CA Plex

Release Notes

r6.1

CA Product References

This document references the following CA products:

- CA Plex

Contact Technical Support

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 (http://www.ca.com/support).

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com (mailto:techpubs@ca.com).

If you would like to provide feedback about CA product documentation, please complete our short customer survey (http://tinyurl.com/6j6ugb), which is also available on the CA Support website.

Contents

Chapter 1: New and Updated Features

This chapter documents the new and updated features for CA Plex r6.1 which include the following:

- Model-based service development
- Group Model Update History
- IPv6 support
- C++ runtime backwards compatibility
- Plex API enhancements
- New Product Folder Structure on Windows Vista and Windows XP
- System i Enhancements
- Java and C# Runtime Scalability Improvements
- New JAR file for Plex EJB Proxy deployment—ObEJBProxyClient.jar
- Improved Support for Native Debugging in Java and C#
- .NET Database Query Result Handling
- SQL Native Client with the MARS Connection option
- .NET Management Console Log File Viewing
- Additional Minor Enhancements
- Features Removed

Model-Based Service Development

This feature strengthens CA Plex support for SOA development by providing model-based service development capabilities directly in the product. Services are represented as objects within the Plex model, using the component modeling approach already established for COM and EJB objects.

WCF service generation is supported with this release and a plug-in architecture enables developers to create their own service generators. For more information, see the sample model and related documentation in the \Samples\Dot NET WCF Support directory.

WCF Service Generation

Windows Communication Foundation (WCF) is a new communication subsystem within the Microsoft .NET Framework that unifies several different communication technologies such as web services, .NET remoting, message queuing, and so on.

The WCF service generation in CA Plex r6.1 enables you to present business logic as services based on WCF. This can include business logic developed in the Plex model and logic from third-party applications.

Service Wrappers and Cross-Platform Interoperability

The new WCF service generation is designed to support the convenient *wrappering* of existing applications as services. This includes Java, i5/OS, and .NET programs. Generally, this means that the target of a FNC implemented by FNC triple can be a Java or RPG function. In the case of RPG, the target function can correspond to an i5/OS program developed outside CA Plex, such as i5/OS programs or programs developed with CA 2E.

Code Library Plug-Ins

The Code Library wizard can now be extended through the use of plug-ins. A template for a Java-based web service generator is provided as a starting point for developers.

Group Model Update History

When updating a group model, you now have the option of specifying a comment describing the update. An option on the Exchange menu controls whether or not this dialog box is displayed.

A *template* comment may be specified in Plex.ini, which will then be loaded in the dialog box when it is first displayed.
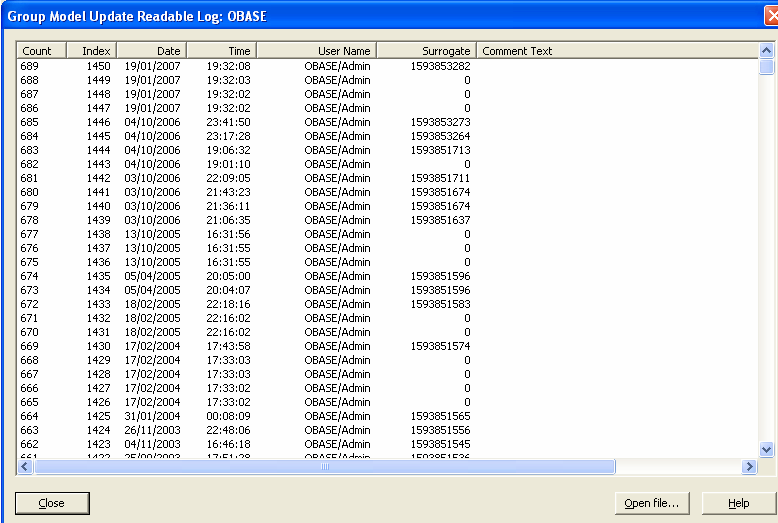
```
[Options]
Group model update comment template=
```
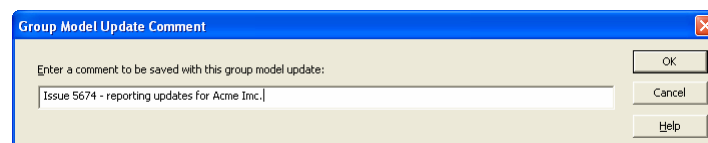
The comments are stored in a file in the group model folder and are displayed along with other information by the new View Update History menu item that is available from the Model menu. Previous releases of CA Plex already stored information about group model updates, but not in a human-readable format. This historical data is presented in the dialog box as well information on new updates that may include comments.

