http://www.lansa.com

LANSA

Newsletter
September 2005

LANSA Pet Shop Benchmark

LANSA the clear winner in productivity

LANSA 2005 *offers a* ***dramatically more productive*** *way to create high quality next-generation applications compared to J2EE and .NET.*

In 2001, Sun's original Java Pet Store Application was created to provide developers with a sample program built using Sun's J2EE best practices (*Information on the Pet Shop and Pet Store applications can be found at the following URLs: Microsoft MSDN .NET Pet Shop and Sun Developer Network Java Pet Store). Soon after, Microsoft, in conjunction with Vertigo, reproduced the Pet Store application using the .NET framework and called it the .NET Pet Shop. Recently, LANSA replicated the Pet Shop application using its own development and integration tools. **LANSA completed the application in days rather than the many weeks taken using J2EE or .NET.****

**Information on Benchmark Data

Number of Lines:	LANSA used an application called Code Counter by GeroneSoft to count the lines of code. Blank lines, comments and HTML tags were not included when counting the lines of code.
Number of Days:	The .NET number (i.e., 40) is referenced in an interview with Scott Stanfield The J2EE number (i.e., 120) is a calculation based on the ratio of lines to days in relationship to the .NET Version (125 lines per day).

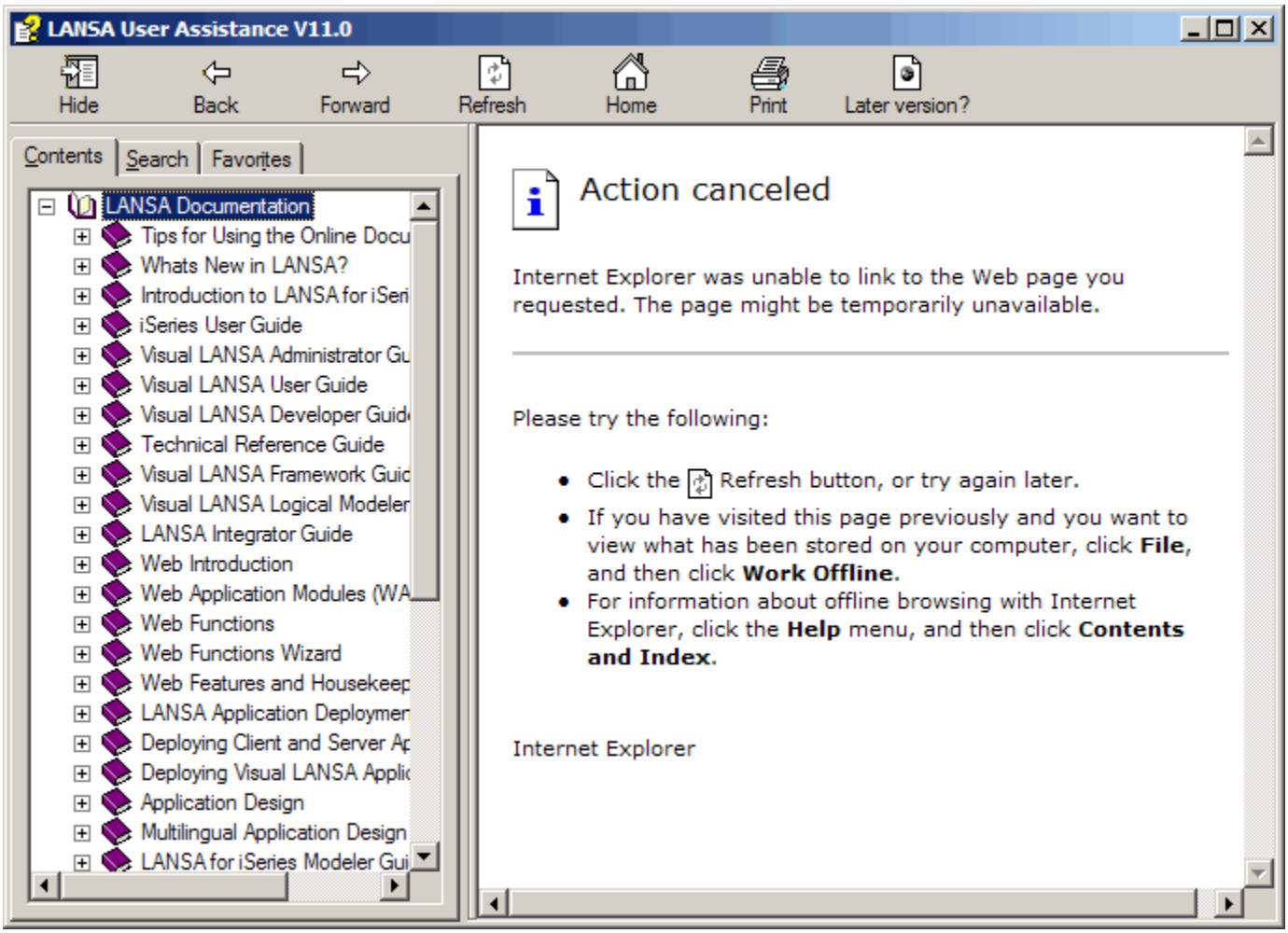
In This Issue

After applying Windows Server 2003 Service Pack (SP1) cannot open .CHM files unless located on local computer

