Varianted Source Code Solution For CA Plex

Roger Griffith – United Heritage Life Insurance

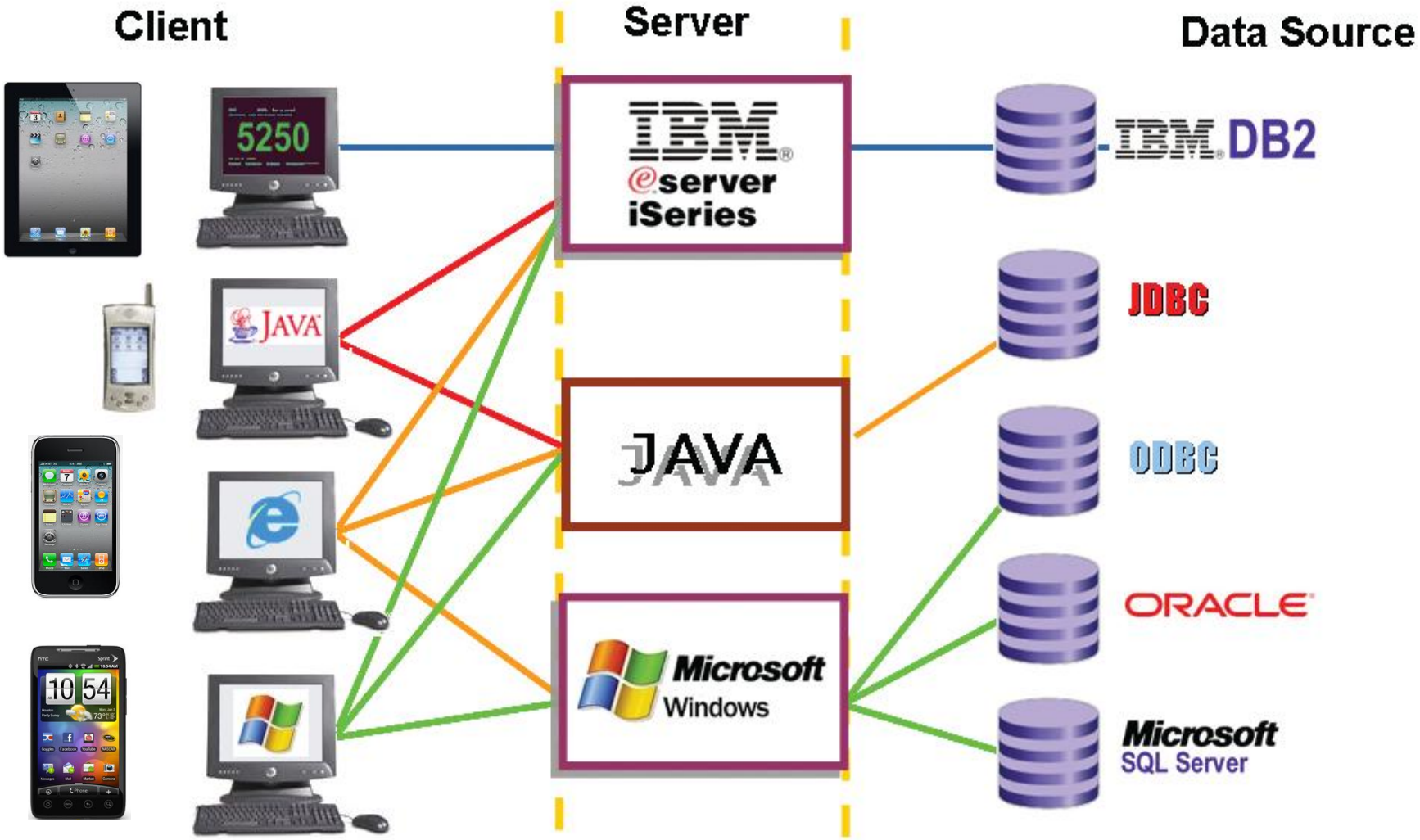
Session 3C

ca technologies

Speaker Bio

- Bachelors degree in Information Sciences from Boise State University
- Plex Lead Developer at United Heritage Life Ins.
- Started model based development in 1988 using Synon 2E
- Have been developing enterprise solutions with Plex since 1997
- Evangelist of Plex as premier development tool
- Proponent of the power of Patterns
- In 10th Year as President of NWPD User Group (http://www.nwpdug.org)

