LANSA NEWSLETTER

LANSA Open for .Net 6

LANSA Open for .NET enables .NET developers with capabilities to incorporate IBM i (AS/400, iSeries, System i) services in Windows rich-client and Web applications, allowing them to view the IBM i data and programs as resources within Visual Studio.

Now, we are pleased to announce that LANSA continues to improve LANSA Open for .NET – now featuring the newly added .NET 6 support. This new capability enables IBM i modernization and integration with applications built with .NET 6.

LANSA Open for .NET – now supporting .NET 6

Building on the solid foundation and success of accessing your IBM i applications for .NET, LANSA Open for .NET new release extends support to include .NET 6. .NET 6 incorporates new features and performance enhancements.

INSIDE THIS ISSUE

This release will help enable application developers to continue to utilize the Visual LANSA repository on IBM i or Windows systems, but also have access to new .NET technologies such as ASP.NET, Blazor and cross-platform development.

In addition, the LANSA Component Model Designer Visual Studio plugin has been updated to work in Visual Studio 2017, 2019 & 2022. Also, DotNet monitor and OpenTelemetry, LANSA Open for .NET provide better Cloud diagnostics, and WebAssembly support is more capable and efficient.

With the launch of this product, LANSA confirms its long-term commitment to updating its products to introduce the latest and greatest technologies to the legendary reliability, security, and performance of the IBM i architecture.

About LANSA Open for .NET

LANSA Open for .NET is a Visual Studio plugin that provides Windows developers direct access to IBM i (AS/400, iSeries, System i) resources within their .NET applications at full speed and without leaving Visual Studio.

HTTPService - Cannot get property HTTP Header "AUTHORIZATION"

This is an IBM iSeries HTTP Apache issue/configuration which seems not to be documented in LANSA, as for example "CGIConvMode BINARY", "DefaultNetCCSID 00813", "DefaultFsCCSID 00875" are documented. This particular setting is not documented in LANSA at all.

In order to allow scripts, such as JSMDIRECT.PGM, to implement HTTP Basic authentication you must allow this in the Apache HTTPD.CONF file.

To do this add the Directive "CGIPassAuth On" for the JSM library, as IBM has set this Directive as "off" by Default!!

<Directory /QSYS.LIB/<LANSA_INTEGRATOR_LIB>.LIB>
CGIPassAuth On
...

After setting the above directive, the Authorization HTTP Header parameter is parsed by LANSA Integrator.

Return status LANSA Integrator

Possible returned values in #JSMXSTS / #JSMSTS (note that this list may not be complete!)

Most LANSA Integrator code samples just check the IF (NOT) OK ……
……. but there are a lot of return JSM status values. For example:

Return Value	***Related to***	***Comment***
OK	General	
ERROR	General	
FATAL	General	
NOLIST	XML service	Not always a problem
NOFRAGMENT	XML service	Not always a problem
NOCONNECT	General	
NO		
NOCELL	Excel	Not always a problem
NOROW	Excel	Not always a problem
NOSHEET	Excel	Not always a problem
NOTEMPLATE	Excel	Not always a problem
NOUID	SMTP	
NOMAIL	SMTP	
NOTEXT	SMTP	Not always a problem
NOATTACHMENT	SMTP	Not always a problem
NOINSTRUCTION		
NOMESSAGE		
NOCONTENT		Not always a problem
SOAPFAULT	SOAP	
WARNING		
NORESULT		
NONE		
EQUAL		
NOT_EQUAL		
EXIST		
NOT_EXIST		
NONEXT		
ELEMENTSTART		
TEXT		
ELEMENTEND		
COMMENT		
INSTRUCTION		
FAILED_REFERRAL	LDAP service	
NOACCESS	LDAP service	
NOENTRY	LDAP service	
NOATTRIBUTE	LDAP service	
ENTRY_EXISTS	LDAP service	
ATTRIBUTE_EXISTS	LDAP service	
OBJECT_VIOLATION	LDAP service	
CONSTRAINT_VIOLATION	LDAP service	

Sample to use these:

```
Mthroutine Name(JSMXCheck) Options(*RECEIVES_MESSAGES *RETURNS_MESSAGES)

*
If Cond('#JSMXSTS *EQ OK')
* continue if all okay
Else
* show error
If Cond('#JSMXSTS *EQ NOCONNECT')
Message Msgtxt('Can NOT connect to JSM, please CHECK JSM Service is running!')
Endif
* close , but only on fatal errors !!
If Cond('(#JSMXSTS *EQ ERROR) *or (#JSMXSTS *EQ FATAL)')
Message Msgtxt(#jsmXmsg)
Use Builtin(JSMX_CLOSE) With_Args(#JSMXHDLE1) To_Get(#JSMXSTS #JSMXMSG)
Else
If Cond('(#JSMXSTS *EQ SOAPERROR)')
* for SOAP: get the SOAP error
* ....  to be build
*
*
Else
* just send a message and continue
#STD_QSEL := #JSMXSTS + '- ' + #JSMXMSG + '- ' + #JSMXCMD
Message Msgtxt(#STD_QSEL)
* or add the message to a file (batch ) / list (form) :
* ...
*  ...
Endif
Endif
Endif
* or just in the most simplest way
* If ('#JSMXsts *NE OK')
* #STD_QSEL := 'JSM Error: ' + #JSMXSTS + ' - in Command : ' + #JSMXCMD
* Message Msgid(DCM9993) Msgf(DC@M01) Msgdta(#JSMXMSG #std_qsel)
* Endif

Endroutine
```

Learn LANSA: new training material

The learn.lansa.com pages contains a lot of online LANSA training material. Recently, some interesting new workshops have become available.