Description

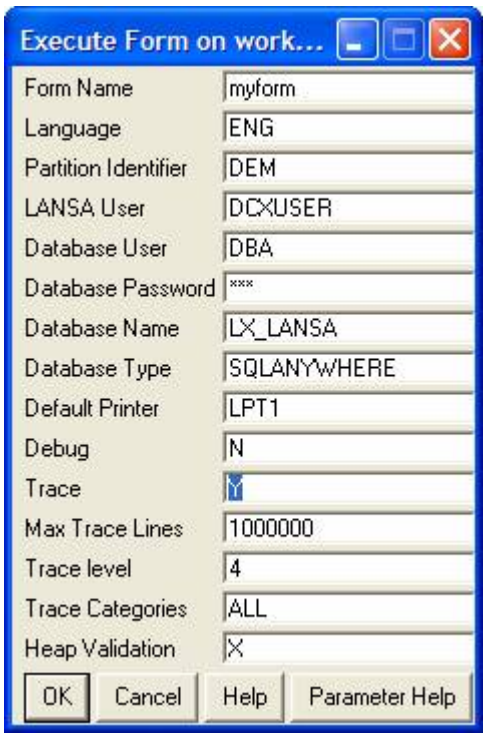
Certain Web applications using the InfoTech protocol after you install Microsoft Windows Server 2003 Service Pack (SP1), MS05-026, or MS04-023 may not work. An example found is when LANSA Help Documents in .CHM format are stored on the network drive.
Visit: http://support.microsoft.com/kb/896054 for details on how to allow certain Web Applications to use the InfoTech protocol. Alternatively install the LANSA Documentation guides to your local PC.

Problematic screenshot as shown:



Improved Tracing in LANSA 2005

Sometimes you may be asked by LANSA support to provide a trace of an executing application to aid in problem diagnosis.
Typically you do this by putting a Y into the "Trace" option when starting the application:



You then execute the application until the problem are is encountered and then look for files named X_Trace*.* to send back to the LANSA support person.

Tracing is actually controlled by a series of ITxx= parameters on the LANSA X_RUN command where IT stands for *Internal Trace*.

Sometimes the approach to tracing from the start of your application is problematic because so much trace data may be generated by your application *before* you get to the problem area.

In LANSA 2005 it is possible to selectively turn tracing on and off by commands embedded within your application. This allows you to turn tracing on immediately before entering the problem area, thus drastically reducing the amount of trace data produced.

Here's some examples of how to dynamically manipulate tracing from within your application:

Checking if tracing is ON or OFF

```
USE GET_SESSION_VALUE (ITRO) (#Trace #RetCode)

If (#Trace = Y)
<< tracing is currently on>>
Else
<< tracing is currently off>>
Endif
```

Turning Tracing ON or OFF
USE SET_SESSION_VALUE (ITRO N) turns tracing off
USE SET_SESSION_VALUE (ITRO Y) turns tracing on

Other Tracing Values can be also altered
For example:
USE SET_SESSION_VALUE (ITRL '2')
USE SET_SESSION_VALUE (ITRC 'FUN')
USE SET_SESSION_VALUE (ITRM '999999999')

These are the most commonly used trace parameters

ITRO=

This parameter specifies whether the application is to produce a trace file. Specify Y to produce a trace file or N to not produce a trace file. Trace files are named X_TRACE.nnn. The highest nnn suffix indicates the newest trace file. The production of trace files severely impacts application performance.

ITRL=

This parameter specifies the level of trace. Valid values are 0 to 4, where 0 provides the lowest level of detail and 4 the highest level of detail. This should not be changed unless requested by your Product Vendor.

ITRM=

This parameter specifies the maximum number of lines in the trace file. The maximum number is 999,999,999.

ITRC=

This parameter specifies the trace categories. It allows you to restrict the areas of LANSA that will generate trace messages. This should not be changed unless requested by your Product Vendor. Use of this value is described in Microsoft Exception. Multiple values can be specified at a time as one string, e.g. DBMUIM.

ALL	All categories
DBM	Database only
UIM	User Interface only
FUN	Standard Function only
PIM	Printer functions only
COM	Communications only
PDF	Platform Dependent Functions only
BIF	Built-In Functions only
PRO	Reserved
RDM	RDML only
RDX	RDMLX only
HEP	Heap Validation only

V5R3 PTF 5722DG1-SI17010 causes HTTP/Server error 500 failures

Description

Application of V5R3 PTF 5722DG1-SI17010 can cause HTTP/500 server failures, thus affecting LANSA Web applications.

HTTPSVR APACHE CAUSES EBADF ERROR WHEN RUNNING CGI
Apache causes EBADF (errno 3450) when running CGI after applying
PTF SI17010. Sometimes MCH3601, HTP8081, CEE9901, and CPF24A3
are all listed in the joblogs.

