# **LANSA Open System Utilities Guide**

- About this Guide
- Before you start
- Installing the Open System Utilities
- Available Open System Utilities
- Direct Calling LANSA Processes and Functions

Edition Date February 14, 2014 © LANSA

#### **About this Guide**

This guide contains technical information about the LANSA Open Systems Utilities which allow you to perform powerful global manipulations of a LANSA system outside many of the normal internal LANSA checks.

You should only use these utilities if you have a sound knowledge of LANSA and its internal database.

**Note:** These facilities support LANSA for iSeriesV11.00 or higher only.

#### **Disclaimer**

The information contained in this guide represents the current view of LANSA as of the date of publication. Because LANSA must respond to changing market conditions, it should not be interpreted to be a commitment on the part of LANSA, and LANSA cannot guarantee the accuracy of any information presented after the date of publication.

All information is for informational purposes only. Every effort has been made to insure that the information in this material is accurate, however, in no event shall LANSA be liable for any damages arising from its use.

LANSA MAKES NO WARRANTIES, EXPRESSED OR IMPLIED.

#### **Abbreviations**

Abbreviations used in this document:

#### **Abbreviation Description**

<<p><<p>Possible
Name of LANSA program library. Often DC@PGMLIB, but can be changed during installation. Refer to the *Installing* 

LANSA on iSeries Guide for details.

<<dtalib>> Name of LANSA data library. Often DC@DTALIB, but can be

changed during installation. Refer to the *Installing LANSA* on

iSeries Guide for details.

<<osulib>> Name of LANSA open systems utility library. Often

DC@OPENLIB, but can be changed during installation.

IFS The iSeries Integrated File System.

jar Java archive. Java programs are shipped in jar files which are

installed on the IFS.

OSU Open System Utilities

#### 1. Before You Start

The Open System Utilities facility is a library of utilities that allow experienced developers to manipulate their LANSA for iSeries environment in a less controlled, but more powerful, manner. They enable global or system level manipulations of a LANSA for iSeries environment.

Before you attempt to install and use the Open System Utilities, it is important that you read the remainder of this guide to find out about actually using these utilities and the special warnings that apply.

Before you start using the Open System Utilities, be sure to review:

- 1.1 Conditions of Use
- 1.2 Integrity
- 1.3 The Golden Rule Backups

#### 1.1 Conditions of Use

The Open System Utilities library is provided to you on the following basis:

- That you have purchased a permanent LANSA Development License for at least one machine in your organization.
- Open System Utilities material may be copied for backup and recovery purposes. However, material must not be copied for purposes of providing to any other organization. This prohibition applies to documentation and computer definition, and/or execution objects.
- That you understand that you (the user of any Open System Utilities feature) are solely responsible for any/all results that it produces, regardless of whether or not the LANSA system involved belongs to the organization that employs you, or to any other organization.
- Open System Utilities allow you to perform powerful global manipulations of a LANSA system outside many of the normal internal LANSA checks. As such, you are solely responsible for any/all results produced. You must exercise due skill, care and caution when using Open System Utilities.
- Open System Utilities are provided on an "as is" basis. No warranty is expressed or implied in the provision of these Open System Utilities.
- You are solely responsible for the installation, maintenance, backup and recovery of all Open System Utilities provided.
- You are solely responsible for ensuring that any version of the Open System Utilities that you use is compatible with the version of LANSA that you are using them against. Software is provided to check on this, but you are ultimately responsible for ensuring the versions are compatible.

### 1.2 Integrity

The self-contained LANSA for iSeries development environment has many facilities to protect its integrity. In some situations, at a global or system level, these integrity checks may appear to make the job of manipulating a LANSA for iSeries system harder or more complex than it needs to be. However, they are necessary in order to maintain the stability of the system.