The history is stored in a tab-delimited text file that can be opened in a spreadsheet and saved or copied as required. The file (GroupModelUpdateReadableLog.txt) is re-created whenever you choose View Update History.

For more information, click Help on the above dialog boxes.

IPv6 Support

Distributed applications generated with CA Plex make extensive use of TCP-IP for communications between the various supported platforms such as Java, Windows C++, .NET, and System i.

Internet Protocol version 6 (IPv6) is a network layer standard used by electronic devices to exchange data across a packet-switched network. It follows IPv4 as the second version of the Internet Protocol to be formally adopted for general use.

IPv6 is intended to provide more addresses for networked devices, allowing, for example, each cell phone and mobile electronic device to have its own address. IPv4 supports 4.3×10^9 (4.3 billion) addresses, which is inadequate to give one to every living person. IPv6 supports 3.4×10^{38} addresses, or 5×10^{28} for each of the roughly 6.5 billion people alive today.

CA Plex 6.1 applications can operate in IPv4-only, IPv6-only, or dual protocol IPv4-IPv6 network environments. In the case of System i applications, the YOBSYTCPDP version of the dispatcher must be run to support IPv6.

C++ Runtime Backwards Compatibility

Typically when upgrading to a new release of CA Plex, you are required to rebuild existing CA Plex-generated C++ (WinC and WinNTC) applications. This is not the case when upgrading from release 6.0 to release 6.1.

To achieve this, the C++ runtime DLLs shipped with CA Plex r6.1 have the same file names as those shipped with 6.0 (ob600lc.dll, ob600nwi.dll, ob600ls.dll, and so on).

Note the following points:

- The 6.0 runtime DLLs are not forwards compatible. Runtime errors will occur if you attempt to run a 6.1 application DLL with a 6.0 runtime DLL.

- Updating the runtime DLLs from 6.0 to 6.1 has the potential to change the behavior of any existing application DLL. C++ runtime backwards compatibility removes the need to regenerate and rebuild application DLLs but it does not remove the need to retest your application when upgrading.

- You can identify which version of a runtime DLL you are using by looking at the Version tab on the Windows Properties dialog.

- C++ applications built with CA Plex r5.5 SP1 and earlier still need to be recompiled when upgrading to CA Plex r6.1.

- If you install 6.0 and 6.1 side-by-side on the same machine, then 6.0 WinC applications will use the 6.1 runtime DLLs by default. This is because the 6.1 Bin folder is added to the beginning of the path statement by the CA Plex installation program.

Limitations with Windows C++ Server Services

The CA Plex Windows C++ Server Services for r6.1 and r6.0 cannot be installed and run side-by-side.

System i Dispatcher Considerations

If you run an r6.0 created WinC application with the r6.1 WinC client runtime, the following error may occur:

```
CPF9801 Object in library *LIBL not found.
```

This error occurs when the r6.0 created client is connecting to the r6.0 version of the System i dispatcher.

If you use the 6.1 WinC client runtime then you must also use the r6.1 System i dispatcher. To avoid this CPF9801 error, perform one of the following environment setting modifications:

■ Modify the .INI file settings for the r6.0 created executable to use the Port of the r6.1 System i dispatcher. Make sure that the Job Description specified within the .INI file includes the r6.1 product library, for example PLEX610.

OR

■ On the System i, rename the dispatchers within the r6.0 product library, for example PLEX600 and then restore the r6.1 System i dispatchers into the PLEX600 library. Application .INI file setting modifications are not needed. The r6.0 created executable will connect to the port number specified on the r6.0 System i environment.

