http://www.lansa.com

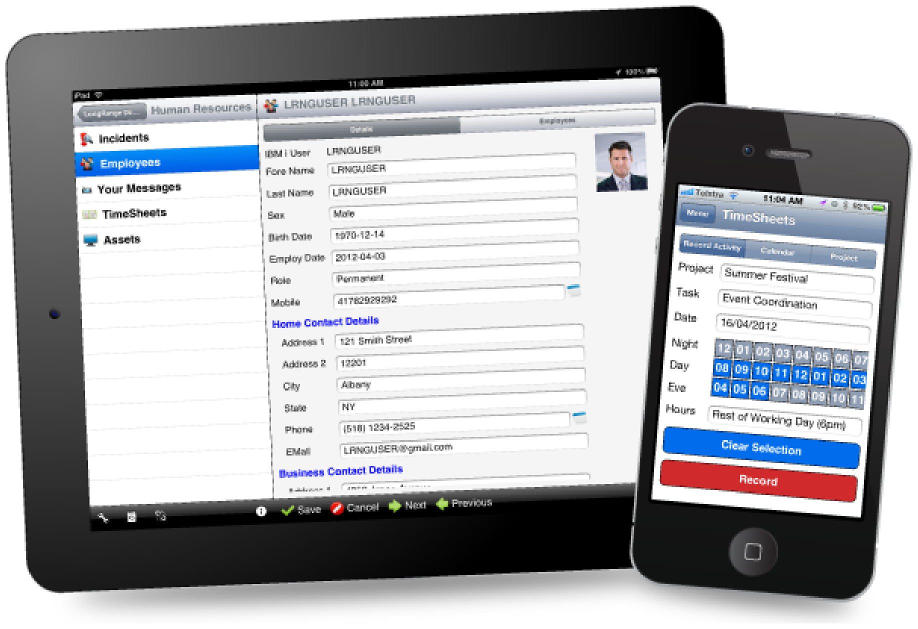
LANSA

Newsletter
August 2012

LongRange for LANSA

Native mobile app builder for LANSA developers.

Build and maintain native mobile apps for Apple iOS and Android using LANSA development tools and methods. LongRange does not require any programming on the mobile device to use features such as photos, videos, audio recordings, documents, maps and geo-location in LANSA applications. Developers don't have to learn any new programming languages such as Objective C (Apple), Java (Android) or other coding techniques like HTML, CSS and JavaScript.



IBM i User LRNGUSER
Fore Name LRNGUSER
Last Name LRNGUSER
Sex Male
Birth Date 1970-12-14
Employ Date 2012-04-03
Role Permanent
Mobile 41782929292

Home Contact Details
Address 1 121 Smith Street
Address 2 12201
City Albany
State NY
Phone (518) 1234-2525
EMail LRNGUSER@gmail.com

Business Contact Details

Project Summer Festival
Task Event Coordination
Date 16/04/2012
Night 12 01 02 03 04 05 06 07
Day 08 09 10 11 12 01 02 03
Eve 04 05 06 07 08 09 10 11
Hours Rest of Working Day (6pm)

In This Issue

Timely and cost effective mobile application development

LongRange encourages results oriented programming as it allows developers to concentrate on business requirements and not mobile device technology. This means that the total-cost-of-ownership for a mobile app becomes a viable proposition, especially for companies without any previous experience in developing apps for mobile devices.

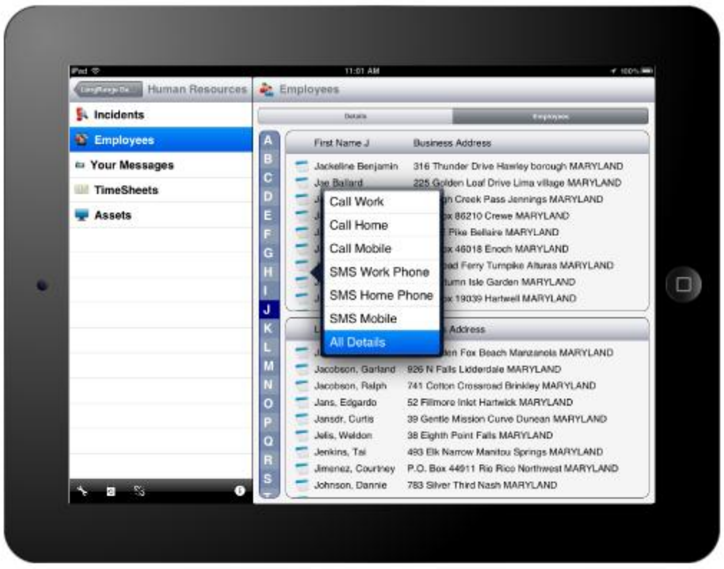
No new skills required

You only need LANSA and you're already an expert with this!