Symptoms:

Error 500:

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>500 Internal Server Error</TITLE>
</HEAD><BODY>
<H1>Internal Server Error</H1>
Cannot read script output pipe.
</body></html>

and webserver traces will show:
[Tue May 17 08:16:58 2005] [error] [client 10.136.12.12] (3450)Descriptor not valid.: poll failed waiting for CGI child

Resolution:

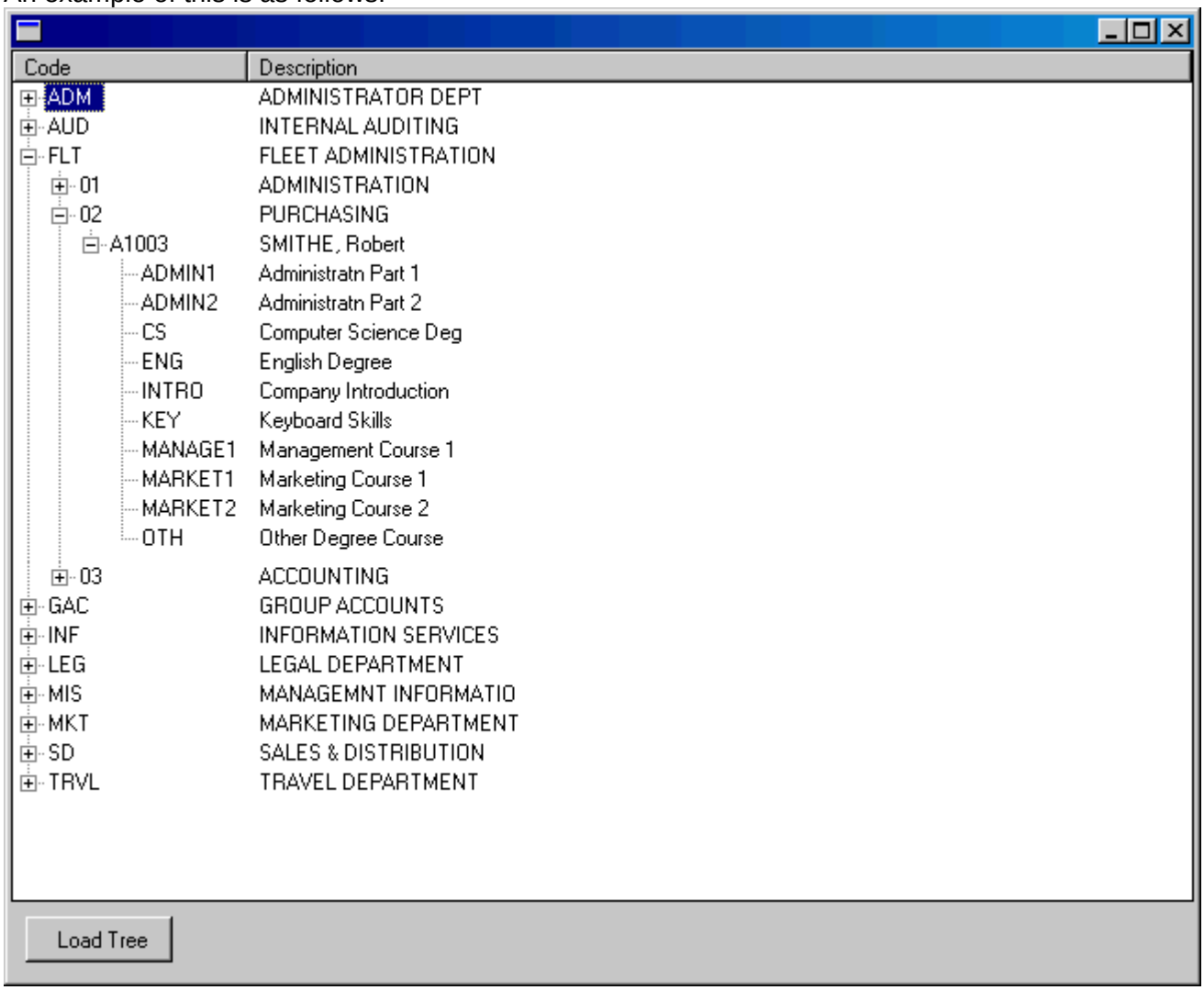
The recommendation from IBM is to apply PTF SI17696 - IBM HTTP Server for iSeries.
Refer to this link for details:
http://www-912.ibm.com/a_dir/as4ptf.nsf/0/8cb186dca1e0e06286256fd5004d0f70?OpenDocument

How to create a Columnar tree view (LANSA2005 only)

Description

Prior to LANSA2005 it was only possible to create a tree view with one visible column. This column became the main tree column. In LANSA2005 it is now possible to now create a Columnar tree, that is a tree with multiple columns above and beyond the main tree column.

An example of this is as follows:



In the above example you will notice how the tree view has two columns. The columnar tree will allow multiple columns. In order to create the above example perform the following in LANSA2005:

- Ensure that you have imported the LANSA demonstration objects.
- Create a form and ensure that the component or partition is enabled for full RDMLX.
- Cut and paste the code example listed below.
- Compile and run.