the power Of Plex – one model, many dimensions



Holy Grail – seamless multi-platform applications

From Plex White Papers…

AllFusion® Plex

Seamless multi-platform application development and deployment has always been one of the "Holy Grail" goals of computer technology. For the software vendor, the goal may be to develop an application that can be sold on most commercially available computer platforms. For a large organization with many enterprise platforms in use, the goal may be to deploy n-tier applications across heterogeneous hardware and operating systems.

Wouldn't it be great if you could model your application in an enterprise development tool, and then have the tool generate application code that is optimized to your target platforms? If this were possible, you could deploy C# / SQL Server code for Windows-based servers, and J2EE Java/RPG/DB2 code to iSeries based servers. Actually, this is something that is not all that futuristic - CA AllFusion Plex is an Architected Rapid Application Development (ARAD) tool that can deliver on this promise today. It does not do magic - there is always some application code that must be customized for each deployment platform. However, it can get you 95% of the way there automatically, and even the last 5% can be handled within the AllFusion Plex managed modeling environment.

multi platform is critical to growth of Plex applications

- Most Plex shops have developed their client/server applications for Windows Client and iSeries RPGIV Server
- As Plex continues to evolve, you want to be in a position to generate your applications in the latest variants to take advantage of modern state of the art platforms like Java and .Net
- If you are looking to extend your applications with WebClient or mobile iPhone/iPad/Android devices, you will need to generate your client functions in Java (and possibly support both WinC and Java)

variants are key to multi platform solutions

- Establish the operating environment of an application
- Specifies a target hardware and software platform
- You specify which variant you want to use in your model configuration
- Typically the base variant holds the content that is common to all operating environments
- Variants allow you to add content that only exists in that variant
- Each variant has content that is specific to that variant

* Use the footnote layout if you need to source.

pattern libraries

- The pattern libraries accommodate for all of the supported variants
- CA has gone to great lengths to ensure the highest degree of parity across all variants
- Goal is to ensure that your application will have same look/feel/functionality regardless of platform generated for
- Not all features are available in all variants
  - Platform specific limitations require that some features are not available to all variants
  - For example, you cannot dynamically re-sort a grid in the Java variant

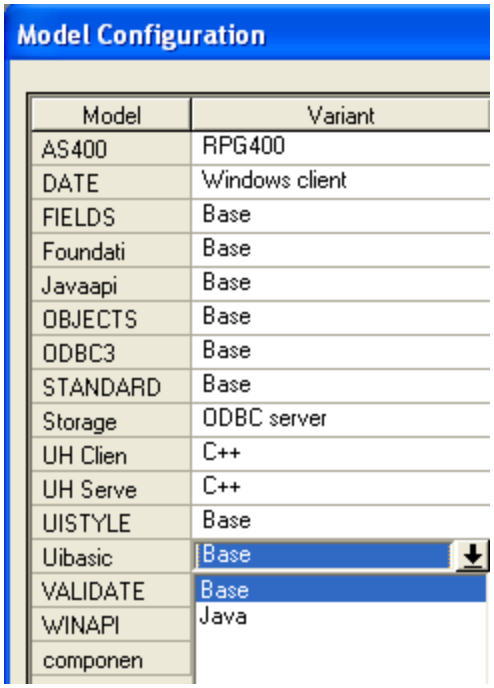
number of Plex variants continues to grow

- Early versions of Plex (Obsydian) included WinC and RPG400 variants
- As Plex evolved, WinNTC, Java, C#, and RPGIV have been added
- Ideally we would like to be able to take advantage of new platforms as they become available, with little to no modifications to our applications
- As number of variants grows, the challenge of supporting all of the variants from a single model is increasingly difficult
- New .Net client variant on horizon?