#### Note that:

- Open System Utilities are provided on an "as is" basis with complete source code and documentation.
- They provide you with extensive knowledge of the internals of your LANSA for iSeries environment, allowing you to extend the shipped utilities or to develop new ones.
- The type of manipulations the Open System Utilities perform require skill and care to be exercised whenever they are used.
- No warranty is expressed or implied in the provision of these utilities.

Some of the facilities provided by these utilities include:

- High speed movement of complete partitions between environments and machines.
- Stripping down of non-development systems (i.e. to run-only) to minimal size.
- Change LANSA for iSeries program and/or data library names.

The number of these utilities is continually growing.

# 1.3 The Golden Rule - Backups



When using the Open System Utilities, always remember to:

# Run Backups FIRST

# 2. Installing the Open System Utilities

You will find the instructions for installing the Open System Utilities or updating the Open System Utilities in Task: Upgrade an Existing LANSA iSeries System in the *Installing LANSA* on *iSeries Guide*.

#### When you install the Open System Utilities you will get:

- Source statements for all objects shipped in the library DC@OPENLIB.
- Messages used by OSU shipped in the library DC@OPENLIB.
- 2.1 Directory of LANSA Objects Affected by OSU
- 2.2 Supported LANSA PC Levels

#### **Supplied for each Utility is:**

- A Function Description
- Name, Type and Source
- Files Used
- Warnings That Apply
- Parameters
- Examples

Be sure to review these details carefully before using the Utility.

# 2.1 Directory of LANSA Objects Affected by OSU

**File Name:** OS@F01

**Description:** Directory of LANSA Objects affected by OSU

Implemented in: 4.00 / D3

**Normal Library:** <<osulib>>

# **Logical Views**

Name	Use	Keys/Description/Comments
OS@F01V1	Read	O01OBJ, O01TYP.
OS@F01V2	Update	Same as DC@F01V1.

# **Record Layout**

Field Name	Type	Len	Dec	<b>Description / Comments / Values</b>
O01OBJ	A	10		Object name
O01TYP	A	8		Object type
O01NLB	A	1		Normal library (P=Pgm, D=Data)
O01RT1	A	1		Run and Compile system
O01RT2	A	1		Run only system
O01DEV	A	1		Development

# 2.2 Supported LANSA PC Levels

**Object Name:** OS@A01

**Description:** Supported LANSA PC Levels

**Implemented in:** 4.00 / D9

**Normal Library:** <<osulib>>

# **Record Layout**

Data Area	Type	Len	Dec	Description
OS@A01	A	20		Supported PC Levels

# 3. Available Open System Utilities

Library (as shipped)	DC@OPENLIB
Available machine types	iSeries
Owner of objects (as shipped)	QOTHPRDOWN
Contents (as shipped)	See table below

Object	Type	Description
QOPNSRC	*SRCPF	Source statement for all objects.
OS@M01	*MSGF	Messages used by OSU.
OS@F01	*PF	Directory of LANSA Objects affected by OSU
OS@A01	*DTAARA	Supported LANSA PC Levels
OS@P0001	*CLLE	3.1 High Speed Partition Export to Tape
OS@P0002	*CLLE	3.2 High Speed Partition Import from Tape
OS@P0003	*CLLE	3.3 High Speed Partition Deletion
OS@P0004	*CL	3.4 Remove Observability from All Objects in a Library
OS@P0005	*CL	3.5 Strip System to Import, Compile and Run Only
OS@P0006	*CL	3.6 Strip System to Import and Run Only
OS@P0007	*CL	3.7 Rename LANSA Libraries
OS@P0008	*CL	3.8 Rename Partition Libraries
OS@P0009	*CL	3.9 Restore a Process from Backup Media
OS@P0010	*CL	3.10 Restore a Function's Definition from Backup Media
OSXP0012	*CLE	3.12 LANSA IFS Directory Root Rename

