http://www.lansa.com

LANSA

Newsletter
November 2012

LongRange for LANSA

Available for Android devices now!!

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.

Read the next pages for all supported platforms and devices!

In This Issue

LongRange for LANSA (Android)	*page 1*	*Using JSM for calling Stored Procedures*	*page 7*
Webroutines fail to compile	*page 5*	*Use AS2 in LANSA Composer*	*page 10*
Creating a PDF Document	*page 6*	*VLF Web problem with Chrome 17*	*page 11*

LongRange Supported Platforms

Platform	Supported Versions	Remarks / Conditions
IBM i Server	V5R3M1 V5R4M1 V5R4M5 V6R1M1 V7R1M0	RPG/COBOL based LongRange systems
	V5R4M1 V5R4M5 V6R1M1 V7R1M0	RPG/COBOL based LongRange systems that also use LongReach to support the Document Views feature
	V5R4M1 V5R4M5 V6R1M1 V7R1M0	LANSA based systems
Windows Server	Same as LANSA	Only applicable to LANSA based system. See http://www.lansa.com/support/supportedversions.htm
Windows Developer PC (LongRange Studio)	Windows XP Windows Vista Windows 7	.NET Framework 3.5 or later required
iOs (iPhone, iPad, iPod)	5.0 or later	
Android	2.3 or later	Also see the following sections

Supported Android Devices

We support Android 2.3 (or later) on a supported device. We define a supported device to be a device that:

- Is commercially available in the USA or Europe and, in other countries, available in an identical form to either the USA or European versions. In other words, a device that's widely available through a network operator or retail sales outlets.
- Has the latest Android version officially supplied by the manufacturer installed on it.
- Meets the following minimum hardware requirements:
    - CPU supports ARM instruction set
    - CPU runs at 600Mhz or above
    - Screen resolution no less than 240 x 320

Android Devices and Versions Used as Testing Reference Platforms

Currently we use these devices as testing reference platforms:

Device Model	Android Version
Huawei Sonic phone	2.3.5
Samsung Galaxy Tab 7”	2.3.6
Samsung Galaxy Tab II 7” Tablet	4.0
Toshiba AT100 Tablet	3.2.1
Samsung Galaxy Tab 10” Tablet	3.2
Samsung Galaxy Tab 2 10.1	4.0.3
Samsung Galaxy Nexus	4.1.1
Samsung Galaxy S3	4.0.4
HTC G2	2.3.4
Acer Iconia A500 Tablet	4.0.3

We normally verify new LongRange Android versions against the above testing reference platforms. We also use them first in trying to reproduce a problem you may report. If a problem cannot be reproduced on a testing reference platform, it may take longer and be more complicated to resolve an issue.

Using Android Devices outside the Supported Device Definition

You can use Android devices outside the supported device definition providing they use Android 2.3 (or later) and ARM chips.

Please note that issues or problems that you report will only be handled under the terms of your license agreement if they are reproducible on a supported device.

If they are not reproducible on a supported device, any remedial and/or enhancement work performed by LANSA may attract a fee for service.

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.

Webroutines fail to compile after upgrading to V12 SP1 when using an existing custom TSP

After upgrading to V12 SP1 you may find that are not able to generate your webroutine/layouts using an existing custom TSP.

The problem webroutine compiles with errors such as:
PRC0047/LII0855E Failed to retrieve configuration for Technology Service
PRC0047/LII0846E Failed to generate XML/XSL

As of V12 SP1 the generator requires that stylesheets contain the weblet_template_call template.

Compliance

To comply with V12 SP1 and beyond, add the following code to the end of your custom TSP:

```
<!-- Template needed by the inliner (even if the TSP doesn't support inlining) -->
            <xsl:template name="weblet_template_call">
            <xsl:param name="field"/>
            <xsl:param name="fld_or_col"/>
            <xsl:param name="name"/>
            <xsl:param name="displaymode"/>
            <xsl:param name="use_weblet"/>
            <xsl:param name="weblet"/>
            <xsl:param name="inline_param" select="false()"/>
            </xsl:template>
```

You will now be able to generate your webroutine without issue.

Creating a PDF Document Using Data from Database

Question

How do you generate a PDF using data from a database? For example, if you have employee details, which includes all their header information and other detail related information like skills, multiple addresses, etc. and you want to put all of this into a PDF document.

What are the solutions worth considering and what is the complexity of each one?

