

Lightning Talk

Use of CA Plex Model API



Sergio Mendes
Lusodata



Client Infrastructure

Client / Server:

- IBM i (AS400) Servers
- Windows PC Clients

Software Updates Distribution (DevOps)

Development:

- Portugal

Tests/QA/Production Environments:

- Portugal
- Greece
- Egypt
- Saudi Arabia
- Morocco
- Spain

Object Collection and Distribution

Server Objects:

- Collected and distributed manually
- Manually updated by SysAdmin

Client Objects:

- DLLs / PNLs / EXEs
- How to collect the correct objects in a controlled way ?
- Deployed objects would have to be fetched by a “check for updates” mechanism

Know what objects to collect

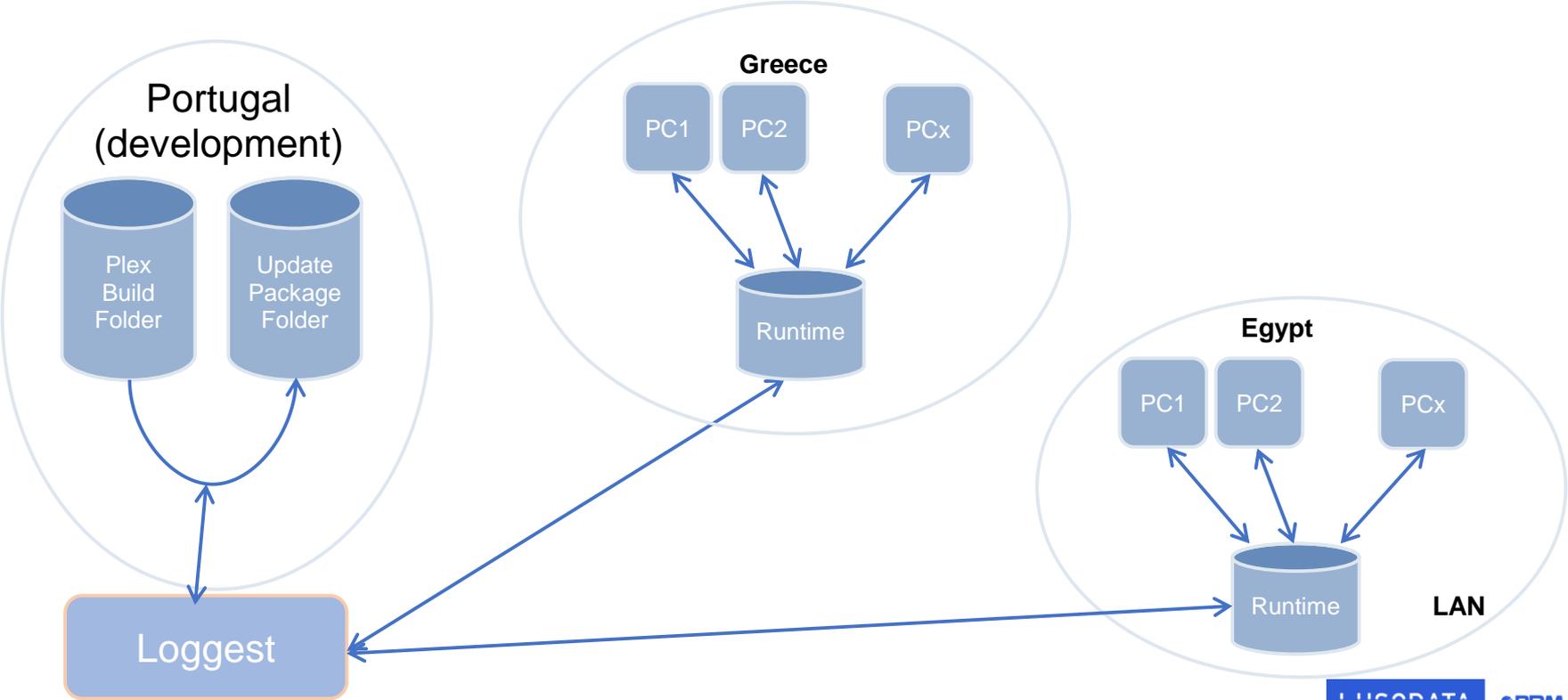
Plex Lists:

- Are the main repository for logical object collection at development time
- Functions in the list are compiled, analyzed and their DLLs/PNLs stored inside “Update Packages”.