Plex class libraries that offer variants

Client

Server

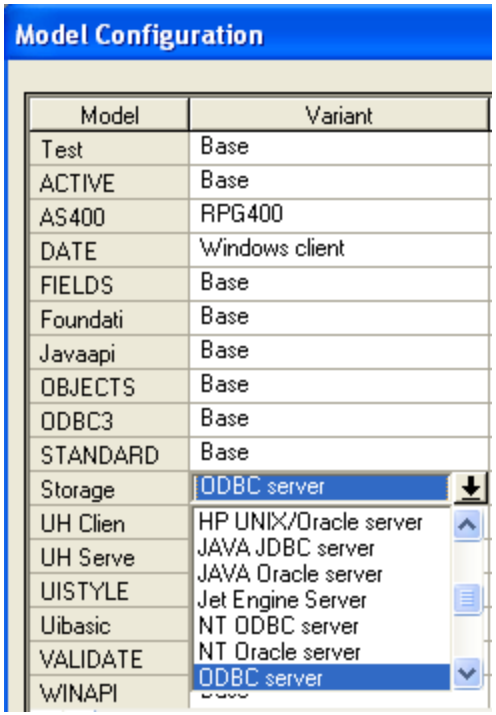


— Active

— Date

— UIStyle

— UIBasic



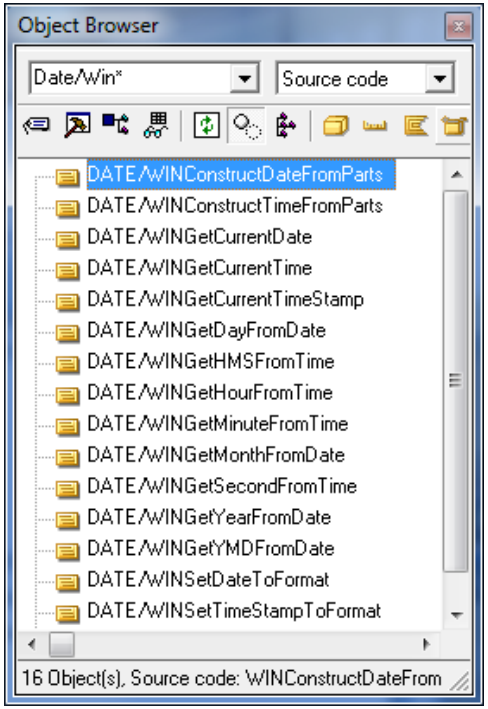
AS400 Storage

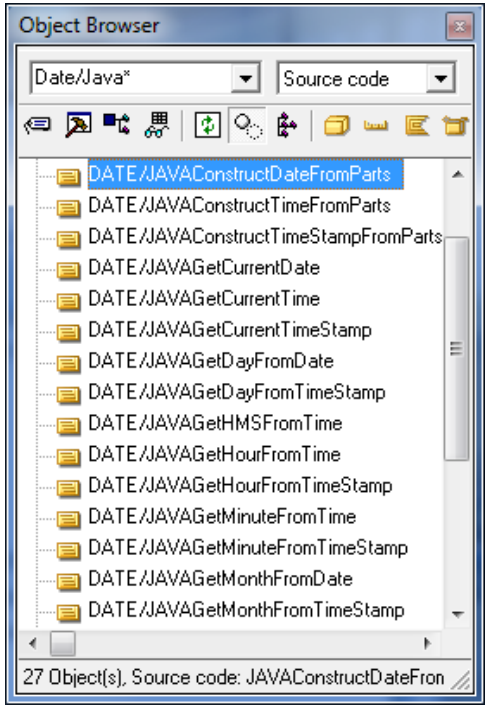
challenges of supporting multi platform

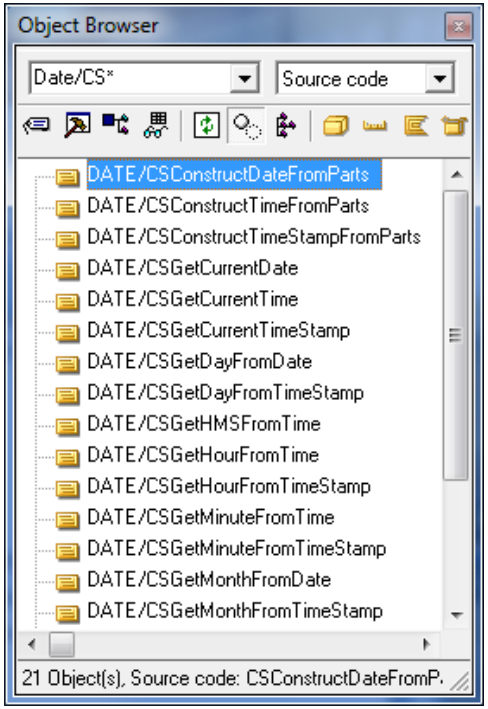
- ActiveX Controls vs. Java Beans
  - Each ActiveX control used requires an equivalent Java Bean
- Use Of Source Code
  - Source Code allows us to extend the capabilities of Plex by embedding language specific source code in our action diagrams
  - If you wish to support all variants, you have to include an API Call for each language you support
    - As this list of variants grows, this becomes very tedious to support
    - As new variants are added, you have to visit everywhere API Calls are used to add code for new variant

source code provided by Plex

- Separate members for each platform (Windows, Java, C#)
- No source code for RPG (most popular platform), go figure
- Each member has syntax specific to that language







Plex implementation of source code in pattern libraries

Where ever source code is used in the pattern libraries, they include a call to each source code member for the supported platforms, and then embed the appropriate source code during the generation process by testing a meta variable

Create Meta Variable For Language

```
Pre Point   End initialize
  Go Sub   Set panel caption
  Call   Uibasic/Meta.Options