Plex API and Add-In Enhancements

CA Plex has a set of APIs that allow developers to interrogate and update models independent of the tool's graphical interface. The APIs allow users to write wizards to document or automatically customize their models.

For CA Plex r6.1, the APIs have been enhanced to do the following:

■ Increase the range of information that may be interrogated and updated such as action diagram and panel large properties.

■ Allow tasks that are normally automated from within CA Plex to be initiated from outside it, notably generating and building code and setting object flags.

■ Increase the efficiency of processing API data so that the overhead of repeated calls to the API is avoided. This is achieved through the use of SAFEARRAY data types.

■ Allow developers to access selected objects from the CA Plex Object Browser in their wizards.

CA Plex r6.1 registers a new 3.0 version of PlexAPI interface. The previous 2.0 version continues to be supported for existing applications that use it.

Object Browser Integration

Add-Ins are now available from the popup menu of the Object Browser. Through the Plex API, Add-Ins may now be coded to operate on the currently selected objects in the Browser.

Default Add-Ins

The CA Plex installation process now registers some example Add-Ins on the Add-Ins toolbar. You can add or remove Add-Ins as required.

Summary of New Plex API Methods

Method	Description
OpenModel	Opens a local model
CloseModel	Closes a local model
SaveModel	Saves a local model
GetCurrentConfig GetLevelName GetVersionName GetLanguageName GetVariantName	Returns the details of the local model's current configuration.
Execute	Executes a process
Build BuildEnumObjects BuildSafearrayObjects	Generate and build code
FindObj	Existing method extended to find a virtual object if no real object exists.
GetLPText	Existing method extended to non-textual types output in XML form.
SetPropertyText	Existing method extended to non-textual types intput in XML form.
GetFullLPText	Extracts the resolved large property in text for simple large properties or XML form for complex large properties.

Method	Description
MakeReal	Executes the functionality of the Make Real command.
GetObjFlag	Gets an object flag
SetObjFlag	Sets an object flag
EnumSafearrayObjects EnumSafearrayDependants EnumSafearrayNamedObjects EnumSafearrayScopedObjects EnumSafearrayTriplesBySource EnumSafearrayTriplesByTarget EnumSafearrayUnscopedObjects	Enumerate objects and triples

New Product Folder Structure on Windows Vista and Windows XP

CA Plex 6.1 is compatible with Windows Vista such that the tool can be used by a standard user without administrator privileges. As part of these updates, multi-user support has also been added. This means that each user of CA Plex on a Windows system will get their own set of configuration files (e.g. plex.ini) when running the CA Plex development environment.

The group model licensing software has also been upgraded to Crypkey 7.1 as a part of this support.

The default install locations of many CA Plex files and folders have changed compared to previous releases. For example, data files such as the pattern libraries, the null model, sample models, and tutorial files are stored in a single location shared by all users (…\All Users\Documents\CA\Plex\6.1).

You can find convenient shortcuts to these locations in your My Documents folder on Windows XP (or Documents folder on Vista) under the CA\Plex\6.1 subfolder.

When you run CA Plex r6.1 for the first time, CA Plex creates your own personal copy of Plex.ini and other configuration files in your My Documents folder on Windows XP (or Documents folder on Vista) under the CA\Plex\6.1 folder. This means that the copy of Plex.ini under Program Files is not actually used when CA Plex is run, except as a template from which your personal copy of the file is first created.

Note: If you want to build WinNTC functions from a sample model for Windows Vista, we recommend that you build the WinNTC functions locally and then copy the DLLs to your server. This is because the Gen directory that needs to be shared for a file transfer is located under the already shared Documents folder.

WinC .EXE and .INI file Vista Compatibility Updates

For improved Windows Vista compatibility, the following changes have been made to WinC applications. Note these changes also apply to applications deployed on Windows XP.

■ The mechanism for creating and loading application .INI files has changed. For more information, see the New Runtime .INI Search Path section in Chapter 2.

■ Executables are now created with default manifest information that sets the execution level *asInvoker*, which assumes that the application can be executed as a standard user. You can override the manifest at build-time by editing a template manifest file called YourAppExe.manifest.