Note: The code is commented in places to assist understanding. Please refer to the LANSA guides for further clarity of new features/command syntax used in this example.

```
Function Options(*DIRECT)

Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(499) Clientwidth(644) Height(533)
Layoutmanager(#ATLM_1) Left(92) Top(130) Width(652)
Define_Com Class(#PRIM_ATLM) Name(#ATLM_1)
Define_Com Class(#PRIM_PANL) Name(#PANL_1) Displayposition(1) Height(42) Left(0)
Parent(#COM_OWNER) Tabposition(1) Tabstop(False) Top(457) Width(644)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_1) Attachment(Bottom) Manage(#PANL_1)
Parent(#ATLM_1)

* A new property called Viewstyle has been added to the PRIM_TRVW
* Unlevelled means that that the tree no longer creates a level per field, and instead, creates a
column per field.
* The developer is now reponsible for the level at which a tree item exists.
* This is governed by setting the PARENTITEM property of a tree item to another tree item

* NOTE - The tree can still be processed using SELECTLIST.
* Entries will be returned in the sequence they were added to the list. This may not reflect the
order as seen in the tree
* To process in order sequence, use the FOR command
Define_Com Class(#PRIM_TRVW) Name(#Personnel) Componentversion(1) Displayposition(2)
Height(457) Left(0) Parent(#COM_OWNER) Tabposition(2) Top(0) Viewstyle(UnLevelled)
Width(644)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_2) Attachment(Center) Manage(#Personnel)
Parent(#ATLM_1)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_1) Caption('Code') Captiontype(Caption)
Displayposition(1) Level(1) Parent(#Personnel) Source(#STD_OBJ)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_2) Caption('Description') Captiontype(Caption)
Displayposition(2) Level(2) Parent(#Personnel) Source(#STD_DESC) Widthtype(Remainder)


Define_Com Class(#PRIM_PHBN) Name(#pb_load) Caption('Load Tree') Displayposition(1)
Left(8) Parent(#PANL_1) Tabposition(1) Top(8)

Evtroutine Handling(#pb_load.Click)

Clr_List Named(#personnel)

#com_owner.Add_departments

Endroutine

Mthroutine Name(Add_departments) Access(*private)

Define_Com Class(#prim_tvit) Name(#Department_item) Reference(*dynamic)

Select Fields(#Deptment #deptdesc) From_File(Deptab)

#com_owner.Add_Entry I_Code(#deptment) I_Description(#deptdesc)
O_Tree_Item(#Department_item)

* Add the sections for the department
#com_owner.Add_sections I_Parent_Item(#department_item) I_Department(#deptment)

Endselect
```

```
Endroutine

Mthroutine Name(Add_sections) Access(*private)
Define_Map For(*input) Class(#prim_tvit) Name(#i_parent_item) Pass(*by_reference)
Define_Map For(*input) Class(#deptment) Name(#i_department)

Define_Com Class(#prim_tvit) Name(#Section_item) Reference(*dynamic)

Select Fields(#section #secdesc) From_File(sectab) With_Key(#i_department)

#com_owner.Add_Entry I_Code(#section) I_Description(#secdesc)
O_Tree_Item(#Section_item) I_Parent_Item(#i_parent_item)

* Add the Employees for the section
#com_owner.Add_Employees I_Parent_Item(#Section_item) I_Department(#deptment)
I_Section(#Section)

Endselect

* Set a margin on the last item to help separate the groups of tree items
#personnel.Currentitem.marginbottom := 5

Endroutine

Mthroutine Name(Add_Employees) Access(*private)
Define_Map For(*input) Class(#prim_tvit) Name(#i_parent_item) Pass(*by_reference)
Define_Map For(*input) Class(#deptment) Name(#i_Department)
Define_Map For(*input) Class(#section) Name(#i_Section)

Define_Com Class(#prim_tvit) Name(#Employee_item) Reference(*dynamic)

Select Fields(#empno #givename #surname) From_File(pslmst1) With_Key(#i_Department
#i_section)

#com_owner.Add_Entry I_Code(#empno) I_Description(#Surname.trim + ', ' + #Givename)
I_Parent_Item(#i_parent_item) O_Tree_Item(#employee_item)

* Add the skills for the employee
#Com_Owner.Add_Skills I_Parent_Item(#employee_item) I_Employee(#empno)

Endselect

* Set a margin on the last item to help separate the groups of tree items
#personnel.Currentitem.marginbottom := 5

Endroutine

Mthroutine Name(Add_Skills) Access(*private)
Define_Map For(*input) Class(#prim_tvit) Name(#i_parent_item) Pass(*by_reference)
Define_Map For(*input) Class(#empno) Name(#i_Employee)

Select Fields(#skilcode) From_File(pslskl) With_Key(#i_Employee)

Fetch Fields(#skildesc) From_File(skltab) With_Key(#skilcode)

#com_owner.Add_Entry I_Code(#skilcode) I_Description(#skildesc)
I_Parent_Item(#i_parent_item)

Endselect
```

```
* Set a margin on the last item to help separate the groups of tree items
#personnel.Currentitem.marginbottom := 5

Endroutine

Mthroutine Name(Add_entry) Access(*private)
Define_Map For(*input) Class(#std_obj) Name(#i_code)
Define_Map For(*input) Class(#std_Desc) Name(#i_Description)
Define_Map For(*input) Class(#prim_tvit) Name(#i_Parent_item) Mandatory(*null)
Pass(*by_reference)
Define_Map For(*output) Class(#prim_tvit) Name(#o_Tree_item) Mandatory(*null)
Pass(*by_reference)

* The same fields are used regardless of the "level" of the tree item
#std_obj := #i_code
#std_desc := #i_Description

Add_Entry To_List(#Personnel)

* Set the new tree item's parent to the supplied parent. A null parent means that the item will
appear as a root node
* The parent is no longer governed by the data. It is a choice the developer can now make.
* Effectively, there is NO LIMIT to the number of levels.
* Note: The parent of a tree item is completely dynamic. It can be set at any time
Set Com(#Personnel.currentitem) Parentitem(#i_parent_item)

* Return the tree item for use as a parent
Set_Ref Com(#o_tree_item) To(#Personnel.currentitem)

Endroutine

End_Com
```