    Copyright © 1994-1998 by Sterling Software, Inc. All Rights Reserved.
    Define meta variables for possible function options
    Not all of these options are used by the pattern library functions
    +++Define   Field: FIELDS/+Normal
    +++Define   Field: FIELDS/+Function
    +++Set Value To Current   Field: FIELDS/+Function
    +For Defined Value   Field: FIELDS/+Function
      +If   FNC language SYS, System: WinC
        +++Define   Field: FIELDS/+WinC
      +If   FNC language SYS, System: RPG400
      +If   FNC language SYS, System: RPGIV
      +If   FNC language SYS, System: Java
        +++Define   Field: FIELDS/+JAVA
      +If   FNC language SYS, System: C#
      +If   FNC language SYS, System: HPUXC
```

Test Meta Variable And Call Source Code For That Language

```
+If   Field: FIELDS/+WinC
  API Call   Source code: WINAPI/SetCaption

+If   Field: FIELDS/+JAVA
  API Call   Source code: Javaapi/SetCaption
```

typical developer use of source code

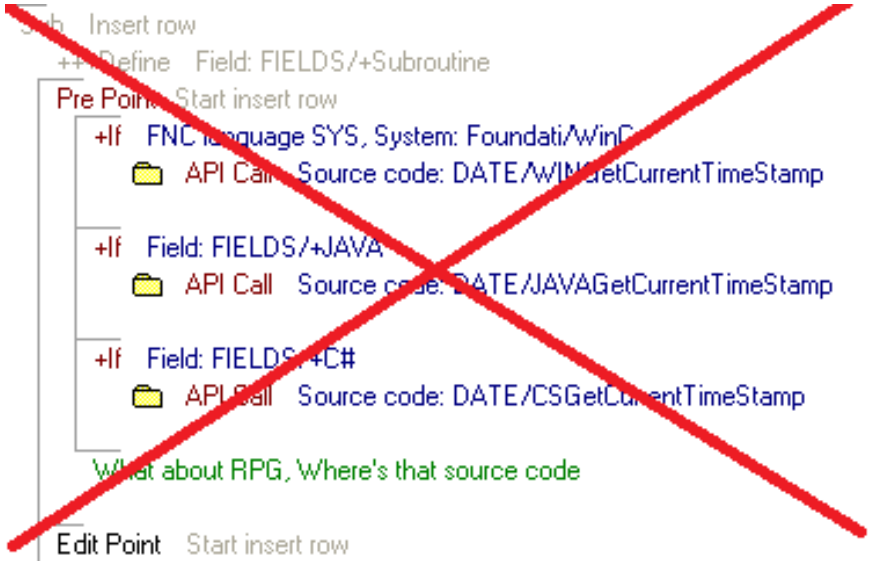
- Developers tend to only add the source code for the variants that you are supporting today
  - If you are not currently generating Java or C#, you are probably not adding source code for those platforms
  - Who programs for the future right?
- The first time you try to generate for a new variant, your functions won't generate because your source code is incompatible with that language
- You may have thousands of functions that you will need to "visit"

