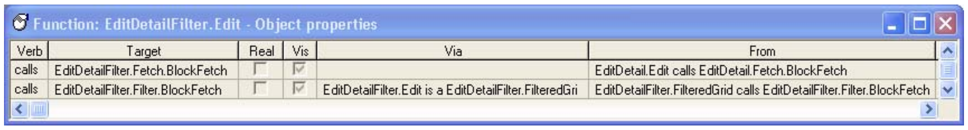
Prior to r5.0, AllFusion Plex did not explicitly handle such action diagram calls. As a result, the resolution of the call did not always follow the usual rules of the inheritance engine. At r5.0 and later a mechanism was put in place to handle such calls. This mechanism has been refined a number of times, most recently at r5.5.

Example 1: The Filter Pattern

A common example of multiple inherited action diagram call can be seen in instances of the Filter.FilteredGrid function. Consider these triples:

```
EditDetailFilter is a ENT STORAGE/RelationalTable
EditDetailFilter is a ENT FOUNDATI/EditDetail
EditDetailFilter is a ENT FOUNDATI/Filter
EditDetailFilter.Edit replaces FNC UIBASIC/Grid
            …by FNC EditDetailFilter.FilteredGrid
```

When you look at the calls for the EditDetailFilter.Edit you will see calls to two different BlockFetch functions:



Function: EditDetailFilter.Edit - Object properties

Verb	Target	Real	Vis	Via	From
calls	EditDetailFilter.Fetch.BlockFetch	☐	✓		EditDetail.Edit calls EditDetail.Fetch.BlockFetch
calls	EditDetailFilter.Filter.BlockFetch	☐	✓	EditDetailFilter.Edit is a EditDetailFilter.FilteredGri	EditDetailFilter.FilteredGrid calls EditDetailFilter.Filter.BlockFetch

Note that both calls are visible (the Vis column is checked). You can see this same behavior in r5.0 and r5.1. So which BlockFetch is actually called in the action diagram? This determination is made at action diagram load time. The result is sensitive to changes in the inheritance engine in recent AllFusion Plex releases. The intention is that the Filter.BlockFetch should be called. To get this result at r5.5, CA has entered an additional replacement triple on the FOUNDATION pattern library:

```
FOUNDATI/Filter.FilteredGrid replaces FNC UIBASIC/UIBasic.Grid.BlockFetch
              ….by FNC FOUNDATI/Filter.Filter.BlockFetch
```

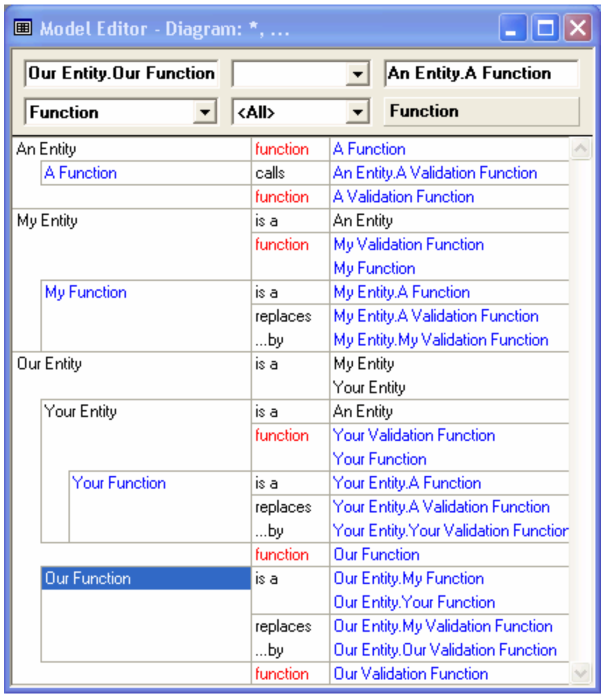
Previously, the BlockFetch function was not explicitly replaced. Instead the replacement was made only on the scoping view:

```
FOUNDATI/Filter.FilteredGrid replaces VW UIBASIC/UIBasic.Grid
                    …by VW FOUNDATI/Filter.Filter
```

With the revised algorithm used in AllFusion Plex r5.5, this was not sufficient to force the required function replacement to occur. By adding an additional, explicit replacement on the BlockFetch itself, CA has been able to retain the required behavior in r5.5. Similar actions may be necessary in your own functions that exhibit this behavior.