LANSA V15 – Get Started with APIs

Learn how to create and publish RESTFul API services with Visual LANSA server modules. Learn also how to consume RESTFul API services from the browser or on the server.

https://learn.lansa.com/courses/get-started-with-apis

LANSA V15 – Create Secure RESTFul APIs

Understand basic and JWT authentication. Learn how to create APIs using Basic or JWT Authentication.

https://learn.lansa.com/courses/lansa-v15-create-secure-restful-apis

LANSA V15 – Implementing Web Routing

A short course which demonstrated how web routing is implemented.

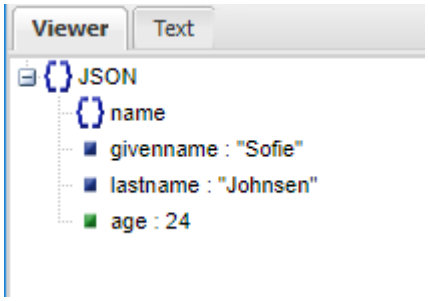
https://learn.lansa.com/courses/visual-lansa-v15-use-web-routing

#XPRIM_Json: Create simple Json

This example creates a very simple Json and based on the generated json, makes a pc file of it.

```
* Simple example:
Define_Com Class(#XPRIM_Json) Name(#json)
Define_Com Class(#XPRIM_JsonObject) Name(#jsonObj) Reference(*DYNAMIC)

#jsonObj <= #json.CreateRootObject

#jsonObj.InsertObject Key('name')
#jsonObj.InsertString Key("givenname") String('Sofie')
#jsonObj.InsertString Key("lastname") String("Johnsen")
#jsonObj.InsertNumber Key('age') Number(24)

#jsonObj.SerializeToFile Path('C:\TEMP\JsonExampleSimple.json')
```

Result:

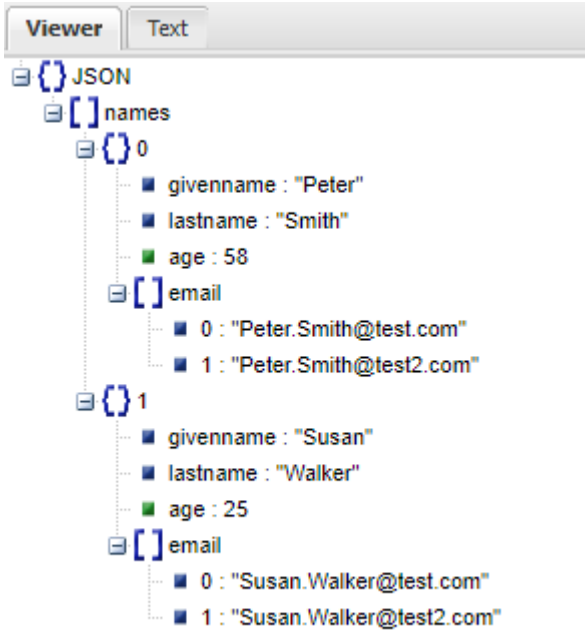


#XPRIM_Json: Create Json with two arrays

This example creates a Json file with two arrays and based on the generated json, makes a pc file of it.