a better solution for implementing source code

- What if you could simply call one source code member that contained the correct syntax for all languages
- You would no longer need to add multiple API Call statements (one for each platform you wish to support)
- As new variants are introduced in Plex, you would never have to visit all of your functions that include source code to add the API Call statements for the new variant
- Source Code would no longer be a challenge that inhibits your ability to take on new variants as they become available

which approach would you prefer to use

Traditional Approach



Varianted Source Code Solution (One call..that's all)

varianted source code solution

- We created two source code models (one for client and one for server) that contain a suite of source code members
- Each model contains variants for all supported languages
  - Client model includes variants for C++ and Java
  - Server model includes variants for C++, C#, Java, and RPGIV
- Simply attach these models as class libraries and use them
- When new variants are introduced in Plex (possibly a C# client), we will add that variant (and necessary source code for that variant) to these models (you can simple re-extract to get new source code)
- When making API Calls to source members, your only decision is whether you are calling from a client or server function

new varianted source code models

Client Model (VSrcClient)

Object Browser

vsrccli* | Source code

VSrcClie/CheckIfNumeric
VSrcClie/ConstructDateFromParts
VSrcClie/ConstructTimeFromParts
VSrcClie/ConstructTimeStampFromF
VSrcClie/Exponentiation
VSrcClie/GetCurrentDate
VSrcClie/GetCurrentTime
VSrcClie/GetCurrentTimeStamp
VSrcClie/GetDateFromTimestamp
VSrcClie/GetDayFromDate
VSrcClie/GetDayFromTimeStamp
VSrcClie/GetGlobalChar
VSrcClie/GetGlobalChar2
VSrcClie/GetGlobalNumeric
VSrcClie/GetGlobalNumeric2
VSrcClie/GetHMSFromTime
VSrcClie/GetHourFromTime
VSrcClie/GetHourFromTimeStamp
VSrcClie/GetMinuteFromTime
VSrcClie/GetMinuteFromTimeStamp

41 Object(s)

Object Browser

vsrccli* | Source code

VSrcClie/GetMonthFromDate
VSrcClie/GetMonthFromTimeStamp
VSrcClie/GetPositionInString
VSrcClie/GetSecondFromTime
VSrcClie/GetSecondFromTimeStamp
VSrcClie/GetStringLength
VSrcClie/GetUserID
VSrcClie/GetYearFromDate
VSrcClie/GetYearFromTimeStamp
VSrcClie/GetYMDFromDate
VSrcClie/MakeAllLower
VSrcClie/MakeAllUpper
VSrcClie/ReplaceQuotes
VSrcClie/SetCaption
VSrcClie/SetGlobalChar
VSrcClie/SetGlobalChar2
VSrcClie/SetGlobalNumeric
VSrcClie/SetGlobalNumeric2
VSrcClie/TriggerEvent
VSrcClie/Trim
VSrcClie/TrimRight

41 Object(s)

Server Model (VSrcServer)

Object Browser

vsrcser* | Source code

VSrcServ/ConstructDateFromF
VSrcServ/ConstructTimeFromF
VSrcServ/ConstructTimeStamp
VSrcServ/Exponentiation
VSrcServ/GetCurrentDate
VSrcServ/GetCurrentTime
VSrcServ/GetCurrentTimeStam
VSrcServ/GetDateFromTimest
VSrcServ/GetDayFromDate
VSrcServ/GetDayFromTimeSta
VSrcServ/GetHMSFromTime
VSrcServ/GetHourFromTime
VSrcServ/GetHourFromTimeStamp
VSrcServ/GetMinuteFromTime
VSrcServ/GetMinuteFromTimeStamp
VSrcServ/GetMonthFromDate
VSrcServ/GetMonthFromTimeStamp
VSrcServ/GetSecondFromTime
VSrcServ/GetSecondFromTimeStamp
VSrcServ/GetStringLength
VSrcServ/GetYearFromDate