System i Enhancements

The following sections describe the System i enhancements.

System i Dispatcher

The YOBSYTCPDP dispatcher has been updated for IPv6 compatibility as described earlier in this chapter. This is the recommended version of the dispatcher for use with CA Plex 6.1. Note that user name limitations, when running this version of the dispatcher, are resolved by V5R4 and later. For more information, see the *Getting Started* guide.

Source File Name and Display File Name Limits

RPG and DDS generated source file names for i5/OS may now be up to 10 characters long. Previously file names were truncated at 8 characters.

Display file (DSPF) object names used with RPG IV functions may now be up to 10 characters. Previously these object names were truncated to 8 characters which remains the limit for RPG400 display files.

Java and C# Runtime Scalability Improvements

The scalability of the CA Plex Java and C# runtimes have been improved by reducing the volume of findApp() methods executed by the runtime.

As part of this change, the source code objects in JAVAAPI and CSAPI are updated to use new methods. The previous methods are still supported for backwards compatibility with existing generated code. Also, the parameter interface of the source code objects is unchanged.

New JAR File for Plex EJB Proxy Deployment— ObEJBProxyClient.jar

A new JAR file ObEJBProxyClient.jar has been added for the client-side Plex EJBProxy deployment. This file contains Plex EJB Proxy bean Home and Remote interfaces with stub and tie class files for the IIOP protocol for remote objects.

Improved Support for Native Debugging in Java and C#

The Java and C# generators have been modified to make it easier to debug generated applications using IDEs such as Eclipse and Visual Studio. The changes enable you to easily step into the source code for a called function while debugging. For example, if Function A (Java) calls Function B (also Java) and you are debugging A, you can step into the code for B provided the call is local. Prior to this change you must set a breakpoint at the function call for stepping into the called function (Function B).

This change does not affect performance or backwards compatibility.

.NET Database Query Result Handling

In CA Plex r6.1, the way that CA Plex .NET (C# Server) Runtime handles database query results has been changed to improve runtime scalability. By default, each Plex C# function will close all database query results that it executes.

You can change this behavior using a new triple added to CA Plex r6.1 to denote whether query results must be left open or closed when a function exits. The syntax for this triple is as follows:

```
FNC close query on exit SYS (Yes/No)
```

The default setting Yes indicates that all query results are closed when a function exits. Setting the triple target to No indicates that the query results will remain open when a function call exists, and will be available for subsequent calls to the same function.

BlockFetch and Get Sequential style functions are examples of server functions that require a persistent database query result between calls, so that additional sets of database records can be read between client-server roundtrips. Therefore, this triple has been set to No for both BlockFetch and Get Sequential functions in the STORAGE and OBASE libraries. Note that this triple only affects the C# generator in CA Plex r6.1.

A new CSAPI/SetDatabaseCloseQueryOnExit source code object is also available to control this behavior dynamically.

SQL Native Client with the MARS Connection option

If you use Microsoft SQL Server and the SQL Native Client Provider, you can opt to use the Multiple Active Result Sets (MARS) support. CA recommends MARS support when using SQL Server and SQL Native Client for .NET application development.

For more information, see Accessing Data with .NET Functions in the CA Plex .NET Platform Guide.

.NET Management Console Log File Viewing

The .NET Management Console is enhanced to allow application logging information to be viewed directly in the Management Console. This feature is enabled when the Plex .NET Runtime is running as a service and if the Logging Type is set to FlatText.