Short time-to-market

Significantly reduce the time to design, program and deploy native apps for multiple mobile platforms:

- Use LANSA on the server; no programming is required for the mobile device.
- The same programs will service both iOS and Android devices.
- The user interface of the LongRange native mobile app automatically reshapes and resizes to accommodate different screen sizes (e.g. smart phones versus tablets).



First Name J	Business Address
Jackeline Benjamin	316 Thunder Drive Hawley borough MARYLAND
Joe Ballard	225 Golden Loaf Drive Lima village MARYLAND
J	n Creek Pass Jennings MARYLAND
J	x 86210 Crewe MARYLAND
J	Pike Bellaire MARYLAND
J	x 46018 Enoch MARYLAND
J	ad Ferry Turnpike Alturas MARYLAND
J	umn Isle Garden MARYLAND
J	x 19039 Hartwell MARYLAND

L	Address
J	en Fox Beach Manzanola MARYLAND
Jacobson, Garland	926 N Falls Lidderdale MARYLAND
Jacobson, Ralph	741 Cotton Crossroad Brinkley MARYLAND
Jans, Edgardo	52 Fillmore Inlet Hartwick MARYLAND
Jansdr, Curtis	39 Gentle Mission Curve Duncan MARYLAND
Jelis, Weldon	38 Eighth Point Falls MARYLAND
Jenkins, Tai	493 Elk Narrow Manitou Springs MARYLAND
Jimenez, Courtney	P.O. Box 44911 Rio Rico Northwest MARYLAND
Johnson, Dannie	783 Silver Third Nash MARYLAND

Low total-cost-of-ownership

Maintaining and extending native apps using familiar development skills will be as easy in the future as it is today:

- There is no additional cost of hiring and retaining developers with specialized mobile app development skills.
- You only need to maintain one set of LANSA source code to support multiple mobile platforms.

Rapid return on investment
Quickly deliver easy-to-use native mobile apps with a rich user interface and enhance your users' productivity.

- Reduced on-going maintenance costs – same source code for both iOS and Android.
- Minimal developer learning curve – no need to learn mobile specific programming.

Using LongRange, companies can develop and deploy apps for mobile devices quickly. The time-to-market is short as development is incremental and evolutionary.

Speed up mobile app development

The LongRange mobile app is a working native app and provides a ready-made infrastructure into which developers can add the programs of a business application. LongRange allows developers to start developing the business logic and database without having to develop the services the LongRange mobile app provides (e.g. touch enabled user interface).

Developers are productive immediately because they can use the same programming model they already use and are well equipped to trace and debug intuitively.

Use LANSA development tools

Developers can use LANSA to build programs for mobile devices without learning new programming languages. There is no need to learn Java, Objective-C or brush up on HTML, CSS and JavaScript skills.

Developing a mobile app using LongRange is a three-step process:

1. Step one defines the navigation, menus, forms and views using LongRange Studio. This definition (application schema) tells the LongRange mobile app how you want the business application to operate and you can run this as a prototype to test your design.
2. Step two allows you to use your LANSA skills to build the business logic, screen layout and data content.
3. Step three is where you add the programs into LongRange by telling the schema what program to call for each screen in the business application.

Use your existing assets

LongRange has been designed to use your existing investments, infrastructure and developer skills and does not replace or change anything already in place or in use.

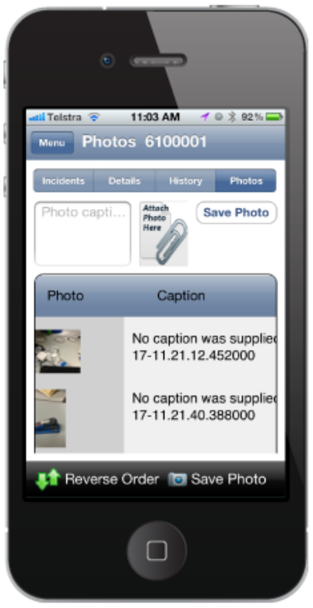
LongRange works in your existing environment and requires:

- No large upfront investments.
- No new OEM software.
- No new hardware.
- No operating system upgrades.

LongRange runs on your existing LANSA environment.

LongRange Features

- LongRange native mobile app
- LongRange server
- LongRange developer studio
- Build native mobile apps
- Use familiar LANSA development tools to build mobile business apps
- Automatically generate mobile app screens
- LANSA programs can use the capabilities of mobile devices, including the camera and geo-location
- Apps can send and receive files between a mobile device and a server
- App templates and extensive code examples
- Multilingual and DBCS language support
- Use existing server infrastructure
- Uses strong security
- TLS/SSL encryption



Requirements

- LongRange requires IBM i V5R4 or later or Windows Server 2008 SP2 or later (coming soon)
- LANSA V12 SP1 or later
- LANSA for the Web
- Document views for IBM i require the installation of LongReach
- Document views for Windows (coming soon).