Using Microsoft C++ 6.0 compiler with Visual LANSA2005

Description

In some cases, after installing Visual LANSA 11.0 on a system that already has Microsoft Visual C++ 6.0 installed, LANSA components may not compile successfully.

This can occur if Microsoft Visual C++ 6.0 was not setup to enable the build tools to run from the command line (the necessary environment variables were not created or modified).
If this is the case, you need to manually establish the environment variables, so that Visual LANSA can locate and correctly execute the Microsoft compiler.

To do this, first locate the batch file vcvars32.bat, usually found in the following location:
C:\Program Files\Microsoft Visual Studio\VC98\Bin

Open vcvars32.bat using a text editor such as notepad.exe and locate the lines that set the PATH, INCLUDE and LIB environment variables. You will need their values below – do not modify this file.

Now choose Start -> Settings -> Control Panel -> System -> and select the Advanced tab, now click the Environment Variables button. Modify or create the PATH, INCLUDE or LIB system environment variables to match the values you obtained from vcvars32.bat. (To do this, you might need to create or expand some of the embedded system variables, such as %MSDevDir%).

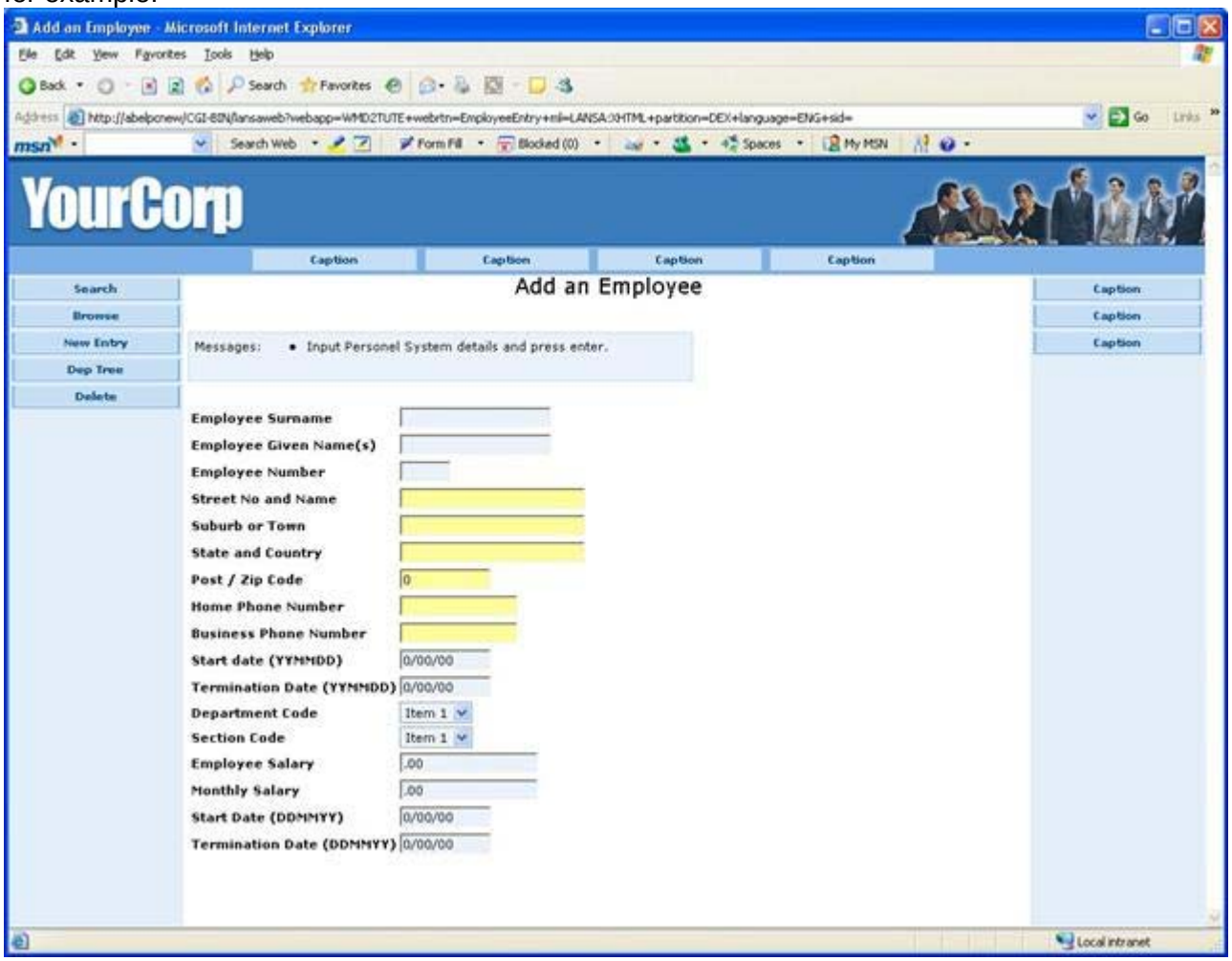
Note: Microsoft support for Visual C++ 6.0 is scheduled to end on 30 September, 2005. After this date, Microsoft Visual C++ 6.0 will not be a supported compiler for Visual LANSA. Users should plan to upgrade to the latest Microsoft Visual C++ compiler for this date.