The underlying log files now include a timestamp in their file names to ensure that logs are preserved between runs (for example, PlexRuntimeService.TCPIP.1998.20080229_010346.log'. The Java runtime log files follow the same naming convention.
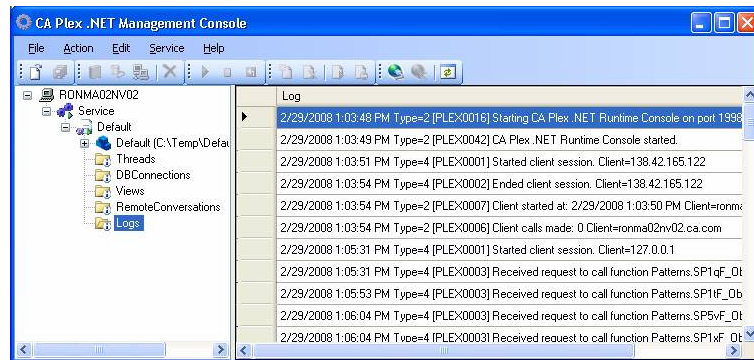
Installing .NET Runtime Service

The CA Plex install program installs .NET Runtime Service and its uninstall program uninstalls the service. CA Plex provides the .NET Runtime check box on the Select Features install screen that is checked by default as shown:
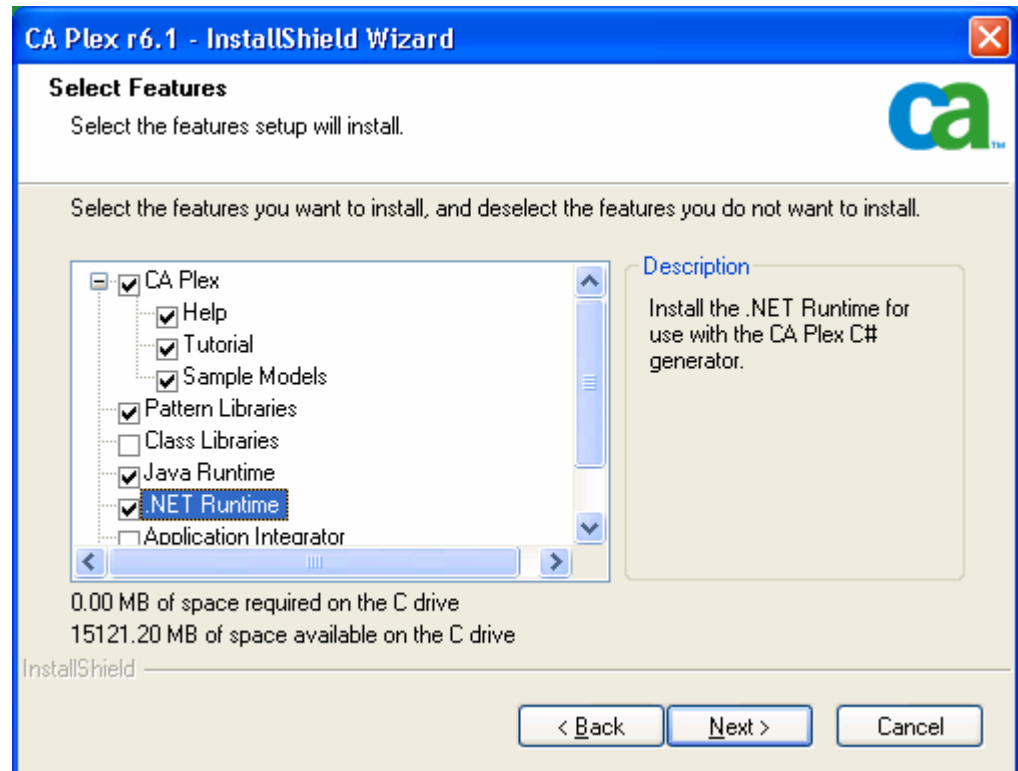
The **Install .NET Runtime Service** and the **Uninstall .NET Runtime Service** menu items on the Start/Programs/CA/CA Plex r6.1/Plex .NET Tools menu have been removed. To install and uninstall the .NET Runtime Service, you can use the bat files - InstallService.bat and UninstallService.bat, that are located in the Plex/ob.Net directory. You must run the bat files with the Run as administrator option.

Summary of Verb Tables Changes

The following table lists the changes to the verb tables:

Verb	Description
FNC close query on exit SYS	Controls database query results.
FNC replaces FLD …by FLD	Enables replacement of inherited fields within the source function.
SRC SQL Statement SYS	Controls the generation of Exec SQL prepared statements in Java and C#. The default value is Prepared.
CDL service PKG	Associates a ServiceConnector package with a code library.
PKG service host type SYS	Specifies the type of host used by a WCF Service. Allowed values are Application, IIS, None, Service or WAS
PKG type SYS	New system value of ServiceConnector
INT endpoint binding NME …address LBL …protocol SYS	Defines the address and protocol of a service endpoint.