Did You Know...

You can trial fully functioning versions of LongRange Studio and LongRange for LANSA FREE for 30-days BEFORE you need to license the software.

Visual LANSA Install can fail due to incorrect Native Client driver

Installing Visual LANSA V12 SP1 on a 64-bit O/S can fail in the following situation:

- Windows 64-bit Operating system
- no SQL Server Native Client driver installed
- Installing Visual LANSA V12 SP1 and selecting one of the following install types
    - Network Client install (Shortcuts to run Visual LANSA installed on another PC)
    - Client to a Slave Database Server
    - Client to an Independent Database Server

The install error is:
ODBC driver <SQL Server Native Client 10.0> for Data Source <LXLANSA> does not exist

The Visual LANSA install establishes that the SQL Server Native driver is not available and attempts to install it. However, the 32-bit version of the Native Client driver sqlncli.msi is incorrectly installed instead of the 64-bit version of the Native Client driver. This installation fails and the MSSQLNavCli_Install.log contains the message:
Microsoft SQL Server 2008 Native Client – Installation of this product failed because it is not supported on this operating system.

Solution

The solution is to manually install the 64-bit version of the Native Client driver **sqlncli_X64.msi** from the following location on the V12 SP1 DVD

```
setup\Installs\MSSQLNAVCLI\sqlncli_X64.msi
```

The Visual LANSA V12 SP1 install can then be rerun.

Compilation errors on RDMLX forms after upgrading to V12 SP1

An issue has been found where certain forms created prior to V12 SP1 will fail to compile after an upgrade to V12 SP1 or after being imported into a V12 SP1 system.

<u>The situations where this can occur are:</u>

- RDMLX Partition
- RDMLX Form or Function
- Form/function contains a SUBROUTINE
- The form/function contains no full RDMLX statements

The build process will complete without errors; however the compilation ends with the following error in the compilation log:

```
c:\<partition source directory>\<form/func name>.C(1773) : error C2065:
'lReturnCode' : undeclared identifier
```

Solution

A simple workaround can be used to get the form to compile by adding a full RDMLX command or converting one of the existing commands to full RDMLX. A common candidate is the #com_owner.Initialize EVTROUTINE, which contains a SET command.

Changing this line of code does the exact same thing, but using RDMLX:
#com_owner.caption := *component_desc

Alternatively, contact your local LANSA vendor to request a hotfix to correct the issue.

Notice of support for 64-bit editions of Microsoft Office

Starting with Office 2010, Microsoft Office will be available in both 32-bit and 64-bit versions. By default, even on a 64-bit edition of Windows, the 32-bit version of Microsoft Office will be installed.

As outlined in the following page, the 32-bit version of Office is recommended for most people to prevent compatibility issues with other 32-bit applications:

http://go.microsoft.com/fwlink/?LinkID=245825

LANSA runs as a 32-bit application, even in 64-bit editions of Windows, so if you select to install the 64-bit version of Microsoft Office, LANSA will not be able to interact with Office ActiveX controls, or access Outlook using MAPI (for example using the MAIL_xxxxx Built-in Functions).

The symptoms will appear as if Office is not installed on the PC (such as a message stating no Mail application is installed), even if Office is installed correctly.

Summary

If your LANSA Application interacts with Microsoft Office through ActiveX, COM, MAPI or using the LANSA MAIL_xxxxx BIFs, then you should advise your users of incompatibility with the 64-bit version of Microsoft Office.

Where possible, encourage users to use the default 32-bit installation of Office, and put safeguards in your application to ensure that no fatal errors occur when the Office ActiveX interfaces cannot be found.

Framework lock is released after period of inactivity

An issue has been found where framework locking (#avFrameworkManager.uLocked) gets automatically reset after a certain period of inactivity. This typically occurs when the user leaves the framework screen to work in another application, and then returns after some time. This has been determined to be an issue related to the autosave feature that is used in the Design Mode of the Framework.

Summary

A fix for this issue will be included in the next VLF build, however as a workaround if framework locks seem to be released automatically after a number of minutes, check the framework's autosave interval and set it to zero.