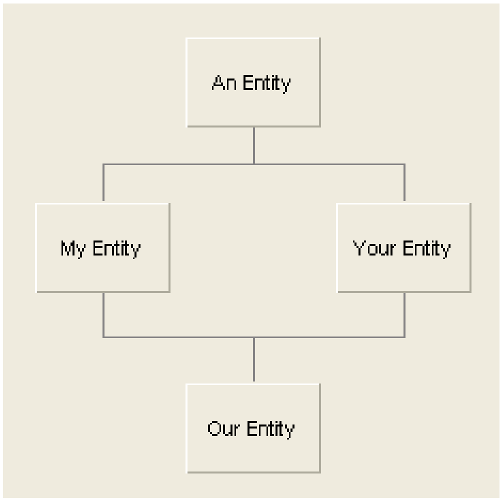
Example 2: Diamond Inheritance

Consider the following set of triples:



Our Entity.Our Function		An Entity.A Function
Function	<All>	Function
An Entity	function	A Function
A Function	calls	An Entity.A Validation Function
	function	A Validation Function
My Entity	is a	An Entity
	function	My Validation Function
		My Function
My Function	is a	My Entity.A Function
	replaces	My Entity.A Validation Function
	...by	My Entity.My Validation Function
Our Entity	is a	My Entity
		Your Entity
Your Entity	is a	An Entity
	function	Your Validation Function
		Your Function
Your Function	is a	Your Entity.A Function
	replaces	Your Entity.A Validation Function
	...by	Your Entity.Your Validation Functio
	function	Our Function
Our Function	is a	Our Entity.My Function
		Our Entity.Your Function
	replaces	Our Entity.My Validation Function
	...by	Our Entity.Our Validation Function
	function	Our Validation Function

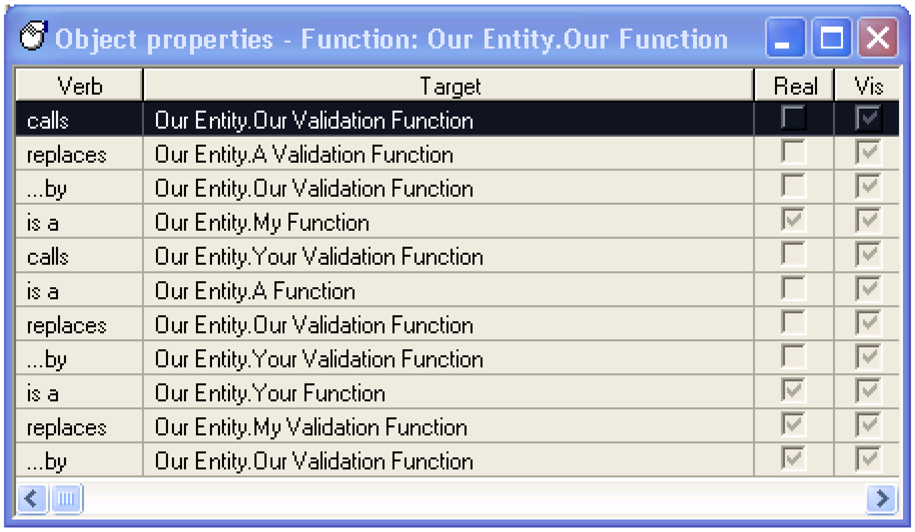
This type of scenario is known as *diamond inheritance* due to the shape of the resulting inheritance hierarchy as shown by the following diagram.



The inheritance hierarchy starts at *An Entity*, diverges, and then the two branches are brought back together at *Our Entity*. This situation is quite complex but the examples in real customer models are often even more complex with further branches and layers of inheritance involved.

If you view the Object Properties for the function Our Entity.Our Function then you can see the two calls triples, both visible. There are two important points to note:

- There is only one actual action diagram Call explicitly entered in this set of functions but two possible calls were resolved in Object Properties
- This scenario can be reproduced in any AllFusion Plex release from 3.5 onwards



Verb	Target	Real	Vis
calls	Our Entity.Our Validation Function		✓
replaces	Our Entity.A Validation Function		✓
...by	Our Entity.Our Validation Function		✓
is a	Our Entity.My Function	✓	✓
calls	Our Entity.Your Validation Function		✓
is a	Our Entity.A Function		✓
replaces	Our Entity.Our Validation Function		✓
...by	Our Entity.Your Validation Function		✓
is a	Our Entity.Your Function	✓	✓
replaces	Our Entity.My Validation Function	✓	✓
...by	Our Entity.Our Validation Function	✓	✓

The only way to determine which function is actually called is to examine the action diagram code. This example again shows the changes in the inheritance engine over recent releases: At r5.0, Our Entity.Your Validation Function is actually called, but in all other releases including r5.5 it is Our Entity.Our Validation.