Answer

What is required is to create PDF's dynamically. There are various options when it comes to creating dynamic PDF's. These include:

- PDFDocument service: This uses the shipped LANSA Integrator service to design and build dynamic PDF documents. This is good for creating basic and complex PDF's but requires knowledge of LANSA Integrator, as well as some expertise with XML when designing the layout (since the layout in XML will need to be hand-cranked).
- CodeStart XSL-FO: This method has a developer UI element which enables the PDF layout to be designed using WAMS or Microsoft Word, along with a custom LANSA Integrator service (FOPService) to generate the dynamic PDF. The advantage of this approach is that it makes it easy to design rich PDF layouts using a WISYWIG editor (e.g. WAM XSL Editor). It does require some expertise with WAM development to create the layouts at design time. Note that this is a custom and chargeable service offered through LANSA Professional Services that can be used either through a CodeStart offering or as part of a larger Services solution. See CodeStart – XSL-FOP PDF Service (http://www.lansa.com/services/xsl-fop-pdf-service.htm) for the CodeStart details.

Using JSM for calling Stored Procedures

SQLService can be used to allow calling (CALL Command) of stored procedures, using IN, OUT and INOUT parameters.

You define each parameter by adding it to a working list in the same order as the stored procedure.

- Use the SET PARAMETER(*CALL) to pass the working list to the SQLService.
- Use the GET OBJECT(*PARAMETERCALL) to get the working list from the SQLService.
- The EXECUTE CALL command will update the parameter call working list with returned values.
- You can still use return parameter and results with this command.

Sample Stored Procedure and JSM Function with Call

IBMi stored Procedure

```
CREATE PROCEDURE JSMJDBC/CALLEXEOUT ( IN CODE CHAR(10), INOUT STS CHAR(10),
OUT VAL INT )
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN
DECLARE SQLCODE INTEGER DEFAULT 0;
DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
UPDATE JSMJDBC/TBLNAME SET SALARY=16000.26 WHERE ID = CODE;
SET VAL = 34;
SET STS = 'DONE';
END
```

JSM Function

```
* ********* Beginning of RDML commands **********
FUNCTION OPTIONS(*DIRECT)
* *********
DEFINE FIELD(#JSMSTS) TYPE(*CHAR) LENGTH(20)
DEFINE FIELD(#JSMMSG) TYPE(*CHAR) LENGTH(255)
DEFINE FIELD(#JSMCMD) TYPE(*CHAR) LENGTH(255)
* *********
DEFINE FIELD(#CDIR) TYPE(*CHAR) LENGTH(6)
DEFINE FIELD(#CTYP) TYPE(*CHAR) LENGTH(10)
DEFINE FIELD(#CVAL) TYPE(*CHAR) LENGTH(50)
DEF_LIST NAME(#CALLLST) FIELDS((#CDIR) (#CTYP) (#CVAL)) TYPE(*WORKING)
* *********
* ********* 'Open service'
* *********
USE BUILTIN(JSM_OPEN) TO_GET(#JSMSTS #JSMMSG)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
* *********
* ********* 'Load service'
* *********
```

```
USE BUILTIN(JSM_COMMAND) WITH_ARGS('SERVICE_LOAD SERVICE(SQLSERVICE)
TRACE(*YES)') TO_GET(#JSMSTS #JSMMSG)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
* *********
CHANGE FIELD(#JSMCMD) TO('CONNECT DRIVER(DB2) DATABASE(JSMJDBC) USER(ALICK)
PASSWORD(xxxx)')
USE BUILTIN(JSM_COMMAND) WITH_ARGS(#JSMCMD) TO_GET(#JSMSTS #JSMMSG)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
* *********
USE BUILTIN(JSM_COMMAND) WITH_ARGS('SET ONWARNING(*STOP) SQLSTATE(*ERROR)')
TO_GET(#JSMSTS #JSMMSG)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
* *********
* ********* Create call parameter
* *********
CLR_LIST NAMED(#CALLLST)
CHANGE FIELD(#CDIR) TO('''*IN''')
CHANGE FIELD(#CTYP) TO('''*CHAR''')
CHANGE FIELD(#CVAL) TO(A1002)
ADD_ENTRY TO_LIST(#CALLLST)
CHANGE FIELD(#CDIR) TO('''*INOUT''')
CHANGE FIELD(#CTYP) TO('''*CHAR''')
CHANGE FIELD(#CVAL) TO(SOMETHING)
ADD_ENTRY TO_LIST(#CALLLST)
CHANGE FIELD(#CDIR) TO('''*OUT''')
CHANGE FIELD(#CTYP) TO('''*INTEGER''')
CHANGE FIELD(#CVAL) TO(*BLANK)
ADD_ENTRY TO_LIST(#CALLLST)
**********
CHANGE FIELD(#JSMCMD) TO('SET PARAMETER(*CALL)
SERVICE_LIST(CDIR,CTYP,CVAL)')
USE BUILTIN(JSM_COMMAND) WITH_ARGS(#JSMCMD) TO_GET(#JSMSTS #JSMMSG
#CALLLST)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
* *********
* ********* Call procedure
* *********
CHANGE FIELD(#JSMCMD) TO('EXECUTE CALL("CALLEXEOUT(?,?,?)")
CALLTYPE(*EXECUTE)')
USE BUILTIN(JSM_COMMAND) WITH_ARGS(#JSMCMD) TO_GET(#JSMSTS #JSMMSG)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
* *********
* ********* Get call parameter
* *********
CHANGE FIELD(#JSMCMD) TO('GET OBJECT(*PARAMETERCALL)
SERVICE_LIST(CDIR,CTYP,CVAL)')
USE BUILTIN(JSM_COMMAND) WITH_ARGS(#JSMCMD) TO_GET(#JSMSTS #JSMMSG
#CALLLST)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
**********
SELECTLIST NAMED(#CALLLST)
DISPLAY FIELDS((#CDIR) (#CTYP) (#CVAL))
ENDSELECT
* *********
USE BUILTIN(JSM_COMMAND) WITH_ARGS('DISCONNECT') TO_GET(#JSMSTS #JSMMSG)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
* *********
* ********* 'Close service'
* *********
USE BUILTIN(JSM_CLOSE) TO_GET(#JSMSTS #JSMMSG)
EXECUTE SUBROUTINE(CHECK) WITH_PARMS(#JSMSTS #JSMMSG)
```

```
* *********
* ********* SUB ROUTINES
* *********
SUBROUTINE NAME(CHECK) PARMS((#JSMSTS *RECEIVED) (#JSMMSG *RECEIVED))
* *********
IF COND('#JSMSTS *NE OK')
* *********
DISPLAY FIELDS((#JSMSTS)(#JSMMSG))
USE BUILTIN(JSM_CLOSE) TO_GET(#JSMSTS #JSMMSG)
* *********
MENU MSGTXT('Java service error has occurred')
* *********
ENDIF
* *********
ENDROUTINE
* ********* End of RDML commands **********
```