Some tips for diagnosing unresponsive browser issues in VLF Web

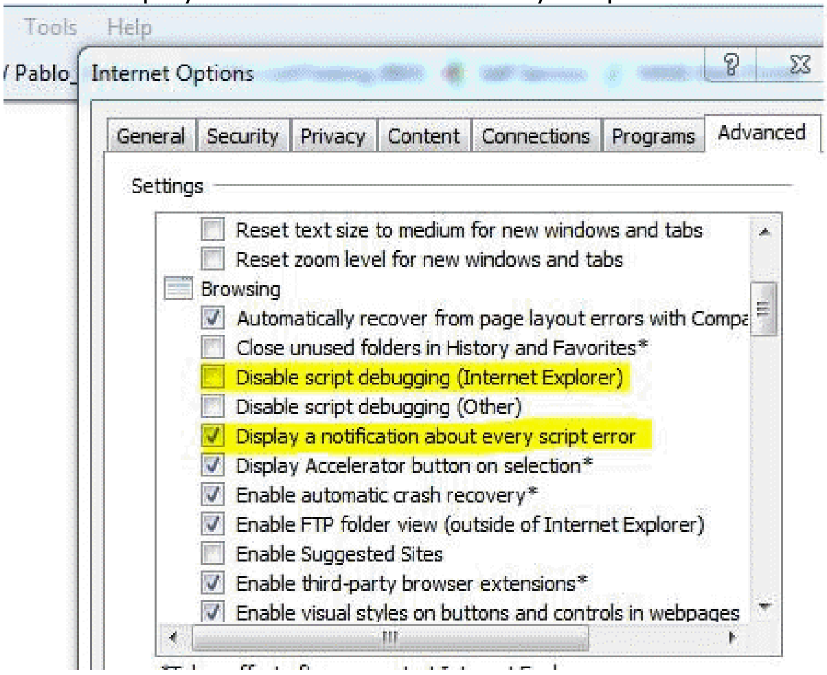
Below are some tips when diagnosing issues on VLF Web, particularly issues where the browser is unresponsive. The steps to follow are different, depending on the browser.

To debug browser problems

Internet Explorer

Under the Tools menu, Internet Options, Advanced tab:

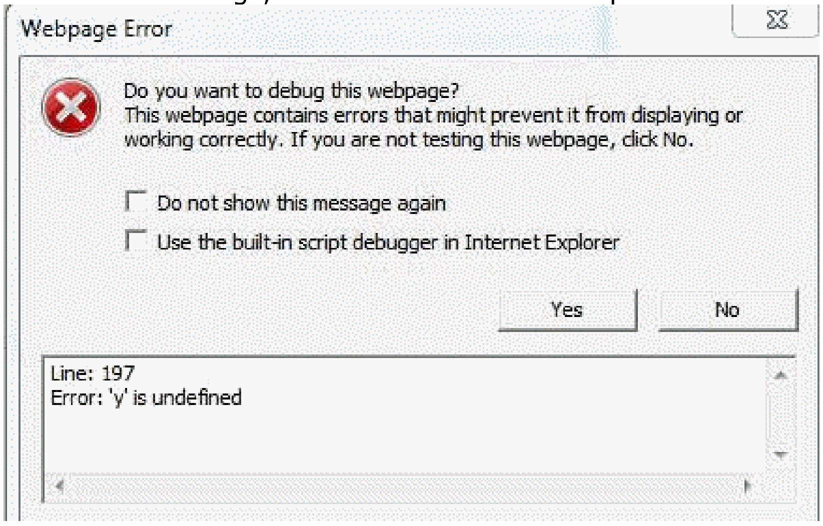
- Disable script debugging is **unchecked**
- Display a notification about every script error is **checked**



- [ ] Reset text size to medium for new windows and tabs
- [ ] Reset zoom level for new windows and tabs

Browsing
- [✓] Automatically recover from page layout errors with Compa
- [ ] Close unused folders in History and Favorites*
- [ ] Disable script debugging (Internet Explorer)
- [ ] Disable script debugging (Other)
- [✓] Display a notification about every script error
- [✓] Display Accelerator button on selection*
- [✓] Enable automatic crash recovery*
- [✓] Enable FTP folder view (outside of Internet Explorer)
- [ ] Enable Suggested Sites
- [✓] Enable third-party browser extensions*
- [✓] Enable visual styles on buttons and controls in webpages

With those settings, when there is a JavaScript error a window like this should appear:

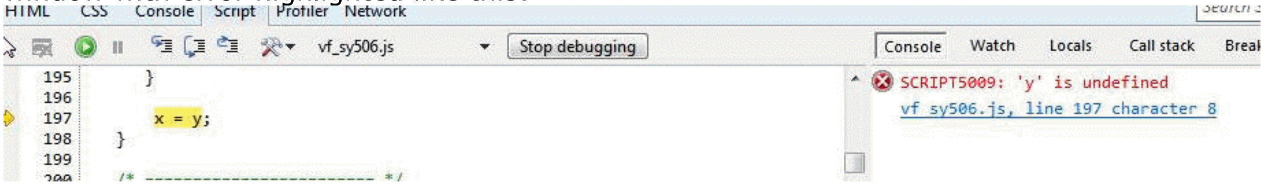


Do you want to debug this webpage?
This webpage contains errors that might prevent it from displaying or working correctly. If you are not testing this webpage, click No.