27 Object(s)

Object Browser

vsrcser* | Source code

VSrcServ/GetYearFromTimeStamp
VSrcServ/GetYMDFromDate
VSrcServ/MakeAllLower
VSrcServ/MakeAllUpper
VSrcServ/Trim
VSrcServ/TrimRight

27 Object(s)

attach as class libraries and set variants as needed

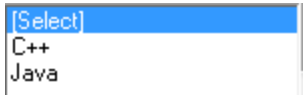
When using the new Source Code in these new Class Libraries, the only decision that needs to be made at design time is whether you are adding the Source Code to a Client Function or a Server Function. You will notice that many of the Source Code members are in both models (for example _GetCurrentDate).

Model Configuration

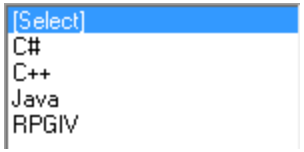
Model	Variant	Language	Version	Level
Test	Base	Base	Base	Base
ACTIVE	Base	Base	V6.1 Patterns	V6.1 Patterns
AS400	RPG400	Base	V6.1 Patterns	V6.1 Patterns
DATE	Windows client	Base	V6.1 Patterns	V6.1 Patterns
FIELDS	Base	Base	V6.1 Patterns	V6.1 Patterns
Foundati	Base	Base	V6.1 Patterns	V6.1 Patterns
Javaapi	Base	Base	V6.1 Patterns	V6.1 Patterns
OBJECTS	Base	Base	V6.1 Patterns	V6.1 Patterns
ODBC3	Base	Base	V6.1 Patterns	V6.1 Patterns
STANDARD	Base	Base	V1.0	V1.0
Storage	.NET server	Base	V6.1 Patterns	V6.1 Patterns
UISTYLE	Base	Base	V6.1 Patterns	V6.1 Patterns
Uibasic	Base	Base	V6.1 Patterns	V6.1 Patterns
VALIDATE	Base	Base	V6.1 Patterns	V6.1 Patterns
VSrcClie	C++	Base	Base	Base
VSrcServ	C#	Base	Base	Base
WINAPI	Base	Base	V6.1 Patterns	V6.1 Patterns

each source code member has correct code for variant

VSrcClient Variants



VSrcServer Variants



VSrcServ | Java

Source code: [(Read-Only)]GetMinuteFromTime

```
&(1:).assign(&(2:).getTimeMinute());
```

Parameters
- &(1:) Minute
- &(2:) TimeISO

VSrcServ | C++

Source code: [(Read-Only)]GetMinuteFromTime

```
&(1:) = &(2:).GetTimeMinute();
```

Parameters
- &(1:) Minute
- &(2:) TimeISO

VSrcServ | C#

Source code: [(Read-Only)]GetMinuteFromTime

```
&(1:).assign(new ObIntFld(&(2:).TimeMinute));
```