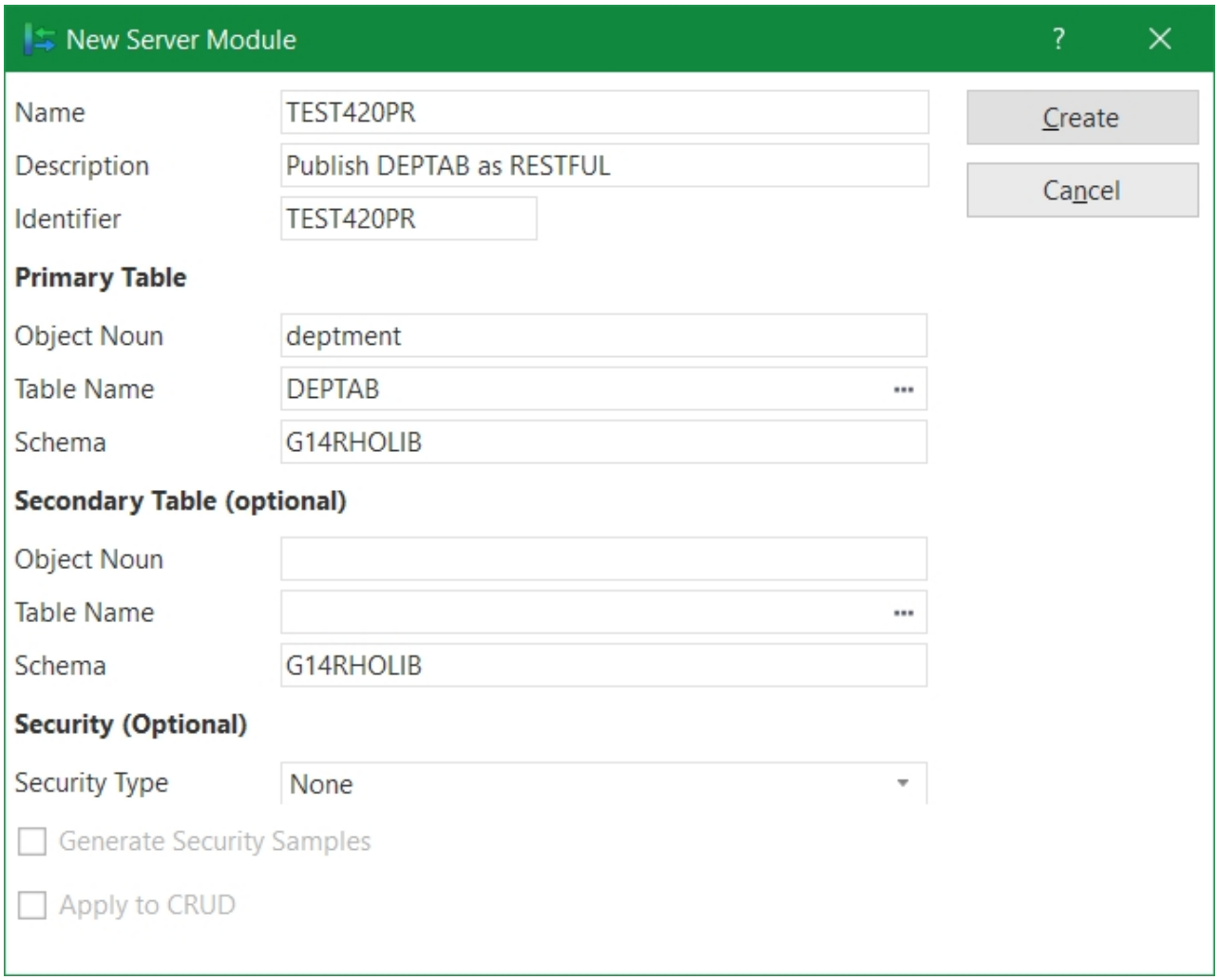
```
* Example with two Array's
Define_Com Class(#XPRIM_Json) Name(#json)
Define_Com Class(#XPRIM_JsonObject) Name(#jsonObj) Reference(*DYNAMIC)
Define_Com Class(#XPRIM_JsonObject) Name(#jsonNameObject) Reference(*DYNAMIC)

Define_Com Class(#XPRIM_JsonArray) Name(#jsonNamesArray) Reference(*DYNAMIC)
Define_Com Class(#XPRIM_JsonArray) Name(#jsonEmailArray) Reference(*DYNAMIC)

* Create root for JSON document
#jsonObj <= #json.CreateRootObject

* Create array for names
#jsonNamesArray <= #jsonObj.InsertArray( 'names' )

* Add name object, first name
#jsonNameObject <= #jsonNamesArray.InsertObject
* Add 1st level data in json name object
#jsonNameObject.InsertString Key("givenname") String('Peter')
#jsonNameObject.InsertString Key("lastname") String("Smith")
#jsonNameObject.InsertNumber Key('age') Number(58)

* Add email array into name object
#jsonEmailArray <= #jsonNameObject.InsertArray( 'email' )
#jsonEmailArray.InsertString String('Peter.Smith@test.com')
#jsonEmailArray.InsertString String('Peter.Smith@test2.com')

* Add name object, second name
#jsonNameObject <= #jsonNamesArray.InsertObject
* Add 1st level data in json name object
#jsonNameObject.InsertString Key("givenname") String('Susan')
#jsonNameObject.InsertString Key("lastname") String("Walker")
#jsonNameObject.InsertNumber Key('age') Number(25)

* Add email array into name object
#jsonEmailArray <= #jsonNameObject.InsertArray( 'email' )
#jsonEmailArray.InsertString String('Susan.Walker@test.com')
#jsonEmailArray.InsertString String('Susan.Walker@test2.com')

#jsonObj.SerializeToFile Path('C:\TEMP\JsonExampleWithTwoArrays.json')
```

Result:


Viewer
Text
JSON
names
0
givenname : "Peter"
lastname : "Smith"
age : 58
email
0 : "Peter.Smith@test.com"
1 : "Peter.Smith@test2.com"
1
givenname : "Susan"
lastname : "Walker"
age : 25
email
0 : "Susan.Walker@test.com"
1 : "Susan.Walker@test2.com"

#XPRIM_JsonObject: Updating a record using PUT

If you publish a restful service, for example for DEPTAB using the assistant:



So, this is now available:
http://localhost:8080/rho/TEST420PR/deptments/ADM

(Notice that in my installation the partition is part of the name. In some others, the partition is not needed)

You can now create a server module (example TEST420SM) to request the update of a record using a PUT like this (just copy/page):