Watch the December 2012 Newsletter!!

Is it possible to use AS2 in LANSA Composer?

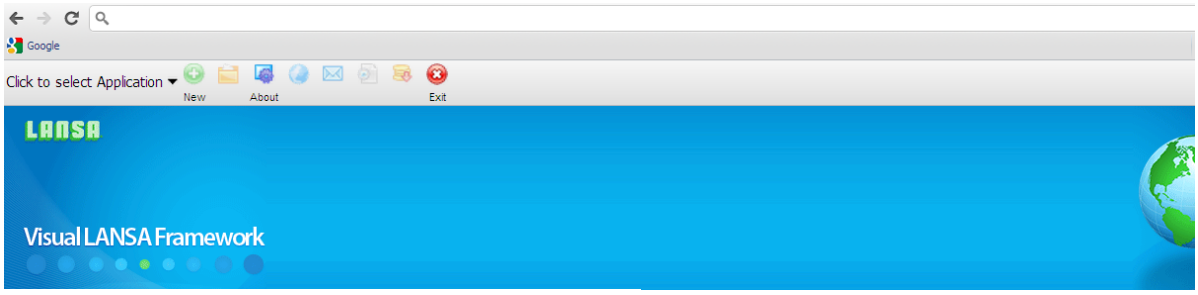
LANSA Composer does not provide AS2 support as part of its standard offerings. That is not to say, however, that the AS2 protocol cannot be implemented in LANSA Composer. There are 2 possible ways to implement AS2 in LANSA Composer

- use the LANSA Data Secure Direct (http://www.lansa.com/products/datasecuredirect.htm) product in combination with LANSA Composer. This method has already been implemented successfully in solutions.
- write a custom solution/activity using the AS2 services that are available in LANSA Integrator (see Appendix D. AS2 and AS3 Services in the LANSA Integrator Guide).

For more information on LANSA Composers EDI mapping capabilities, refer to LANSA Composer for EDI (http://www.lansa.com/products/composer_edi.htm).

EPC870 VLF-Web does not work on CHROME 17

If you run VLF-Web on Chrome 17 you will find that the Applications and Business Objects appear on the tool bar regardless of your Framework's setting and that you can no longer execute them.



This is now verified as a low priority defect which can be worked around by adding +touch=N into the URL.

Previous versions of Chrome browsers do not have this problem.