Additional Minor Enhancements

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

- The CA Plex version number is removed from the filenames of the Java .properties files used by the CA Plex runtime. They are now called obsrv.properties, obclient.properties and obusr.properties.

- Now the CA Plex Java runtime (Obrun.jar) is compiled with JDK 1.6. The javac1.6 is the default Compiler Alias setting for Plex Java ANT builds.

- Additional source code objects in CSAPI library. ReplaceQuotes and GetWindowsUserId

- Additional source code objects added to JAVAAPI and CSAPI to return SQL state information in Java and C#.

- A confirm prompt is added to the Model Translation Import command that indicates to the local model configuration into which you are importing.

- Administrator privileges are now required to run the Windows C++ Server Build Manager, Dispatch Manager, and Environment Manager.

Features Removed from CA Plex r6.1

The following features have been removed from CA Plex r6.1:

- The COOL:Biz import feature has been removed from Application Integrator.

- Quickstart.mdl sample model—This sample model provided an introduction to the Windows C++ Server deployment environment. The C# or Java generator are now the recommended options for deploying applications to the Windows Server platform.

- JClient sample model—This sample model is superceded by the Java tutorial model described in the *Getting Started* guide.

The Crypkey 4.2 licensing software is no longer supplied on the product CD. CA Plex 6.1 uses Crypkey 7.1 for group model licensing.

Chapter 2: Upgrading from Earlier Releases

This chapter provides and overview of upgrade requirements for upgrading from previous releases of CA Plex.

Before You Begin

Before upgrading, review the readme (Readme.html) for system requirements and late-breaking upgrade notes. Also, check the CA Plex home page at CA Technical Support (http://ca.com/support) for additional information or fixes that may have been made available after this guide was published.

Multiple releases of CA Plex and its generated applications can be installed on the same machine. For example, CA Plex r5.5 and CA Plex r6.1 can be installed and run on the same machine.

Group and Local Model Upgrade

It is a good practice to update local models to the group model regularly. Therefore, we recommend 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 (and build files) after upgrading the group model to the new release.

All group models and associated library models must be upgraded at the same time.

Logging in to a group model causes it to be upgraded to CA Plex r6.1. Thereafter, 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 CA Plex at the same time. Two developers cannot work on the same model simultaneously if they are using different releases of CA Plex.

Local Model Upgrade

It is possible to upgrade a local model from an earlier release simply by opening it with the later release. This is often useful for testing purposes, however as described above this is not the recommended approach for a formal release upgrade.

Upgrading from CA Plex r6.0

This section explains upgrade requirements related to upgrading from r6.0.

Windows C++ Functions Upgrade Requirements

When upgrading from CA Plex r6.0 to CA Plex r6.1 you do *not* need to regenerate or recompile existing C++ functions.

For more information, see the section on C++ Runtime Backwards Compatibility in the chapter "New and Updated Features."

New Runtime .INI Search Path

To better support Windows Vista security requirements and multi-user deployments, the algorithm that the WinC runtime uses to locate and create runtime .INI files has been changed.

Review these changes carefully, especially, if you have developed custom .INI file handling or deploy CA Plex applications in server-based multi-user environments.

Typical Example

In a typical end-user environment, your applications are installed in the Program Files folder including an .ini file. The files in Program Files are read-only for standard users.

When the Plex WinC application starts, it searches for a *writeable* .INI file. If the file is not found, a new .INI file is created in the end-user's personal folder (My Documents on Windows XP, or Documents on Windows Vista). The .INI file is located in a sub-folder with the same name as the executable. For example, MyApp.exe would have the .INI file "My Documents\Myapp\Myapp.ini"

Therefore, in this example, each standard user who runs the application will have their own copy of the application's .INI file.