- [ ] Do not show this message again
- [ ] Use the built-in script debugger in Internet Explorer

```
Line: 197
Error: 'y' is undefined
```

Select the box *{Use the built-in script debugger in Internet Explorer}* which will display a window with error highlighted like this:



```
195        }
196
197            x = y;
198        }
199
200        /* ------------------------ */
```

```
SCRIPT5009: 'y' is undefined
vf_sy506.js, line 197 character 8
```

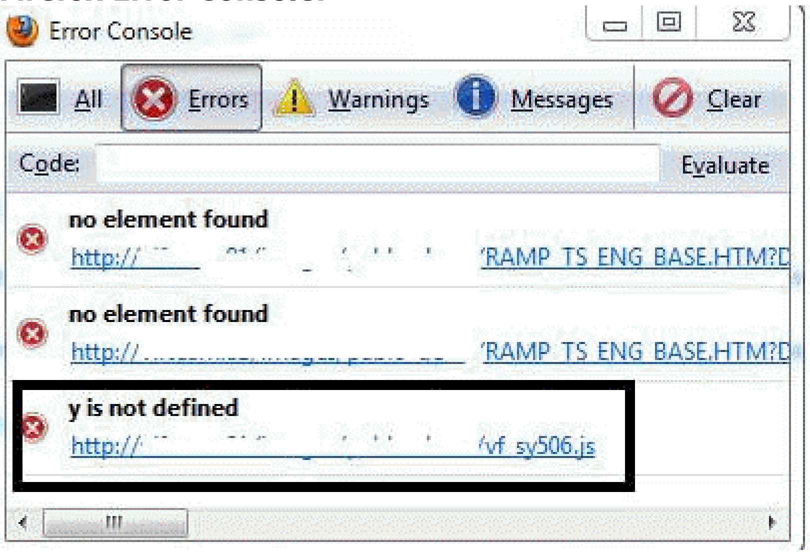
Firefox, Chrome and Safari

Unlike IE, the errors will not show but they will be recorded into their respective web consoles.

When the browser seems not to be responsive it is probably because some error has occurred.

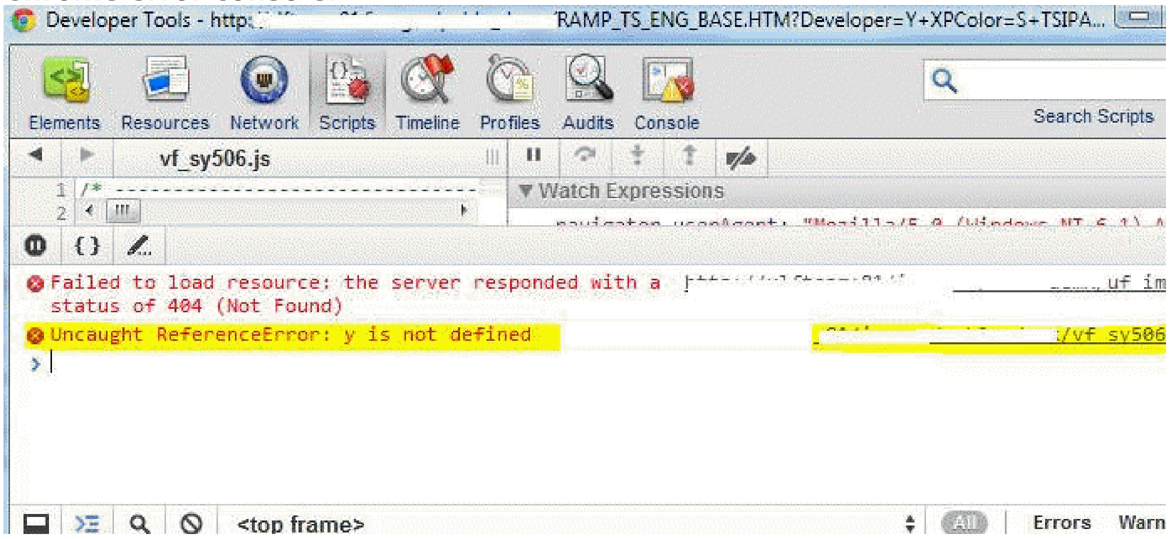
Firefox and Chrome: press Ctrl+Shift+J

Firefox Error Console:



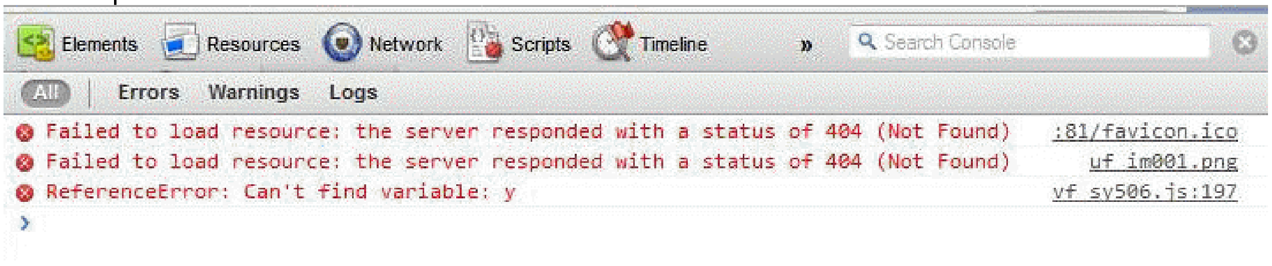
```
no element found
http:// [illegible] RAMP_TS_ENG_BASE.HTM?D

no element found
http:// [illegible] RAMP_TS_ENG_BASE.HTM?D

y is not defined
http:// [illegible] /vf_sy506.js
```

Chrome error console:



```
Failed to load resource: the server responded with a     http:// [illegible] uf_im
status of 404 (Not Found)
Uncaught ReferenceError: y is not defined                 [illegible] /vf_sy506
```

Safari: press Ctrl+Alt+C



```
Failed to load resource: the server responded with a status of 404 (Not Found)    :81/favicon.ico
Failed to load resource: the server responded with a status of 404 (Not Found)       uf_im001.png
ReferenceError: Can't find variable: y                                            vf_sy506.js:197
```

Depending on which browser you are using, follow the settings listed above to retrieve the error information and send it to LANSA Support. This information will be very useful for technical support purposes.

Note

While you may be able to use the debugging to determine why the browser has become unresponsive or has generated an error, the main reason for enabling the script debugging and retrieving the error is so that the details can be sent to LANSA Support for diagnosing why the error is being generated.

Receiving chunked transfer encoded content from a HTTP client

If you have a HTTP client sending you 'chunked transfer' encoded content, the following information details how to receive it on an IBM i server using Apache server.

You can configure a Proxy instance to 'dechunk' the content and send it to another Apache instance as content and Content-Length property.

The directive to use is:
SetEnv proxy-sendcl 1

Refer to http://httpd.apache.org/docs/current/mod/mod_proxy.html for detailed explanation and examples.

Sample configuration on the Proxy (Web Application) server:

```
#
# Apache Configuration
#
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
#
Listen nn.nn.nn.nn:nnnn
#
ProxyReverse On
ProxyRequests Off
#
ProxyTimeout 3000
<Proxy *>
Order Allow,Deny
Allow from all
</Proxy>
#
SetEnv proxy-sendcl 1
ProxyPass /cgi-bin/jsmdirect http://myserver:nnnn/cgi-bin/jsmdirect
```

LANSA shipped objects flagged as requiring conversion by Analyze Object Conversion tool

For details of the conversion which occurs when migrating to 6.1, refer to Details of support for IBM i 6.1 (http://www.lansa.com/support/tips/t0453.htm). The tip refers to 6.1 but also applies to 7.1.

IBM has provided the Analyze Object Conversion (ANZOBJCVN) tool for IBM i 5.3 and 5.4 (i5/OS V5R3 and V5R4) to help you plan for program conversion. This tool helps you identify potential conversion difficulties, if any, and estimates the time required for program conversion.

You should ensure you are on a supported version of LANSA (current LANSA supported versions are deemed 6.1 Ready)when running the conversion tool to avoid large numbers of LANSA objects being flagged as requiring conversion. For LANSA customers that are on supported versions of LANSA, not only are the number of objects greatly reduced but the remaining objects generally fall into specific categories. The following item lists might help you to determine whether the objects are required in your environment or not.

- **P@D*** - *PGM objects located in DC@DEMOLIB. These are LANSA demo programs (PSLSYS) no longer shipped and are candidates for deletion.
- **@LM*** - *PGM objects are Data Modeler objects. See Note 1
- **P@*** - objects in MODLIB are also Data Modeler objects. See Note 1
- **M@SYS*** - system variable evaluation programs See Note 2

Note 1

For the Modeler objects, if you no longer use the modeler, then they can be ignored and/or deleted. If you still use or require the modeler, then they can import it from the SP5 CD. The Installing LANSA on IBM i Guide states:

4.5.2 Imports on the LANSA IBM i Software DVD or CD-ROM

The Data Modeler is no longer installed during the Partition Initialization. If it is required, you can import it using the Partition Initialization option on the Administration menu.

Note 2

Refer to http://www.lansa.com/support/tips/t0501.htm, which explains why you may have old versions of system evaluation programs in partition program libraries from previous export/imports and provides instructions to clean them up.