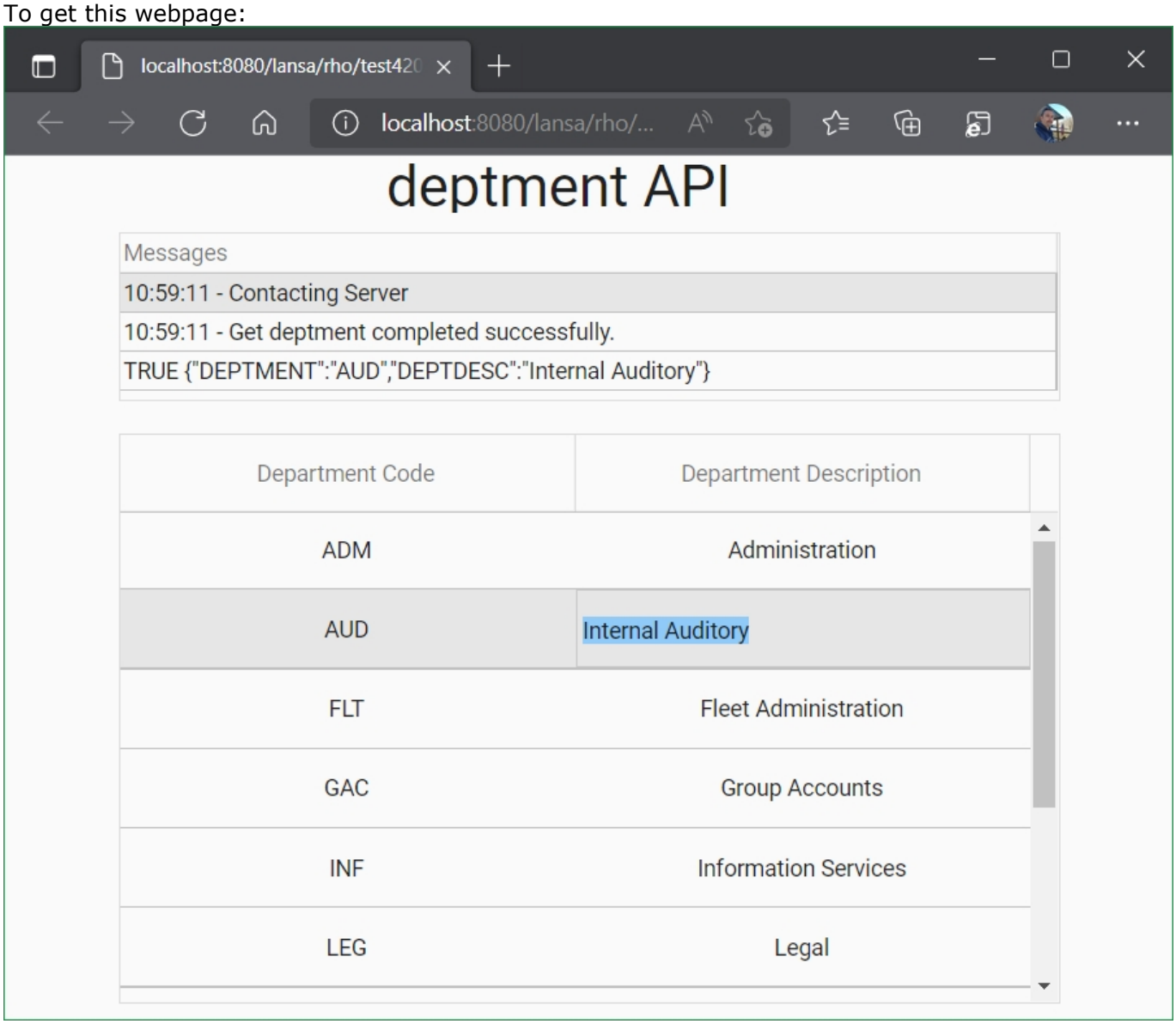
```
Begin_Com Role(*EXTENDS #PRIM_SRVM)
Group_By Name(#Fields) Fields(#DEPTMENT #DEPTDESC)

Srvroutine Name(SendPUT)
Group_Map For(*Input) Group(#Fields)
Field_Map For(*Output) Field(#STD_QSEL)

Define_Com Class(#XPRIM_HttpRequest) Name(#Req)
Define_Com Class(#XPRIM_UriBuilder) Name(#Url)
Define_Com Class(#XPRIM_JsonObject) Name(#JsonObject)

#Url.SetScheme( 'http' )
#Url.SetHost( 'localhost:8080' )
#Url.SetPath( ("/" + #partition + "/TEST420PR/deptments/" + #DEPTMENT) )

#JsonObject.InsertString Key("DEPTMENT") String(#DEPTMENT)
#JsonObject.InsertString Key("DEPTDESC") String(#DEPTDESC)

#Req.Content.AddJsonObject( #JsonObject )

* Execute the request (POST verb)
#Req.DoPut Url(#Url)

If (#Req.Response.IsSuccessfulRequest)
#STD_QSEL := #Req.Response.IsSuccessfulRequest + ' ' + #Req.Response.AsString.AsNativeString
Else
#STD_QSEL := #Req.Response.ErrorCode.AsNativeString + ' ' + #Req.Response.ErrorMessage.AsNativeString
Endif
Endroutine

End_Com
```