.INI File Algorithm

The full set of rules is set out below and provides flexibility for other deployment scenarios dependent on the access rights of the user who runs the application:

If a Plex generated executable is called app.exe, then the respective .ini file will be app.ini.

At run time, an application app.exe searches for an app.ini file as follows:

1. In the same directory where the executable (app.exe) is located. If app.ini file is found there and has read-write (RW) access then the executable app.exe uses it.

2. Otherwise, the app.exe searches in the local user \Documents\app\ directory (where app is the name of the executable). If the file app.ini is found there and has RW access, then the executable app.exe uses it.

3. Otherwise, the executable uses the PATH statement to locate an app.ini file that has RW access.

4. If such an app.ini file is still not found, then the executable app.exe creates a new app.ini file inside the local user \Documents\app directory with RW access. This new app.ini will be a copy of the first read-only app.ini found, or, If not found, a blank app.ini is created.

Java, C# and RPG Function Upgrade Requirements

When upgrading to CA Plex r6.1, you do not need to regenerate or rebuild existing Java, C# or RPG functions.

.NET Server Runtime Upgrade Considerations

The following sections explain the NET Server Runtime upgrade considerations.

Version Property

Unless you have a specific technical reason, the Version property for the .NET listener must be left blank. The value of *600* must be deleted if it is present.

At CA Plex r6.1, if this property is empty, the default version number from the Plex .NET runtime is taken.

The Version property is provided for backwards compatibility reasons. For example, to use r6.0 WinC clients to connect to the r6.1 version of .NET runtime, you must set this property to *600*.

Upgrading from CA 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 CA Plex releases earlier than r6.0 must be recompiled to be compatible with the CA Plex r6.1 C++ run-time system.

A single Windows application cannot contain DLLs created with different releases of CA Plex (except in the case of 6.0 and 6.1). CA 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 incompatible releases. In your development environment, you need to rebuild all of your application DLLs when you move from CA Plex r5.5 or earlier up to CA Plex r6.1.

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

For more information about the new features in CA 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 CA 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 CA Plex r6.1 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 CA Plex r6.0 SP1 product.

Upgrading from CA Plex r5.5

If you regenerate a Java function at CA Plex r6.1, then you must also regenerate all other 5.5-generated functions in the call graph of that Function. This is due to changes in the parameter formats at CA Plex r5.5 SP1 and later. We recommend a full regeneration and recompilation of all Java functions when upgrading from CA 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 CA 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 CA Plex r6.1 Java run time (obrun.jar) with a CA Plex r5.5 Java application, you need to have these settings in your ob600xxx.properties file:

```
Version=610
```

```
SPVersion=0
```

Java and C# Exec SQL action diagram source code

In CA Plex 6.0 and 6.1, the Java and C# generators generate all Exec SQL statements as prepared statements by default. The SRC SQL statement type SYS triple can be used to indicate source code objects that must not be generated with prepared statements.

Generating prepared statements may improve performance and scalability in cases where the same SQL statement is executed repeatedly. However, the results may also depend on application and DBMS-specific factors. CA recommends benchmarking with different values for SRC SQL statement type SYS for determining the optimum setting in each case.

Also, generating prepared statements assumes that all parameters to the Exec SQL statement correspond to columns in a table. This assumption is not always true because the Exec SQL statement can be used to implement a wide variety of SQL code. The SRC SQL statement type SYS triple can be used to indicate source code objects that should not be generated with prepared statements. When upgrading Java applications from r5.5 or earlier, it may be necessary to add such triples to some source code objects to avoid errors.

This change does not affect Exec SQL usage with C++.

Java and Oracle varchar null support

Java applications that target the Oracle database and use optional varchar fields may behave differently after upgrading from r5.5 or earlier. Depending on how the varchar is modeled in Plex, unexpected optimistic locking errors may occur after upgrade. For example:

"Instance of .... changed by another user"

The typical solution is to add FLD null VAL triples to the varchar field where the target value is empty. For more information, search the CA Support Online web site for problem CPLEX 1296.

Java Source Code when upgrading from CA 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 CA 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 CA Plex r6.1.