OSXP0013	*CLE	3.13 Create/Recreate Java Programs in LANSA Classpath
OS@P8001	*CL	3.14 Lock and/or Unlock a Complete LANSA System
OS@P8002	*CL	3.15 Retrieve System Details from DC@A01 Data Area
OS@P8003	*CL	3.16 Retrieve Partition Details from DC@F46
OS@P8004	*CL	3.17 Set Library List for Initial System Access
OS@P8005	*CL	3.18 Set Library List for Partition Access
OS@P8006	*CL	3.19 Copy Records for One Partition Only
OS@P8007	*CL	3.20 Copy Records for All Partitions
OS@P8008	*CL	3.21 Check Release and PC Level Compatibility
OS@P8009	*CL	3.22 Copy Records for a Process
OS@P8010	*CL	3.23 Build List of Help Pointers for a Process
OS@P8011	*CL	3.24 Build List of Help Pointers for a Function
OS@P8012	*RPG	3.25 Update a Partition's Libraries
OS@P8013	*CL	3.26 Copy Selected Records
OS@P8014	*RPG	3.27 Update Security File Partition Libraries
OS@P8015	*CL	3.28 Copy All Records
OS@P8016	*RPGIV	3.29 Re-assign or Zero RDML Audit Tokens
OSXS802X	*SRVPGM	3.30 Service Program for OSXP802X Series

#### 3.1 High Speed Partition Export to Tape

# **Functional Description**

Library QTEMP is cleared.

This program receives the identifier of a partition and copies all required data from the LANSA data library to temporary files in library QTEMP.

When all data has been copied to QTEMP a merged list of the following is created. Partition module library, partition default file library, and all other specified libraries. The merged list is then stored in a data area in QTEMP as a "directory" of what should be found on the export tape. This data area is called OS@DIR.

If the partition has IFS objects, the IFS objects are first saved into a save file named IFS\_SAVF in QTEMP. This save file is then saved together with all the other objects copied to QTEMP.

All objects from QTEMP are saved to the tape device specified, followed by all libraries in the resulting merged list. Once this operation has completed a message is sent to the system operator asking whether a repeat of all save operations is required (i.e., Do you want another copy?). This allows multiple save tapes to be easily prepared from one export run.

Name	OS@P0001
Type	ILE Control Language Program
Source provided	Yes

### **Files Used**

All files in the <<dtalib>> are used by this job.

# Warnings That Apply

- Remember The Golden Rule.
- You must back up <<dtalib>> before using this utility.
- <<osulib>> must be in the library list when running this job.
- The partition specified must exist.
- This routine does not do any security checking.

- No one should be using the partition at the time of export.
- Tapes are not initialized by the export operation.
- This program is best run in batch.
- If the target release is anything other than \*CURRENT, ensure that all objects being saved have been created so that they are compatible with this target release.

#### **Parameters**

No	Type	Len	Dec	Description
1	Alpha	3		Identifier of partition to be exported.
2	Alpha	10		Name of tape device to be used for export.
3	Alpha	8		Indicates the TGTRLS that the save will use.  Valid values include:  "*CURRENT"  "*PRV"  "*VxRyMz"  where "VxRyMz" is a valid OS/400 version.  If left blank, the target release used is *CURRENT, not the default for the command.
4	Packed	15	5	Number of extra libraries to be saved.
5	Alpha	200		List of 20 x alpha(10) extra libraries. This list and the partition module library and default file library are merged to form the final list of libraries to be saved.

# **Examples**

Export partition DEM to TAP01 for current target release and no other libraries.

# CALL OS@P0001 (DEM TAP01 '\*CURRENT' 0 ' ')

Export partition DEM to TAP01 for previous target release and also library <<comlib>>.

# CALL OS@P0001 (DEM TAP01 '\*PRV' 1 <<comlib>>)

Export partition DEM to TAP01 for target release V3R7M0 and also libraries <<comlib>> and TEST.

CALL OS@P0001 (DEM TAP01 'V3R7M0' 2 '<<comlib>> TEST where six spaces follow <<comlib>> and TEST.