If you are still uncertain or unclear about these LANSA objects, or any other LANSA objects, contact LANSA Support for advice.

Customising JSM trace folders based on Application groupings

Many organizations have a requirement to constantly audit/check the trace files. There can be many reasons to do this; for instance to confirm transactions. When JSM application(s) are generating a large number of trace files, it can become a task to locate the required trace file(s).

JSM provides a way to control or name folders into "transaction groups/types". This makes navigating through the generated trace files easier.

In the SERVICE_LOAD command, in addition to TRACE(*YES), put TRACE_NAME(*myGroup1*).
This will concatenate the specified trace name to the folder name. i.e. CLIENT000001_myGroup1 or CLIENT00001_ePayment …etc

<u>You can use a literal naming convention as above, or any of the following special values:</u>

```
>-- TRACE_NAME ------ name -----------------------|
*SERVICE
*PROCESS
*FUNCTION
*JOBNAME
*JOBUSER
*JOBNUMBER
```

Refer to the LANSA Integrator Guide for further details.

How to ensure a particular row stays at the bottom of a grid/list during sort

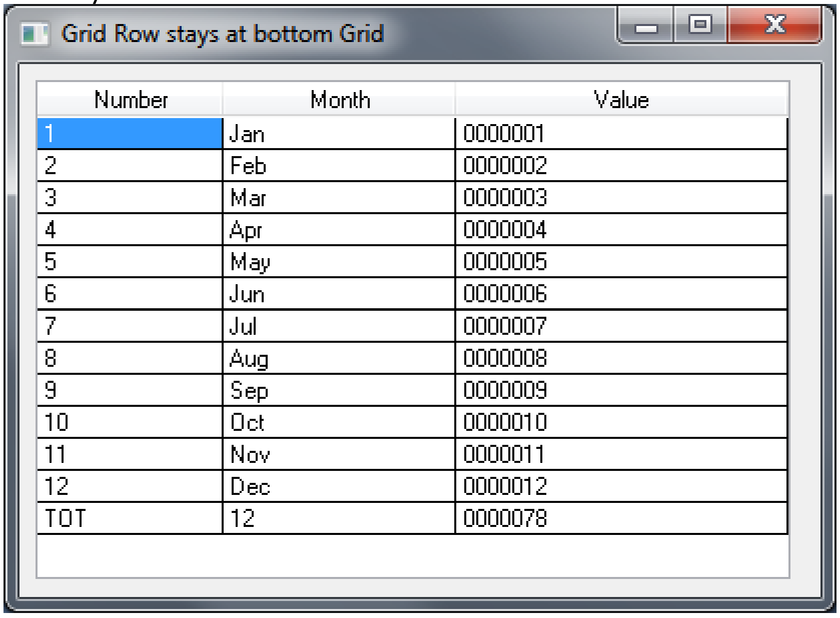
The following sample code describes one method of how how to might implement a grid with a totals row, that allows sorting for all rows while keeping the totals row at the bottom. It is achieved by having a hidden column and manually assigning sort order based column header click events.