And call this server module for example from this webpage (just copy/paste):

```
Begin_Com Role(*EXTENDS #PRIM_WEB) Theme(#SYS_THEME<MaterialDesignBlue>) LayoutManager(#LayoutMain)
Height(721)

Define_Com Class(#PRIM_TBLO) Name(#LayoutMain)
Define_Com Class(#PRIM_TBLO.Column) Name(#LayoutMainColumn1) DisplayPosition(1) Parent(#LayoutMain)
Width(2)
Define_Com Class(#PRIM_TBLO.Column) Name(#LayoutMainColumn2) DisplayPosition(2) Parent(#LayoutMain)
Width(20)
Define_Com Class(#PRIM_TBLO.Column) Name(#LayoutMainColumn3) DisplayPosition(3) Parent(#LayoutMain)
Width(2)
Define_Com Class(#PRIM_TBLO.Row) Name(#LayoutMainRow1) DisplayPosition(1) Parent(#LayoutMain) Height(2)
Define_Com Class(#PRIM_TBLO.Row) Name(#LayoutMainRow2) DisplayPosition(2) Parent(#LayoutMain) Height(6.76)
Define_Com Class(#PRIM_TBLO.Row) Name(#LayoutMainRow3) DisplayPosition(3) Parent(#LayoutMain)
Height(20.24)
Define_Com Class(#PRIM_TBLO.Item) Name(#LayoutMainItem1) Manage(#laTitle) Parent(#LayoutMain)
Row(#LayoutMainRow1) Sizing(None) Column(#LayoutMainColumn2)
Define_Com Class(#PRIM_TBLO.Item) Name(#LayoutMainItem2) Manage(#Messages) Parent(#LayoutMain)
Row(#LayoutMainRow2) Column(#LayoutMainColumn2) MarginBottom(10) MarginLeft(10) MarginRight(10)
MarginTop(10)
Define_Com Class(#PRIM_TBLO.Item) Name(#LayoutMainItem3) Manage(#deptmentlist) Parent(#LayoutMain)
Row(#LayoutMainRow3) Column(#LayoutMainColumn2) MarginBottom(10) MarginLeft(10) MarginRight(10)
MarginTop(10)

Define_Com Class(#PRIM_LABL) Name(#laTitle) Caption('deptment API') DisplayPosition(1) Ellipses(Word)
Height(33) Left(476) Parent(#COM_OWNER) TabPosition(1) TabStop(False) Top(9) VerticalAlignment(Center)
Width(249) ThemeDrawStyle('Title')
Define_Com Class(#PRIM_LIST) Name(#Messages) DisplayPosition(2) Left(110) Parent(#COM_OWNER)
TabPosition(2) Top(60) Width(980) Height(148)
Define_Com Class(#PRIM_LIST.String) Name(#MessagesColumn1) ColumnWidth(1) DisplayPosition(1)
Parent(#Messages) SortOnClick(True) Source(#STD_QSEL) ColumnCaption('Messages') ColumnCaptionAlign(Left)
ColumnCaptionType(Caption) ColumnUnits(Proportion)
Define_Com Class(#PRIM_LIST) Name(#deptmentlist) ColumnLines(False) DisplayPosition(3) Height(483)
Left(110) Parent(#COM_OWNER) TabPosition(3) Top(228) Width(980) ColumnHeaderHeight(48) RowHeight(48)

* The first column will not have the columnreadonly false
Define_Com Class(#PRIM_LIST.String) Name(#deptmentColumn01) ColumnWidth(1) DisplayPosition(1)
Parent(#deptmentlist) SortOnClick(True) Source(#DEPTMENT) DisplayAlignment(Center) ColumnUnits(Proportion)
Define_Com Class(#PRIM_LIST.String) Name(#deptmentColumn02) ColumnWidth(1) DisplayPosition(2)
Parent(#deptmentlist) SortOnClick(True) Source(#DEPTDESC) DisplayAlignment(Center) ColumnUnits(Proportion)
ColumnReadOnly(False)

Define_Com Class(#STD_INT) Name(#HttpStatus)
```

```
Evtroutine Handling(#Com_owner.Initialize)
Clr_List Named(#Messages)
#COM_OWNER.Getdeptment
Endroutine

Mthroutine Name(Getdeptment) Access(*Private)
Define_Com Class(#Prim_Web.HttpRequest) Name(#Getdeptment)
Define_Com Class(#prim_web.JsonElement) Name(#JRoot_Element) Reference(*Dynamic)
Define_Com Class(#PRIM_WEB.json) Name(#Json)

#Getdeptment.Url := ("/" + #partition + "/TEST420PR/deptments")

#STD_QSEL := #TIMEX.AsDisplayString( HHsMMsSS ) + " - Contacting Server"
Add_Entry To_List(#Messages)

#Getdeptment.ExecuteAsync

Evtroutine Handling(#Getdeptment.Completed)
Clr_List Named(#deptmentlist)
#HttpStatus := #Getdeptment.Response.StatusCode

#Json := #Getdeptment.Response.Content
#JRoot_Element <= #Json.RootItem

If (#HttpStatus <> 200)
#STD_QSEL := #TIMEX.AsDisplayString( HHsMMsSS ) + " - Error Status Code: " + #HttpStatus.AsString + "
Message: " + #Getdeptment.Response.StatusText + " received from server."
Add_Entry To_List(#Messages)
Else
For Each(#ArrayElem) In(#JRoot_Element)
For Each(#Entry) In(#ArrayElem)
Case (#Entry.Key)
When Value_Is(= "DEPTMENT")
#DEPTMENT := #Entry.AsString.Trim
When Value_Is(= "DEPTDESC")
#DEPTDESC := #Entry.AsString.Trim
Endcase
Endfor
Add_Entry To_List(#deptmentlist)
Endfor

#STD_QSEL := #TIMEX.AsDisplayString( HHsMMsSS ) + " - Get deptment completed successfully."
Add_Entry To_List(#Messages)
Endif
Endroutine

Evtroutine Handling(#Getdeptment.Failed)
#Sys_web.Alert Caption("Get deptment Failed")
Endroutine
Endroutine

* Just keep the routines you want for the fields you want to allow update
Evtroutine Handling(#deptmentColumn02.Enter)
#COM_OWNER.Updatedeptment
Endroutine

Mthroutine Name(Updatedeptment) Access(*Private)
Define_Com Class(#TEST420SM.SendPUT) Name(#SendPUT)
Group_By Name(#Fields) Fields(#DEPTMENT #DEPTDESC)

#SendPUT.Execute Fields(#Fields) STD_QSEL(#STD_QSEL)
Add_Entry To_List(#Messages)
Endroutine
End_Com
```

To get this webpage:



Differences between #Com_Owner, #Com_Ancestor and #Com_Self

#Com_Owner, #Com_Ancestor and #Com_Self

The generic name #Com_Owner can be used to refer to the current component. In the context of inheritance there are two other generic names you can use for components:

- #Com_Ancestor
- #Com_Self

#Com_Ancestor is a generic reference to the ancestor of the current component and #Com_Self is a generic reference to the currently executing component.