Solution:

- Make use of “CA Plex Model API” to process those lists
- Develop a Plex App, called: “LogGest”

Software Updates Distribution



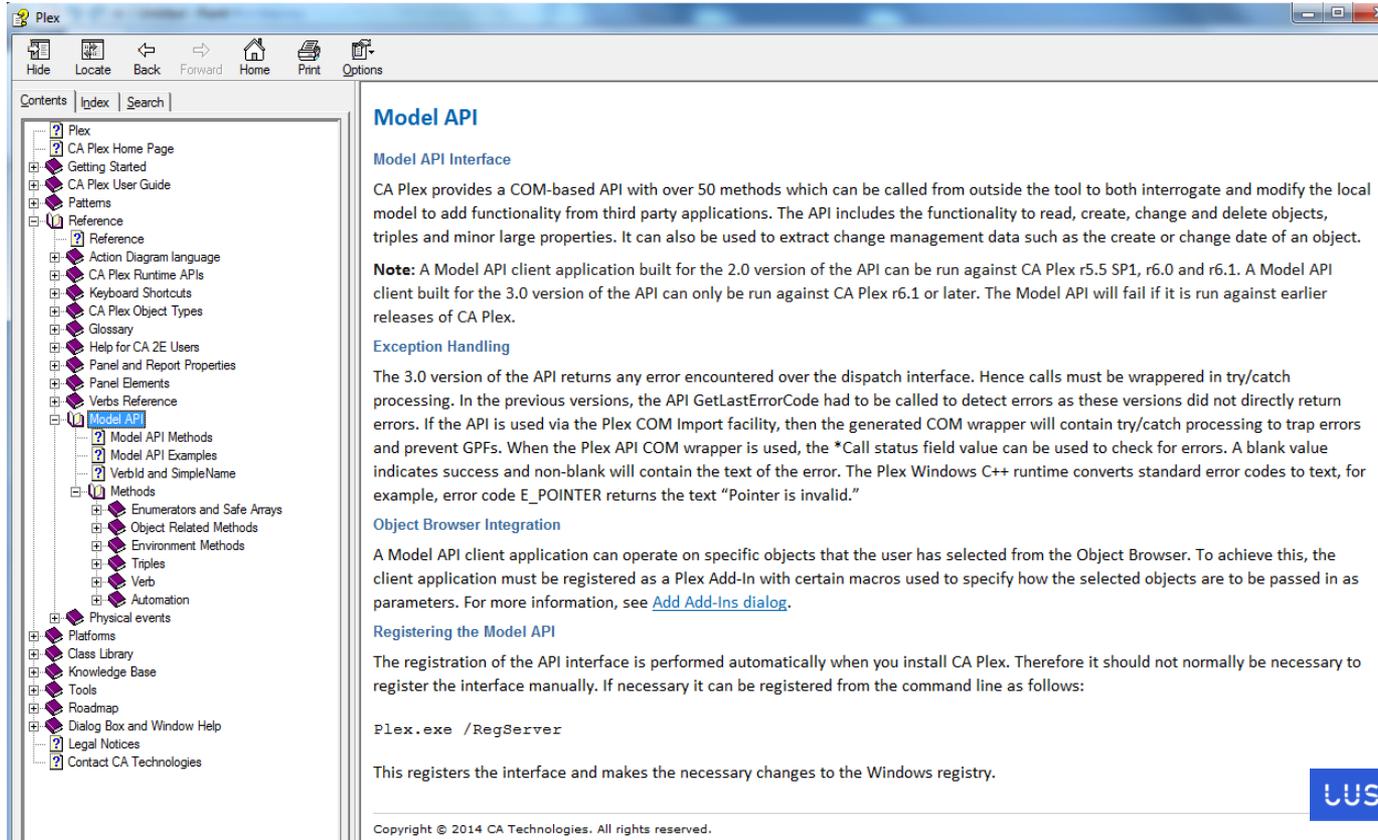
The Model API (using C++)

Plex Model API: COM-based API with over 50 methods

- Allows external applications to interrogate and even modify Local Models.
- Can be used to extract “Change Management” data
- Documented in Plex Help
- Used version 1.0 back in 2007

Current Version is 3.0, has better exception handling.