Parameters
- &(1:) Minute
- &(2:) TimeISO

VSrcServ | RPGIV

Source code: [(Read-Only)]GetMinuteFromTime

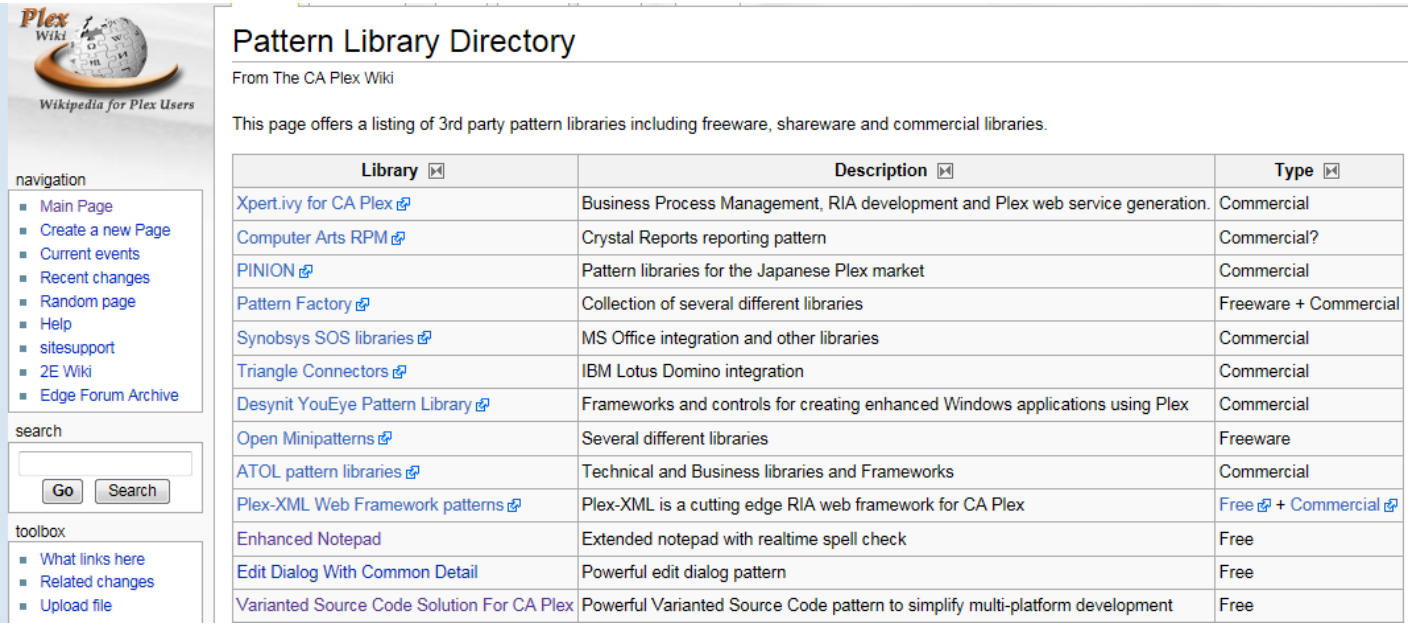
```
C      &(2:)      DIV      100      &(1:)
```

Parameters
- &(1:) Minute
- &(2:) TimeISO

how do I implement varianted source code as painless as possible

You basically have two options…

1. You could create your own varianted source code models and enter all of the required source code for every variant (don't forget to test every member in every variant)
2. Or you can simply visit the Plex WIKI and download the new varianted source code models absolutely free!



Plex Wiki

Wikipedia for Plex Users

navigation

- Main Page
- Create a new Page
- Current events
- Recent changes
- Random page
- Help
- sitesupport
- 2E Wiki
- Edge Forum Archive

search

Go Search

toolbox

- What links here
- Related changes
- Upload file

Pattern Library Directory

From The CA Plex Wiki

This page offers a listing of 3rd party pattern libraries including freeware, shareware and commercial libraries.

Library	Description	Type
Xpert.ivy for CA Plex	Business Process Management, RIA development and Plex web service generation.	Commercial
Computer Arts RPM	Crystal Reports reporting pattern	Commercial?
PINION	Pattern libraries for the Japanese Plex market	Commercial
Pattern Factory	Collection of several different libraries	Freeware + Commercial
Synobsys SOS libraries	MS Office integration and other libraries	Commercial
Triangle Connectors	IBM Lotus Domino integration	Commercial
Desynit YouEye Pattern Library	Frameworks and controls for creating enhanced Windows applications using Plex	Commercial
Open Minipatterns	Several different libraries	Freeware
ATOL pattern libraries	Technical and Business libraries and Frameworks	Commercial
Plex-XML Web Framework patterns	Plex-XML is a cutting edge RIA web framework for CA Plex	Free + Commercial
Enhanced Notepad	Extended notepad with realtime spell check	Free
Edit Dialog With Common Detail	Powerful edit dialog pattern	Free
Varianted Source Code Solution For CA Plex	Powerful Varianted Source Code pattern to simplify multi-platform development	Free

let’s take this new varianted source code solution out for a test drive…

ca technologies

summary

- Plex is the premier development tool on planet that offers true multi platform applications from a single design
- Plex continues to add new variants to expand the number of supported platforms
- The new opportunities available to us using Plex demand that we support variants beyond traditional WinC and RPGIV
- Source code ***can be*** a major challenge, but using varianted source code simplifies development, and allows you to seamlessly take on new variants as they become available

thank you

ca technologies