#Com_Owner and #Com_Self

To see in what way #Com_Owner and #Com_Self differs, create two reusable parts with this code:

BUTTON1

This button reusable part contains a method Method_One which displays a message saying “Hello”. The message is displayed when this button (Button 1, in other words #Com_Owner) is clicked.

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_PHBN) Caption('Button 1') Height(24) Width(185)
Displayposition(1) Tabposition(1)

Mthroutine Name(Method_One)
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK INFO 'Message' 'Hello') To_Get(#STD_OBJ)
Endroutine

Evtroutine Handling(#COM_OWNER.Click)
#com_owner.Method_One
Endroutine

End_Com
```

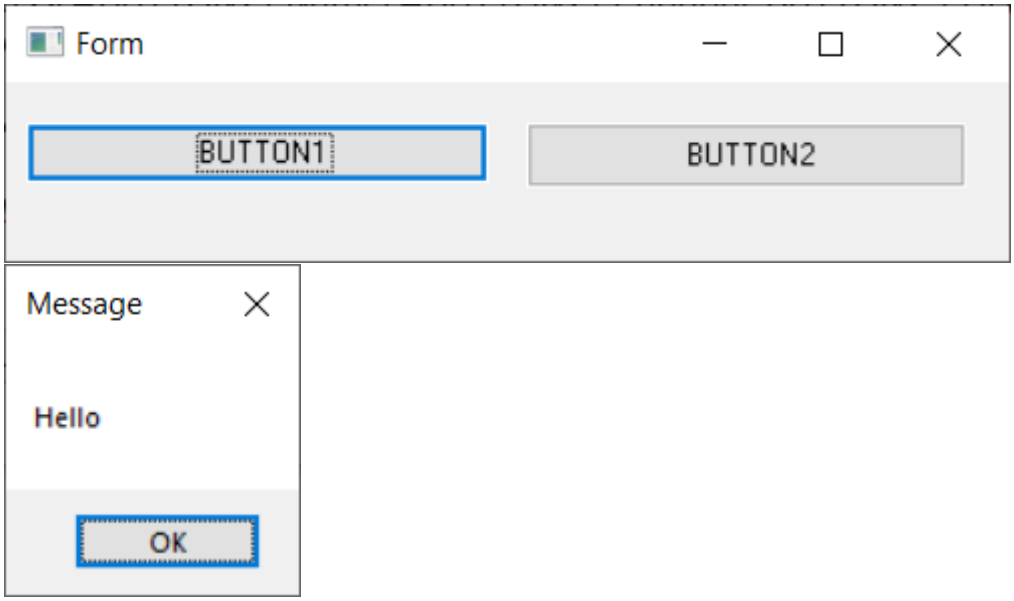
BUTTON2

This button reusable part inherits from BUTTON1 and redefines Method_One so that the message says “Bye”.

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #BUTTON1) Caption('Button 2') Height(26) Width(176)

Mthroutine Name(Method_One) Options(*REDEFINE)
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK INFO 'Message' 'Bye') To_Get(#STD_OBJ)
Endroutine

End_Com
```

Add these buttons to a form and compile and execute the form. Click on both buttons. They both display a message saying 'Hello':

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientwidth(401) Clientheight(71) Componentversion(2)
Left(742) Top(193)

Define_Com Class(#BUTTON1) Name(#BUTTON1) Caption('BUTTON1') Left(8)
Parent(#COM_OWNER) Top(16)
Define_Com Class(#BUTTON2) Name(#BUTTON2) Caption('BUTTON2') Displayposition(2) Left(208)
Parent(#COM_OWNER) Tabposition(2) Top(16)

Evtroutine Handling(#com_owner.CreateInstance)

Set Com(#com_owner) Caption(*component_desc)

Endroutine

End_Com
```

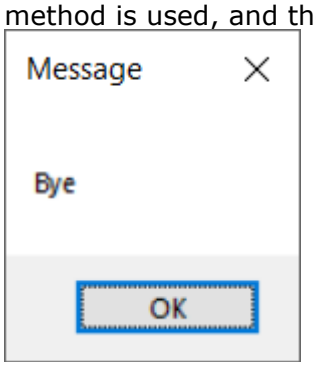


In other words, the redefined method which would display the message 'Bye' is not used for Button 2 because the Click event defined in Button 1 invokes the method in the component where it is defined (#Com_Owner):
#com_owner.Method_One

Next, change the invoke command in the Click event of Button 1 to use #Com_Self instead of #Com_Owner so that the Click event invokes the method in the component where the event is executed:
#com_self.Method_One

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_PHBN) Caption('Button 1') Height(24) Width(185)
Displayposition(1) Tabposition(1)

Mthroutine Name(Method_One)
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK INFO 'Message' 'Hello') To_Get(#STD_OBJ)
Endroutine

Evtroutine Handling(#COM_OWNER.Click)
#com_self.Method_One
Endroutine

End_Com
```

Compile BUTTON1 and execute your form again. This time when you click on Button 2, the redefined method is used, and the message 'Bye' is displayed:



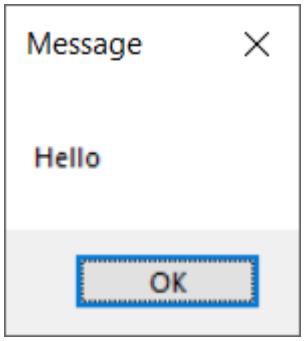
#Com_Ancestor

If you want to invoke a method in an ancestor component, you can refer to it by qualifying the name of the method using the generic name #com_ancestor.