LANSA for the web input fields have a pale yellow background

Description

When testing or running a Web application some input fields may have a canary-yellow background regardless of the background color or style selected,

for example:



This may be caused by one of the toolbars installed on your browser. Google Toolbar, MSN Desktop Search toolbar and similar tools have a form Auto-Fill feature which can automatically fill out forms for you. These tools highlight the form fields they can automatically fill. This is not a LANSA for the Web (or WAM) issue.

By disabling the Auto-Fill feature the fields will no longer be highlighted.

First time performance penalty when JVM Loads

Description

The first time penalty happens when the Java Virtual Machine loads the class into memory for the first time. If a class has not been used for 20 minutes the JVM unloads the class from memory. So when the service is used again at a later date, the class has to be reloaded (load penalty again).

To overcome the first time penalty, a custom Startup class has been written to create a thread that runs every 20 minutes to load the classes and keep them in memory.

Note: You do not need to just load the classes, you could execute a dummy XML transform or dummy SOAP transaction, etc.

You could also try the *NOCLASSGC option on the STRJSM command.
STRJSM OPTION(*NOCLASSGC)
-noclassgc Turns off garbage collection of Java classes. By default, the Java interpreter reclaims space for unused Java classes during garbage collection.

http://amath.colorado.edu/computing/software/man/java.html

Note: If you are going to use the sample below, remember to remove the System.out.println lines. They are only there for debug and to illustrate the logic.

Sample Startup Class

```
package com.acme ;

public final class ActiveStartup implements Runnable
{
private int m_sleepTime = 0 ;

public ActiveStartup ()
{
/*
JSMManager uses the zero argument constructor
*/

int seconds = 60 * 20 ; // Every 20 minutes

Thread thread = new Thread ( new ActiveStartup ( seconds ) ) ;

thread.start () ;
}

public ActiveStartup ( int seconds )
{
/*
Specify sleep time
*/

if ( seconds <= 0 )
{
seconds = 0 ;
```

```
}

m_sleepTime = seconds * 1000 ;
}

public void run ()
{

if ( m_sleepTime == 0 )
{
/*
JSMManager call
*/

System.out.println ( "JSM warmup call" ) ;

try
{
warmup () ;
}
catch ( Exception e )
{
e.printStackTrace () ;
}

return ;
}

/*
ActiveStartup call with sleep time
*/

while ( true )
{
try
{
Thread.sleep ( m_sleepTime ) ;

System.out.println ( "ActiveStartup repeat warmup call" ) ;

warmup () ;
}
catch ( Exception e )
{
e.printStackTrace () ;
}
}

}

private final void warmup () throws Exception
{
System.out.println ( "ActiveStartup: warmup" ) ;

}
}
```

Identify which XSLT Processor is being used on iSeries for WAMs

Description

There may be a need or it may be useful to be able to identify which XSLT processor is being used in your WAM applications. To determine this, in your XSLT stylesheet, add the following:

```
<table>
<tr>
<td>XSLT Processor:</td>
<td><xsl:value-of select="system-property('xsl:vendor')" /></td>
</tr>
</table>
```

iSeries:

On the iSeries, if you have a joblog of the LWEB_JOB, to see which XSLT processor ran, refer to the message for the PASE startup:

1. ILE XSL4C = No message
2. Apache Xalan = PASE started (no bit mode details in 2nd level message). This applies to V11.0 GA.
3. libxslt = PASE started in 32-bit mode
4. Apache Xalan = PASE started in 64-bit mode

DEF_LIST - Dynamic vs Static for RDMLX components to avoid memory errors

Description

Since LANSA2005, there are two types of working list. The first list is one that specifies *MAX for the ENTRYS parameter. This kind of list dynamically allocates memory and is referred to as a Dynamic Working List. The second specifies any other value for the ENTRYS parameter. These are static working lists.
The *recommended* kind of list to use in an RDMLX object is *DYNAMIC* as the list is only limited by the available resources.