Case 2: Changes to Sequence of Parameters

In some cases the sequence of inherited parameters on a generated function call may change compared to earlier releases. In many cases, such changes have no negative consequences since all functions concerned are regenerated in the course of upgrading to the new release. However, there are cases where such changes may be significant:

- If you are exploiting the backwards compatibility of existing RPG and Java functions, then this change may require that additional functions be regenerated
- If you have hand-coded programs that integrate with programs generated by AllFusion Plex, then those hand-coded programs may be modified to take account of the revised parameter interface

These cases can be identified by testing the generated application, comparisons of generated source, or comparisons of action diagram code. If you require further assistance in identifying such problems, contact CA Technical Support.

Case 3: Changes to the Sequence of Events, Subroutines and Collection Points

The sequence of inherited Event constructs and Subroutines can be altered compared to previous releases. Changes to the sequence of Subroutines have no negative consequences for the generated application. In general, this is also true for Event constructs. However, the sequence of Events may be significant if you referenced the same logical event on multiple Event constructs or used an unqualified Event construct. For example:

```
Events Handler
    Event
        //unqualified event triggered for every event
        Go Sub Generic Event Processing
    Event Delete
        Go Sub Delete Processing
    Event Delete
        Go Sub More Delete Processing
```

If the previous sequence of Event constructs is different, then the behavior of the generated application will change. In practice, this functionality is rarely exploited and it is even rarer for the inheritance engine changes to cause significant differences in the sequence. As in other cases, such problems can be identified through testing of the generated application, comparisons of generated source or comparisons of action diagram code.

In principle, similar considerations apply to the sequence of inherited code blocks in Collection Points (Pre and Post Points). Testing by CA has not revealed examples of such results but you should be aware that the possibility exists.

Upgrading from AllFusion Plex r5.0

In addition to the upgrade requirements documented in the earlier sections, review the following sections when you are upgrading from r5.0.

MFC Native Controls in Windows Clients

In r5.0 and later, AllFusion Plex supports two sets of GUI controls in Windows clients. By default, WinC applications use MFC controls for all types of controls, except the grid. The alternative set of GUI controls is called Winwidgets, which was the default before r5.0.

There are advantages and disadvantages associated with each set of controls. Winwidgets controls have been used in all releases of AllFusion Plex since 1.0. Consequently, they represent a mature technology that has been implemented successfully for many years by many AllFusion Plex customers. When upgrading from an earlier release of AllFusion Plex, it is often simpler to use the Winwidgets controls since this is likely to minimize any backwards compatibility problems.

MFC controls provide a range of advantages including:

- Compatibility with third-party testing tools
- Windows standard look and feel
- Access to the MFC API for low-level control

Backwards Compatibility Options

With some exceptions (see the following section), all the functionality previously available with Winwidgets controls should also be available with the MFC controls. However, to safeguard against unexpected upgrade issues, run-time options are available that enable you to revert to the Winwidgets controls.

Note: Due to known issues with numeric fields, single line edit controls for numeric fields currently default to Winwidgets instead of MFC controls. There is a separate INI file option available to enable the MFC control for numeric fields, if required.

A separate option is available for each type of control:

```
[NativeControls]
ListBox=1
SpinButton=1
SingleLineEdit=1
    SingleLineEditNumeric=0
MultiLineEdit=1
ComboBox=1
RadioButton=1
CheckBox=1
PushButton=1
Statics=1
```

Set the required option to 0 if you want to revert the control concerned to Winwidgets.

Note: Report any undocumented upgrade issues to CA Technical Support.

Upgrade Issues with MFC Controls

Known upgrade issues include:

- **Disabled text color.** When MFC controls are disabled, the text displays in the Windows standard color (typically, light gray). The Text Color property is ignored when controls are disabled. Note that for edit controls you can use the combination of properties

  `Mode=Read-only`

  and

  `Disabled=No`

  to define a read-only edit control that does support the Text Color property.

- **Combo box size.** Unlike other control types, the size of the edit control portion of a combo box cannot be changed directly; it is determined by the font size of the text within the control. You can change the size of the edit portion at design time but it is ignored at run time. Instead, changing the size of the edit portion at design time, changes the size of the drop-down list at run time.

- **Transparency**. MFC controls do not fully support transparency. This may change the appearance of panels in cases where controls overlap.

- **z-order**. The z-order of MFC controls is reversed compared to the default z-order of Winwidgets controls. This may impact some panel designs that rely upon overlapping controls.

Note: To allow the z-order of design time controls to more closely match the run time, set the Clip Control property to Yes for each control concerned. This is useful when working with overlapping controls such as a frame within a frame.