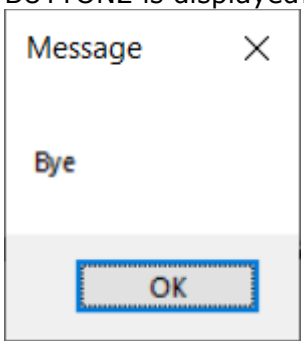
For example, to continue with the above example, you can invoke the Method_One method in BUTTON1 from the redefined Method_One in BUTTON2 like this:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #BUTTON1) Caption('Button 2') Height(26) Width(176)

Mthroutine Name(Method_One) Options(*REDEFINE)
#com_ancestor.Method_One
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK INFO 'Message' 'Bye') To_Get(#STD_OBJ)
Endroutine

End_Com
```

Compile BUTTON2 and run your test form. When you click on Button 2 first the Method_One from the ancestor BUTTON1 is invoked:



And then when you click OK, the Message_Box_Show builtin showing the text 'Bye' defined in BUTTON2 is displayed:



jsmbcprov.jar is not signed by a trusted signer

There is now a fix available for an issue that has been reported many times.

This issue can be encountered when using LANSA Integrator services or LANSA Composer services, but generally has been reported by customers using AS2.

A typical scenario would be

A client applies the latest IBM PTFs causing AS2 transaction errors or services that have been previously executing OK.

stack trace: java.lang.SecurityException: JCE cannot authenticate the provider BC
*Caused by: java.util.jar.JarException: file:/LANSA_licpgmlib/jsm/instance/jar/**jsmbcprov.jar is not signed by a trusted signer.***
at javax.crypto.a.a(Unknown Source)

The fix

There is a hotfix for LANSA integrator V14 and a hotfix for LANSA Integrator V15.

Reminder: Composer (and aXes) use Integrator services (JSM services) under the covers. So, if this issue is reported for Composer, the patch EPC142006INT_40504 can be applied, as Composer uses Integrator V14.

If this issue is reported for LANSA Integrator, the appropriate patch can be applied, depending on the version of Integrator.

Payload Http/Json now 50Mb

The payload of HTTP/Json transactions has recently been changed/increased from 8Mb to 50Mb.

You can make use of this new feature if you have installed LANSA V15 with EPC level 150040 and install hotfix HF150038:

HF150038 for CCS ID 921291: VL-WEB send data limit change from 8MB to 50 Mb

Reusable Part HTTP logging

If you are dealing with Http requests, it could be interesting, in case of solving problems, to be able to log the request & response data such as headers & body.

You can change the logging settings so the following will be logged:

- Request general information (URL details)
- Request headers
- Request body
- Response headers (not available on IBM i)
- Response body

Details how to do this can be found at:
https://docs.lansa.com/15/en/lansa018/index.htm#lansa/webserviceseng01_0395.htm

This article delivers the source of a Reusable Part (**rpHTTPlogging**) that handles this logging for you.

The use of the Reusable Part is very simple. You define the component (line 1) and activate the logging (line 2):

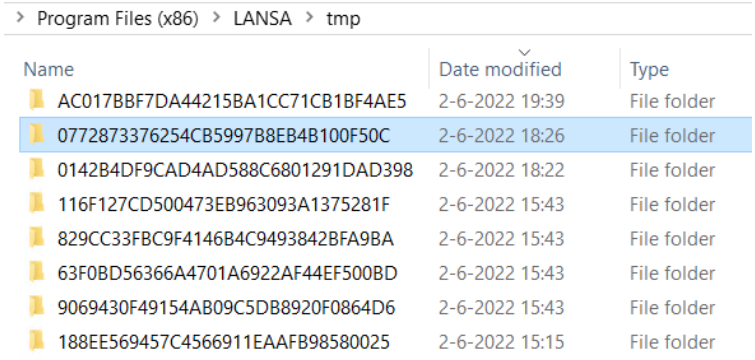
```