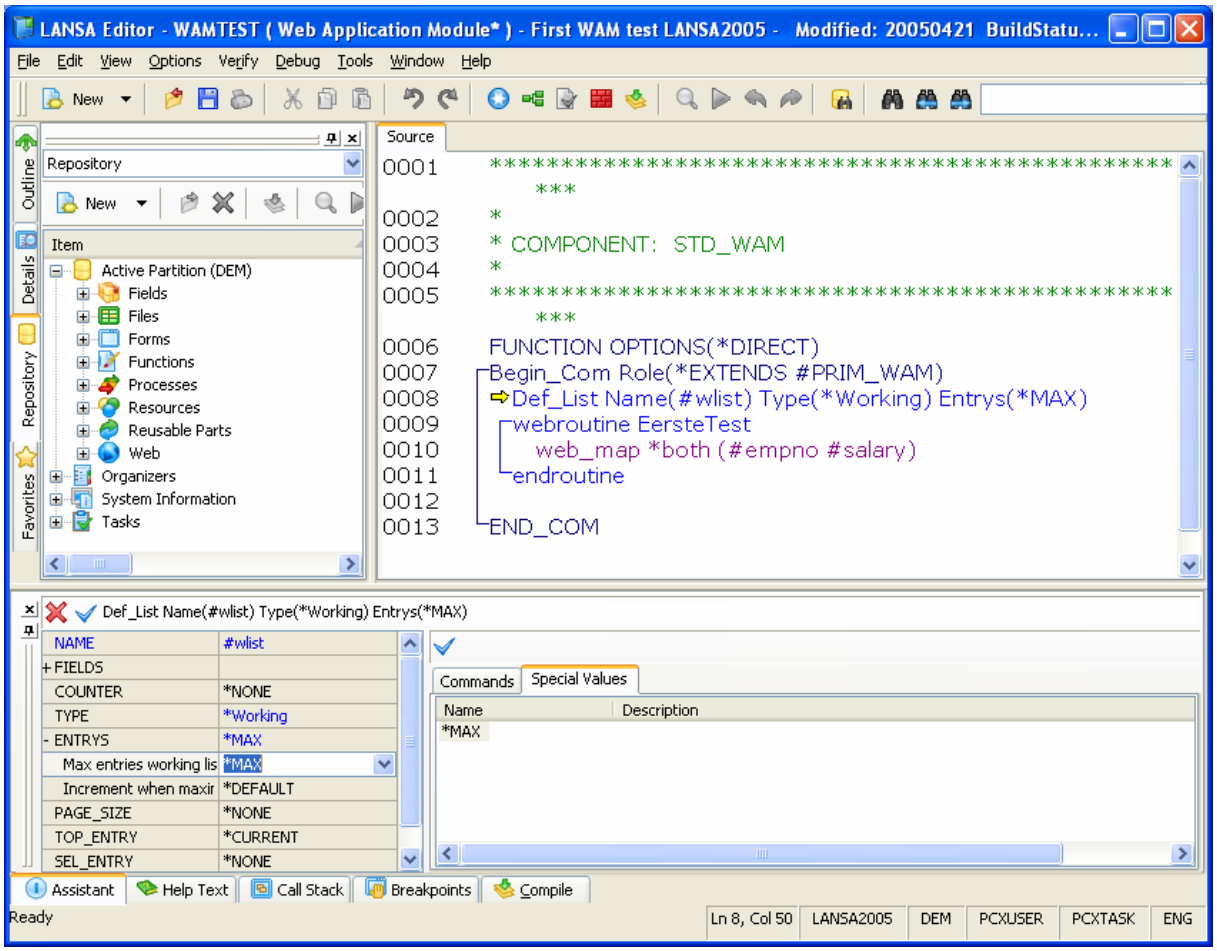
Dynamic - Specifies *MAX for the ENTRYS parameter

A dynamic working list in an RDMLX object allocates and releases memory on demand and only pre-allocates a small amount of memory to hold pointers to the list entrys. Then, as more space is required it is allocated with one page of operating system memory or the size of one entry, which ever is the larger. On Microsoft Windows the size of a page is 32 Kbytes. Memory is also released as entrys are deleted from the list. If you were to keep adding entries indefinitely, memory would eventually run out on windows.

This is the recommended kind of list to use in an RDMLX object, though it has severe restrictions when used with the SORT_LIST command. The aggregate entry length cannot exceed 2 Giga bytes in a primary list. Also, String and Binary data memory needs are on top of this as they are not stored in the list itself. Thus each entry could have many Strings each up to 64 Kbytes long. Thus it is very easy to consume very large amounts of memory.

Static - Specifies any other value for the ENTRYS parameter

The list will contain up to the number of entries specified but which has a maximum of 2 giga entries. The aggregate entry length cannot exceed 2 Giga bytes in a primary list. Also, String and Binary data memory needs are on top of this as they are not stored in the list itself. Thus each entry could have many Strings each up to 64 Kbytes long. Thus it is very easy to consume very large amounts of memory.



WARNING - keep in mind when defining static working lists that the amount of storage allocated to each working list will be equal to the entry length multiplied by the number of entries on the list, so the amount of storage space allocated to a function using many working lists can increase quite substantially.

When a static working list will exceed typical available Windows process memory, messages 870, 871, 872, 873 and 874 may be displayed. An example of message 871 is:
"Maximum 32-bit windows server process memory of 3 GB will be exceeded."

These messages should be considered as near fatal errors. They are indicating that the memory requirement is beyond the capability of particular windows configurations. The capabilities of other platforms is generally larger, like the iSeries, but to raise these warnings still indicates a design that should be re-considered.

Message 874 contains the dimensions of the list. An example of message 874 is:
"List page size = 1098000000 bytes Entry length = 549 bytes."