Calling Java Functions

As of AllFusion Plex r4.5, when calling Java functions from the command line or from hand-coded Java, the function name must be prefixed with its package (package.func).

Note that at AllFusion Plex r5.0 Service Pack 1 and later, there is a PackageList .properties file entry that can be used to provide a list of packages that are searched if no package is specified within the call. Typically, a function's package is generated directly into the code for each function call. However, this may not be the case in a dynamic partitioning scenario. For example, at generation time the target of a function call could be a Windows function (in which case no package will be included in the generated call). At run time, if the target function is switched to Java then the PackageList can be used to locate the function.

Upgrading from AllFusion Plex r4.5

This section explains upgrade requirements related to r4.5.

Inheritance and Property Resolution Changes

AllFusion Plex r5.0 and later includes enhancements to the inheritance engine that may change the properties of inherited objects in some circumstances.

Replacing the Target of an Inheritance Triple

Consider the following example:

A **is a** B

B **is a** C

A **replaces** C **by** D

When B **is a** C arrives on A, A no longer inherits from C, as B did, but it now inherits from D.

In previous releases, anything inherited from D, which is in contention with that inherited from B, defers to the version inherited from B. In r5.0, anything contentious inherited from D takes precedence over things from B which arose from its inheritance from C. In other words A **is a** D takes precedence over A **is a** B and B **is a** C. This result is now consistent with the general rules of inheritance, where *later* triples take precedence over *earlier* triples.

Triples for the Same Property That Have Been Entered at More Than One Level

It is possible to duplicate triples for an object by entering a triple, changing configuration to an earlier level and entering the same property. For example:

A **type** Character (Level 3)
A **type** Numeric (Level 1)

With the configuration set to the later level (Level 3 in the example) both triples would show in the Model Editor prior to r5.0 and are passed on to the inheritance engine. The last triple in sequence (as seen in the Model Editor) takes precedence. So in the previous example, A would be Numeric.

In AllFusion Plex r5.0, the *later hides earlier* principle is applied to the levels and only A **type** Character is seen in the Model Editor and passed on for inheritance.

Meta-Variables in RPG Internal Functions and OBASE/Set Current Date and Time

At r5.0, a fix was added so that internal RPG functions now have their own meta-variable space, and no longer share the meta-variable space of their calling functions. This change is in accordance with the published functionality regarding the scope of meta-variables (which states that a meta-variable's state only persists on calls to functions of type Meta) and is consistent with other generators.

This fix highlighted a place within the class libraries where meta-variable state information was expected to pass from an external function to an internal function. It was calling the OBASE/Set current date and time function in the OBASE variants AS400 5250 and Windows/AS400.

Note: OBASE/Server Set current date and time is a new function used with RPG400. It is recommended that you call this function instead of the OBASE/Set current date and time function with OBASE set in the Windows/AS400 variant and OBDATE set in the Windows client variant. This is exemplified in the call change in OBASE/Audited entity.Set audit fields.

To assist customers in tackling any upgrade issues associated with this fix, the following Plex.ini file option can be used in AllFusion Plex r5.1:

```
[RPG Generator]
Share Meta-Variables With Internal Functions=1
```

An entry of 1 reverts to pre-r5.0 behavior. If no entry is present, the default setting of 0 is used, providing the same behavior as r5.0.

Use of Single Quote Character in Java Messages and Source Code

The Java Generator at AllFusion Plex r4.5 and earlier required two single quote characters ('') to be used in messages and source code and required a single quote to be used at run time. For example, this was required when embedding parameters in the source code for use with the Exec SQL statement. This behavior was inconsistent with other generators.

This problem was corrected in AllFusion Plex r5.0. Now only a single quote character (') is required. However, any existing messages or source code that used the previous workaround need to be edited when upgrading to r5.0.

Use of Backslash Character in Values for Java

The Java Generator at AllFusion Plex r4.5 and earlier required that two backslash characters (\\) be used in values whenever you required a single backslash character to be used at run time. This behavior was inconsistent with other generators.

This problem was corrected in AllFusion Plex r5.0. Now only a single backslash character (\) is required. However, any existing values using the previous workaround will need to be edited when upgrading to r5.0.

Upgrading from AllFusion Plex r4.0

This section explains upgrade requirements related to r4.0.

Replacement and Scoped Objects

AllFusion Plex r4.5 introduced a change to the resolution of replacement triples attached to scoping objects to provide better control over replacement and performance improvements. This fix is not enabled by default. It requires a setting to be added to the Plex.ini file because it can cause significant backwards compatibility issues when it is enabled. For a full discussion, search the online help for the topic Replacement and scoped objects.