Plex Model API Documentation



Model API

Model API Interface

CA Plex provides a COM-based API with over 50 methods which can be called from outside the tool to both interrogate and modify the local model to add functionality from third party applications. The API includes the functionality to read, create, change and delete objects, triples and minor large properties. It can also be used to extract change management data such as the create or change date of an object.

Note: A Model API client application built for the 2.0 version of the API can be run against CA Plex r5.5 SP1, r6.0 and r6.1. A Model API client built for the 3.0 version of the API can only be run against CA Plex r6.1 or later. The Model API will fail if it is run against earlier releases of CA Plex.

Exception Handling

The 3.0 version of the API returns any error encountered over the dispatch interface. Hence calls must be wrapped in try/catch processing. In the previous versions, the API `GetLastErrorCode` had to be called to detect errors as these versions did not directly return errors. If the API is used via the Plex COM Import facility, then the generated COM wrapper will contain try/catch processing to trap errors and prevent GPFs. When the Plex API COM wrapper is used, the `*Call` status field value can be used to check for errors. A blank value indicates success and non-blank will contain the text of the error. The Plex Windows C++ runtime converts standard error codes to text, for example, error code `E_POINTER` returns the text "Pointer is invalid."

Object Browser Integration

A Model API client application can operate on specific objects that the user has selected from the Object Browser. To achieve this, the client application must be registered as a Plex Add-In with certain macros used to specify how the selected objects are to be passed in as parameters. For more information, see [Add Add-Ins dialog](#).

Registering the Model API

The registration of the API interface is performed automatically when you install CA Plex. Therefore it should not normally be necessary to register the interface manually. If necessary it can be registered from the command line as follows:

```
Plex.exe /RegServer
```

This registers the interface and makes the necessary changes to the Windows registry.

Copyright © 2014 CA Technologies. All rights reserved.

Plex Model API – some source codes

```
#include <wizardapidefs.h>
#include "import.h"

&(3:) = (ObLongFld) CoInitializeEx (NULL, (DWORD) &(2:));
```

Parameters

- &(1:) MDDebugOn ?
- &(2:) dwCoInit
- &(3:) hrReturnCode

&(2:) -> Concurrency Level (usually, COINIT_MULTITHREADED)

&(3:) -> Return Code (S_OK is good)

Plex Model API – get List Ptr

```
#include <wizardapidefs.h>
#include "import.h"
{

    int ret_val;
    long listobject = 0;

    try {
        IPlexAPIPtr p(__uuidof(PlexAPI));
        ret_val = p->FindObj((LPCTSTR)&(1:), _TypeList, &listobject);

        p = NULL;
    }
    catch (_com_error e) {
        AfxMessageBox(e.ErrorMessage());
    }

    &(2:) = (ObLongFld)listobject;
}


```

List Name (parameter 1)

List object pointer (used by other APIs)

Parameters

- &(1:) LG List Name
- &(2:) LG List Ptr

Plex Model API – get FNC name

```
#include <wizardapidefs.h>
#include "import.h"
#include <atlconv.h>

#ifdef av_USES_CONVERSION
#define av_USES_CONVERSION
USES_CONVERSION;
#endif
{
    &(2:) = " ";

    int ret_val;

    try {
        IFlexAPIPtr p(__uuidof(PlexAPI));
        BSTR name = 0;
        ret_val = p->GetObjName((long) &(1:), &name);

        &(2:) = (LPCSTR)W2A(name);
        p = NULL;
    }
    catch (_com_error e) {
        AfxMessageBox(e.ErrorMessage());
    }
}
}
```

Function object pointer

Function's name

Parameters

- &(1:) LG Object Ptr
- &(2:) LG Object Name

Plex Model API – last step

Unitialize (free) COM

```
CoUninitialize();
```

LogGest DEMO

LUSO DATA

PDM Company

Contact



to fill in



sergio.mendes@lusodata.pt



www.lusodata.pt