NOTE: Whilst the iSeries has a greater total amount of memory available it is limited to a maximum of 16MB in any single memory allocation. This means that on iSeries each STATIC working list is limited to a TOTAL size of 16MB and each DYNAMIC working lists is limited to a maximum ENTRY width of 16MB. Thus a dynamic working list has a far greater total capacity - only limited by the total amount of memory that the operating system has available for the process to use. This is strictly just an iSeries limitation. All other platforms have the same limit for each list as for the total memory used by all lists.

Incorrect translation of certain characters in LANSA for the Web generated HTML

Description

When exporting or during checkin/checkout of generated HTML in Visual LANSA, a problem MAY occur within the EBCDIC/ASCII translation.

Symptoms

Appearance of incorrect characters appearing on web pages which should be hidden. Examples of characters which can be incorrectly translated are: "!', "[", "]", "^", and "|". An example is comments. This is caused by the incorrect translation of the "!" character, which denotes a comment in the HTML.

In LANSA for iSeries, the definitions required to run the Host Monitor and to export using the PC export type can be created using the LANSA REQUEST(PCMAINT), or can be created automatically by performing a PLUGIN/REFRESH from Visual LANSA.
The names of the translation tables used to convert data to and from EBCDIC and ASCII are included in these definitions. These tables are only used by export to PC and the Host Monitor and NOT by LANSA Open or SuperServer which have their own tables.

Resolution

Two sample iSeries translation tables have been provided to correct this problem. These tables are ONLY suitable for English language users with code page 037. If you are already using modified translation tables and have the comment (!) problem for example, you would need to change the entry for "5A" to translate to "21" in the EBCDIC to ASCII table and entry "21" to translate to "5A" in the ASCII to EBCDIC table for the correct translation of the exclamation(!) character.

Download the two sample English tables - L4WTTBL

We recommend restoring these tables to QGPL, e.g.:
*RSTOBJ OBJ(L4WEBCDIC L4WASCII) SAVLIB(QGPL) DEV(OPT01) OBJTYPE(*TBL) RSTLIB(QGPL)*

When exporting from LANSA for iSeries to Visual LANSA the export program looks first in PCMAINT for a PC definition called WIN_DFT. If it does not find it, it uses default table names of QASCII and QEBCDIC which are IBM® supplied iSeries tables.

You can modify existing PC definitions in PCMAINT to use the sample tables for the Host Monitor and/or modify/create the WIN_DFT definition for exporting. Table L4WASCII performs the EBCDIC to ASCII translation and L4WEBCDIC the ASCII to EBCDIC translation.
For help in creating the PC definition refer to the LANSA for iSeries User Guide, section LANSA PC Development, Defining Personal Computers to LANSA, Creating (Changing) PC Definitions.

Non-English

If you are using a language other than English and a different code page you will have to add the appropriate characters to handle the correct translation of these characters in error.

Example Steps

Steps to fix the translation tables for the following characters: ![]^|
The tables are changed using the following OS/400 commands:
WRKTBL QEBCDIC
WRKTBL QASCII

Use the Work with Table Dialogue to copy the tables (i.e. save them), then convert the tables to text files using the RTVTBLSRC command , e.g.:
RTVTBLSRC TBL(QEBCDIC)
SRCFILE(QGPL/QCLSRC)

RTVTBLSRC TBL(QASCII)
SRCFILE(QGPL/QCLSRC)

Now use SEU to edit both source file members created by RTVTBLSRC as per the following tables:

Changes to table QSYS/QEBCDIC

Char	Source File	Original Value	New Value
!	L02P03	4F	5A
[	L03P55	4A	BA
]	L03P59	5A	BB
^	L03P61	5F	B0
|	L04P57	6A	4F

Changes to table QSYS/QASCII

Char	Source File	Original Value	New Value
!	L03P53	5D	21
[	L06P53	E2	5B
]	L06P55	E3	5D
^	L06P33	D8	5E
|	L03P31	21	7C

Example

In QASCII , change Line(3),position(53),length(2) to 21
In QEBCDIC , change Line(2),position(3),length(2) to 5A

Create the tables from the amended text files with the CRTTBL command, e.g.:
CRTTBL TBL(QGPL/L4WEBCDIC) SRCFILE(QGPL/QCLSRC)
CRTTBL TBL(QGPL/L4WASCII) SRCFILE(QGPL/QCLSRC)

Note: we do not recommend overwriting the IBM supplied default tables. If you do wish to override these, then you would use.
CRTTBL TBL(QSYS/QEBCDIC) SRCFILE(QGPL/QCLSRC)
CRTTBL TBL(QSYS/QASCII) SRCFILE(QGPL/QCLSRC)

Invitation for IBM event 5 October

We like to invite you for the ***"innovative solutions for iSeries"*** event at 5 October at IBM Brussels.

Information and registration:
http://www-5.ibm.com/be/events/isi/

During this event you can find us at stand 3.

We will show our new LANSA2005 version and we will also give a presentation in the Rubens room from 10.00 to 10.30.

We will show how easy and quickly it is in LANSA2005 to develop your applications. The types of applications during this demonstration are:

- Windows Applications
- Web Applications for a browser and PDA
- Web Services (client & server site) and integration with 5250 software

We hope to see you at 5 October!.