Upgrading from AllFusion Plex r3.5

In addition to the instructions in Upgrading from AllFusion Plex r4.0, review the following sections.

Change of Behavior with For Update Option

In r4.0, the behavior of the For Update option on Position and Fetch action diagram statements was changed to support pessimistic concurrency. It is possible that these changes may significantly change the behavior of ODBC-based applications (including NT BackOffice applications). For more information about backwards compatibility options, see the topic Row Locking in SQL Implementations in the online help.

Change in CONCAT Operator in C++ Code

The documented behavior of the CONCAT operator is that trailing blanks are removed from field values. In practice, it was possible to retain trailing blanks in C++ functions in cases where the values concerned were not displayed on panels. This inconsistency was corrected in AllFusion Plex r4.0 in that trailing blanks are now removed in all cases. The recommended technique for including blanks in concatenated strings is to use the Format Message statement with the blanks embedded in the message.

Avoiding Run-Time Level Checks When Accessing the System i Field Values File

The YOBDDL program resides within the AllFusion Plex library on the System i. When building the System i Field Values File within AllFusion Plex, a copy of the YOBDDL program is created in the System i Object Library. This copy is called YOBVALSV. A copy is not made if YOBVALSV already exists.

The YOBDDL program was modified for r4.0 of AllFusion Plex. For this reason, the old versions of YOBVALSV need to be deleted from each of your System i Object Libraries. Enter the following command on the System i:

```
DLTPGM  PGM(Object-Library/YOBVALSV)
```

The new version of YOBVALSV needs to be placed in to each of the System i Object Libraries. This can be done by either:

- Rebuilding the System i Field Values File
- Creating a duplicate object by entering the following command on the System i:

```
CRTDUPOBJ OBJ(YOBDDL) FROMLIB(PLEX)
OBJTYPE(*PGM)
  TOLIB(Object-Library) NEWOBJ(YOBVALSV)
```

Failure to perform these steps results in a level-check at run time when attempting to access the field values selection list.

Dynamic Application Partitioning APIs

If you have used the GetLocationInformation and SetLocationInformation APIs, note that the structure ObLocationInfo was changed for AllFusion Plex r4.0. If you have created source code objects that implement these APIs, you must edit the code to avoid compile errors.

The following three fields are removed:

```
ObLongFld m_fDataConv
ObLongFld m_fObTran
ObCharFld m_ObTranDLL
```

and replaced with the following single field:

```
ObLongFld m_iDataConv
```

See the Odap.mdl sample model for examples of the required source code.

Linker Options in Upgraded Local Models

AllFusion Plex r4.0 introduced a new feature called custom C++ build options. The linker option /OPT:NOWIN98 is used by default. Without this option the size of compiled DLLs is significantly increased. If your local model was created before r4.0 of AllFusion Plex, this option does not appear by default and should be added manually. For more information, see the online help topic Customizing C++ Builds.

Loading the Run-Time Property DLL

The source code required to load the run-time property DLL was changed in AllFusion Plex r4.0. If your application uses the SetProperty API, change the source code that loads the run-time property DLL, as follows:

```
#ifdef _DEBUG
  ObPropertyAPI::SetValue(Ob600Prpd.dll,
                           OB_ATOM_ATOM_INSTALL, 0, 0)
#else
  ObPropertyAPI::SetValue(Ob600Prp.dll,
                           OB_ATOM_ATOM_INSTALL, 0, 0)
#endif
```

User Data (in BSUPPORT) and Business Contacts (in BCONTACT) Patterns

The functions with panels were moved from the container Services to the container UI in AllFusion Plex r4.0. If you have previously used one of these patterns in AllFusion Plex r3.5, you must rescope some of these functions inherited from that pattern. After extracting from the new version of the library, you can see your existing panel functions in the Object Browser under Services as real (in yellow), and new functions with the same names under UI (not yet real, in gray). Rescope the existing panel functions by dragging them from the Services function and dropping them onto the UI function. (Note that the UI function must be made real before doing this.)

Upgrading from AllFusion Plex r3.1 and r3.0

In addition to the instructions in Upgrading from AllFusion Plex r4.0 and Upgrading from AllFusion Plex r3.5, note that AllFusion Plex r3.1 was the last release of AllFusion Plex that supported the creation of 16-bit Windows applications.

Upgrading from AllFusion Plex 2.51 and Earlier

You cannot upgrade AllFusion Plex 2.51 or earlier without first upgrading to AllFusion Plex r3.0. Follow the upgrade instructions published with AllFusion Plex r3.0.