Define_Com Class(#rpHttpLogging) Name(#RPHTTPLOGGING)
#RPHTTPLOGGING.mSetLogging( TRUE )
```

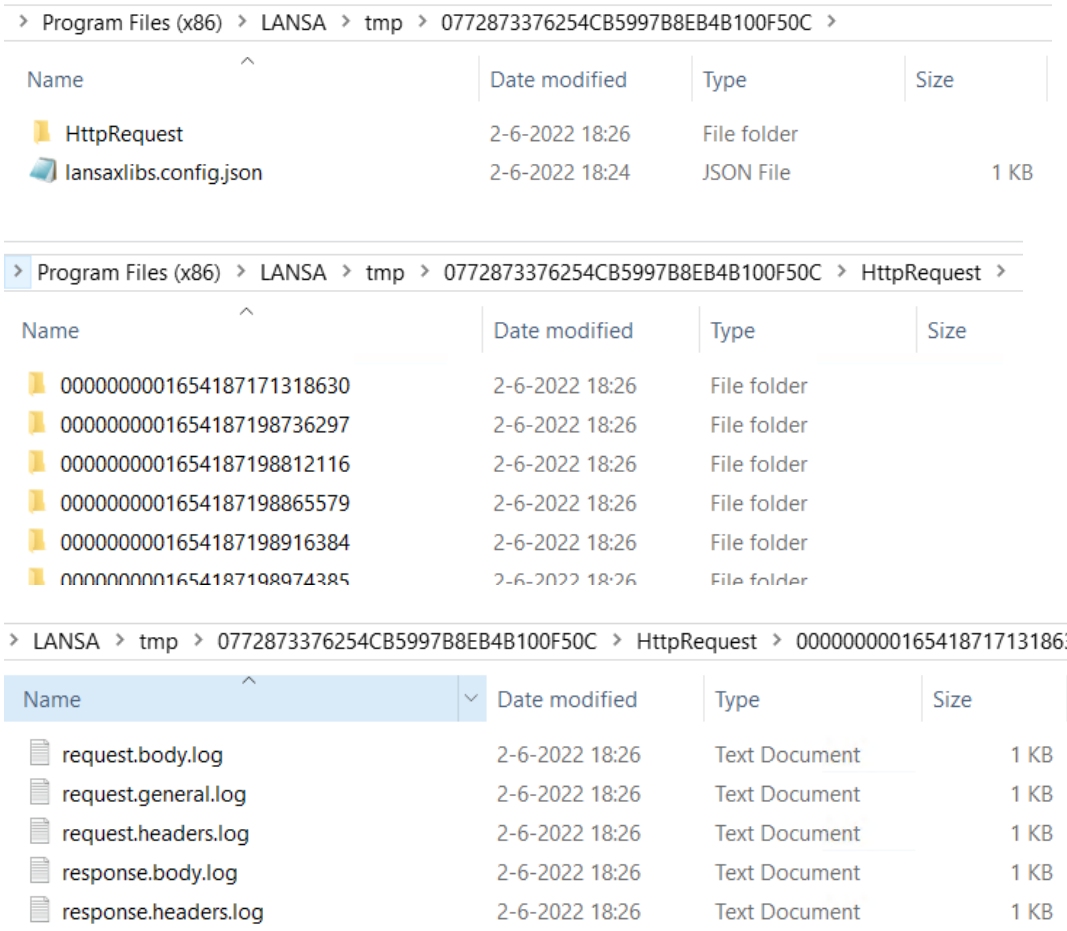
The article on the LANSA website related to http logging describes that to activate logging, you should create a config file. The Reusable Part in this article creates this file dynamic for you (in Windows or on the iSeries).

The logging in Windows will be generated in the tmp directory of the related LANSA install (for example C:\lansa\tmp). On the iSeries it will be generated on the IFS (for example /LANSA_dcxpgmlib/x_lansa/tmp/). The Reusable Part uses the LANSA system variable *TEMP_DIR to do this.

If logging is set to TRUE, it will create a lot of logging, so if you do not need it anymore, set logging to FALSE.

The Reusable Part logging creates:





The source of Reusable Part rpHTTPlogging:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_OBJT)

Mthroutine Name(mSetLogging)
Define_Map For(*INPUT) Class(#PRIM_BOLN) Name(#LoggingSetOn) Mandatory(False)
Define_Com Class(#XPRIM_OSUtil) Name(#OSutil)
Define_Com Class(#prim_alph) Name(#logDir)

#logDir := #COM_SELF.CreateXlibsLogFile( #LoggingSetOn )
#OSutil.SetEnvironmentVariable Name('LANSA_XLIB_CONFIG') Value((#LogDir +
'/lansaxlibs.config.json'))
Endroutine

Mthroutine Name(CreateXlibsLogFile) Access(*PRIVATE)
Define_Map For(*INPUT) Class(#PRIM_BOLN) Name(#LogValues) Mandatory(True)
Define_Map For(*RESULT) Class(#PRIM_ALPH) Name(#Result)

Define_Com Class(#PRIM_IOC.FileStream) Name(#FileStream)
Define_Com Class(#PRIM_IOC.StreamWriter) Name(#StreamWriter) Stream(#FileStream)
Define_Com Class(#PRIM_JSON.Writer) Name(#JsonWriter) Textwriter(#StreamWriter)

Define_Com Class(#XPRIM_Directory) Name(#Dir)
Define_Com Class(#XPRIM_File) Name(#OutFile)

Define_Com Class(#PRIM_ALPH) Name(#TmpDirOut)
Define_Com Class(#PRIM_ALPH) Name(#Guid)
```

```
#Guid := *GUID
#Dir.SetPath Directory(*TEMP_DIR)

If (#LogValues)

#Dir.AppendPath Path(#Guid)

Endif

#OutFile.SetPath Directory(#Dir) Filepath('lansaxlibs.config.json')
#OutFile.EnsureFileDirectoryExists

If (#OutFile.Exists)

#OutFile.Delete

Endif

#FileStream.Path := #OutFile.Path
#FileStream.FileMode := Create
#FileStream.FileAccess := Write

#TmpDirOut := #Dir.Path
#TmpDirOut := #TmpDirOut.ReplaceAll( (92).AsUnicodeString '/' )

#JsonWriter.BeginObject
#JsonWriter.BeginObject Membername("Common")
#JsonWriter.WriteString Membername("logOutputDir") Value(#TmpDirOut)
#JsonWriter.EndObject
#JsonWriter.BeginObject Membername("HttpRequest")
#JsonWriter.BeginObject Membername("logging")
#JsonWriter.WriteBoolean Membername("requestHeaders") Value(#LogValues)
#JsonWriter.WriteBoolean Membername("requestBody") Value(#LogValues)
#JsonWriter.WriteBoolean Membername("responseHeaders") Value(#LogValues)
#JsonWriter.WriteBoolean Membername("responseBody") Value(#LogValues)
#JsonWriter.EndObject
#JsonWriter.EndObject
#JsonWriter.EndObject

#JsonWriter.Close

#Result := #TmpDirOut

Endroutine

End_Com
```