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 CA Plex Windows Application Server environment settings have changed because of the rebranding from Advantage to AllFusion.

The CA Plex r6.1 installation program automatically copies pre-existing environment settings to the new key values. However, if you do not see the copied settings in the CA 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 CA Plex r6.1.

Build (.BLD) file compatibility

As discussed earlier, 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 CA Plex r6.1, CA 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 CA Plex 6.1.

Pre 6.0 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 CA Plex r5.1

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

Inheritance Resolution Changes in CA Plex r5.5

At r5.0 of CA 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. CA Plex r5.1 included some fixes for these problems, but some significant problems remained.

CA 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

CA 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 CA 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 CA Plex determines which of the available functions should actually be called when it is loaded.
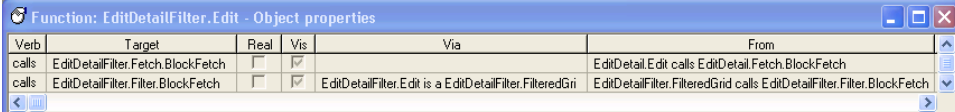
Prior to r5.0, CA Plex did not explicitly handle such action diagram calls. As a result, the resolution of the call did not always follow the usual rules of the inheritance engine. At r5.0 and later a mechanism was put in place to handle such calls. This mechanism has been refined a number of times, 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:

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 CA 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 CA 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:

This type of scenario is known as *diamond inheritance* due to the shape of the resulting inheritance hierarchy as shown by the following diagram.
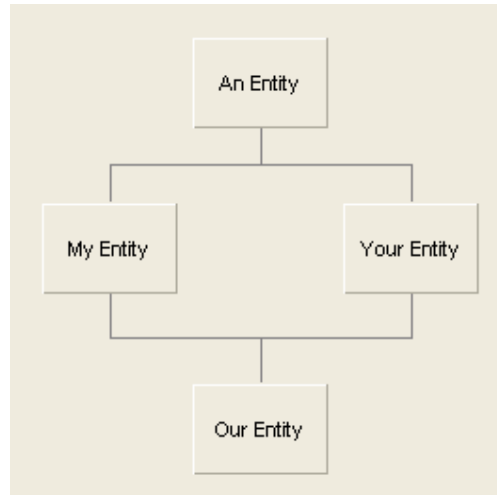
The inheritance hierarchy starts at *An Entity*, diverges, and then the two branches are brought back together at *Our Entity*. This situation is quite complex but the examples in real customer models are often even more complex with further branches and layers of inheritance involved.

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 CA 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 CA 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 CA 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, CA 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 CA Plex since 1.0. Consequently, they represent a mature technology that has been implemented successfully for many years by many CA Plex customers. When upgrading from an earlier release of CA 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.

A separate option is available for each type of control:

```
[NativeControls]
;GUI controls either display as MFC controls (1) or Winwidgets (0).
;These settings have no effect on the WinC grid which is always Winwidgets
ListBox=1
SpinButton=1
SingleLineEdit=1
SingleLineEditNumeric=1
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. The MultilineEdit setting is an exception—Setting this value to 0 invokes an earlier implementation of the native control as Winwidgets do not provide a multiline edit control.

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

Upgrade Issues with MFC Controls

Known upgrade issues are listed below. Report any undocumented upgrade issues to CA Technical Support if they cause a problem in your application.

- **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 CA 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 CA 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 CA Plex r4.5

This section explains upgrade requirements related to r4.5.

Inheritance and Property Resolution Changes

CA 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 CA 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 CA 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 CA 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 CA 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 CA 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 CA 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 CA Plex r4.0

This section explains upgrade requirements related to r4.0.

Replacement and Scoped Objects

CA 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 CA Plex r3.5

In addition to the instructions in Upgrading from CA 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 CA 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 CA Plex library on the System i. When building the System i Field Values File within CA 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 CA 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 CA 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

CA 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 CA 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 CA 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 CA Plex r4.0. If you have previously used one of these patterns in CA 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 CA Plex r3.1 and r3.0

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

Upgrading from CA Plex 2.51 and Earlier

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