Create an RDMLX form with the following code:

```
* ************************************************
*
* COMPONENT: STD_FORM
*
* ************************************************
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(266) Clientwidth(400)
Height(304) Left(702) Top(179) Visible(False) Width(416)
Define_Com Class(#PRIM_GRID) Name(#GRID_1) Captionnoblanklines(True)
Columnbuttonheight(18) Columnbuttonpress(True) Columnscroll(False)
Componentversion(1) Displayposition(1) Height(249) Left(8)
Parent(#COM_OWNER) Showselection(True) Showselectionhilight(False)
Showsortarrow(True) Tabposition(1) Top(8) Width(377)
Define_Com Class(#PRIM_GDCL) Name(#GDCL_1) Caption('Number')
Captiontype(Caption) Displayposition(1) Parent(#GRID_1) Source(#STD_CODE)
Width(25)
Define_Com Class(#PRIM_GDCL) Name(#GDCL_2) Caption('Month')
Captiontype(Caption) Displayposition(2) Parent(#GRID_1) Source(#STD_CODEL)
Width(31)
Define_Com Class(#PRIM_GDCL) Name(#GDCL_3) Caption('Value')
Captiontype(Caption) Displayposition(3) Parent(#GRID_1) Source(#STD_COUNT)
Width(21) Widthtype(Remainder)
* The STD_NUM field is the hidden column that is used to determine which
row is displayed last
Define_Com Class(#PRIM_GDCL) Name(#TotalSort) Parent(#GRID_1)
Sortposition(1) Source(#STD_NUM) Visible(False)

Evtroutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)
* Make some records
Clr_List Named(#GRID_1)
Begin_Loop Using(#STD_ENTRY) To(12)
#STD_CODE := #STD_ENTRY.AsString
#STD_CODEL := (#STD_ENTRY * 100).AsDate( MMYY ).AsDisplayString( MMMMMMMMMM
).Leftmost( 3 )
#STD_COUNT := #STD_ENTRY
#STD_NUM := 0
Add_Entry To_List(#GRID_1)
End_Loop

* Make a record for Total
#STD_CODE := TOT
#STD_CODEL := '12'
```

```
#STD_COUNT := 78
#STD_NUM := 1
Add_Entry To_List(#GRID_1)
Endroutine


Evtroutine Handling(#GDCL_1.Click #GDCL_2.Click #GDCL_3.Click)
Options(*NOCLEARMESSAGES *NOCLEARERRORS) Com_Sender(#RowHeader)
* Make sure the hidden column is the first sort position
#TotalSort.SortPosition := 1
* Is it already sorted by this column?
If Cond((#RowHeader *As #PRIM_GDCL).SortPosition = 2)
* If so, we just change the sort direction
If Cond((#RowHeader *As #PRIM_GDCL).SortDirection = Ascending)
(#RowHeader *As #PRIM_GDCL).SortDirection := Descending
(#RowHeader *As #PRIM_GDCL).Image <= #FP_BM004
Else
(#RowHeader *As #PRIM_GDCL).SortDirection := Ascending
(#RowHeader *As #PRIM_GDCL).Image <= #FP_BM005
Endif
Else
* If not, we set it as the sort column with ascending sort direction
(#RowHeader *As #PRIM_GDCL).SortPosition := 2
(#RowHeader *As #PRIM_GDCL).SortDirection := Ascending
(#RowHeader *As #PRIM_GDCL).Image <= #FP_BM005
Endif

* Reset all other sort positions
For Each(#Column) In(#GRID_1.Columns)
* Already processed the row header that was clicked
If_Ref Com(#Column) Is_Not(*EQUAL_TO #RowHeader)
* Do not change sort order of the hidden column (stays at one)
If_Ref Com(#Column) Is_Not(*EQUAL_TO #TotalSort)

#Column.SortPosition := 0
#Column.Image <= *null

Endif

Endif
Endfor

Endroutine
End_Com
```

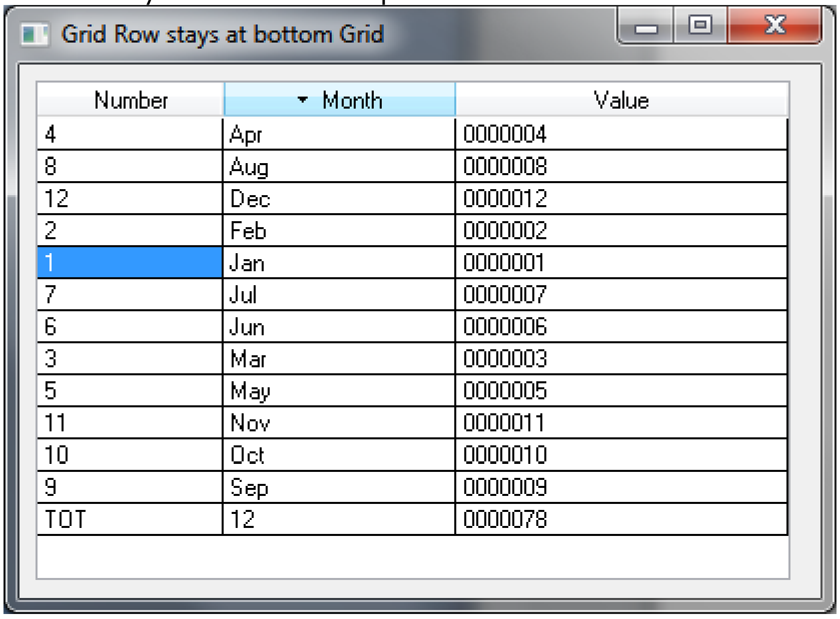
When you start the form:



Grid Row stays at bottom Grid

Number	Month	Value
1	Jan	0000001
2	Feb	0000002
3	Mar	0000003
4	Apr	0000004
5	May	0000005
6	Jun	0000006
7	Jul	0000007
8	Aug	0000008
9	Sep	0000009
10	Oct	0000010
11	Nov	0000011
12	Dec	0000012
TOT	12	0000078

And sort by Month for example:



Grid Row stays at bottom Grid

Number	Month	Value
4	Apr	0000004
8	Aug	0000008
12	Dec	0000012
2	Feb	0000002
1	Jan	0000001
7	Jul	0000007
6	Jun	0000006
3	Mar	0000003
5	May	0000005
11	Nov	0000011
10	Oct	0000010
9	Sep	0000009
TOT	12	0000078

The last entry, the Total row is still at the bottom.