### 3.2 High Speed Partition Import from Tape

### **Functional Description**

An attempt to exclusively lock the LANSA system will be made by this job. If unsuccessful, the job will terminate at this point.

All objects saved in QTEMP are restored from the specified tape device to library QTEMP.

All libraries in OS@DIR are restored. (See OS@P0001 Technical Description for details regarding OS@DIR).

This program then copies the data in QTEMP to the LANSA data library, effectively recreating the previously deleted (and saved) partition.

If IFS objects were saved during the 3.1 High Speed Partition Export to Tape, the IFS objects are restored into the IFS as well.

Name	OS@P0002
Type	ILE Control Language Program
Source provided	Yes

#### **Files Used**

All files in the <<dtalib>> are used by this job.

# **Warnings That Apply**

- Remember The Golden Rule.
- You must backup <<dtalib>> before using this utility.
- <<osulib>> must be in the library list when running this job.
- The partition specified must not already exist in the target system.
- Existing partitions must have different identifiers to the one about to be restored.
- Partition to be imported must not have the same unique prefix as an existing partition (even if the partition identifiers are different).
- This routine does not do any security checking.
- No one should be using the system at the time of import.

• This program is best run in batch.

# **Parameters**

ľ	No	Type	Len	Dec	Description
1		Alpha	10		Name of tape device to be used for import.

# **Examples**

Import previously saved partition from TAP01.

**CALL OS@P0002 (TAP01)** 

#### 3.3 High Speed Partition Deletion

## **Functional Description**

An attempt to exclusively lock the LANSA system will be made by this job. If unsuccessful the job will terminate at this point.

Job will end if the partition to be deleted is the SYS partition.

All data, excluding the data for the partition specified, from the LANSA data library is copied to temporary files in library QTEMP.

All data from QTEMP is then copied back to the <<dtalib>>, effectively removing the specified partition from the LANSA system.

LANSA system will be released.

Name	OS@P0003
Type	ILE Control Language Program
Source provided	Yes

#### **Files Used**

All files in the <<dtalib>> are used by this job.

# **Warnings That Apply**

- Remember The Golden Rule.
- You must backup <<dtalib>> before using this utility.
- <<pgmlib>> and <<osulib>> must be in the library list when running this job.
- Not to be used on partition SYS (this is checked).
- The partition specified must exist.
- This routine does not do any security checking.
- No one should be using the system at the time of deletion.
- This program is best run in batch.
- Program failure will leave system in a corrupted state. To recover from corrupted state restore <<dtalib>>.
- If the library is not used by another partition, it must be deleted after this

process has been completed.

# **Parameters**

No	Type	Len	Dec	Description
1	Alpha	3		Identifier of partition to be deleted.

# **Examples**

Delete partition DEM.

CALL OS@P0003 (DEM)

# 3.4 Remove Observability from All Objects in a Library

# **Functional Description**

All programs in the specified library will have their symbolic debug table removed. This table can account for up to 60% of a program's object size. For more information on this topic refer to the appropriate IBM manual.

Name	OS@P0004
Type	Control Language Program
Source provided	Yes

### **Warnings That Apply**

- Remember The Golden Rule.
- <<osulib>> must be in the library list when running this job.
- The library specified must exist.
- If you want to debug any programs in this library after this job has completed they will need to be recompiled.
- This utility can be run repeatedly with no problems, even over libraries already stripped.
- This job can be run on any library in the system.

#### **Parameters**

No	Type	Len	Dec	Description
1	Alpha	10		Name of library to be stripped.

#### **Examples**

Remove observability from objects in library TEST.

CALL OS@P0004 (TEST)

#### 3.5 Strip System to Import, Compile and Run Only

## **Functional Description**

This program reads the file OS@F01 and according to the flags set in this file for each object, decides whether or not the object is needed for a run and compile system. If not the object is removed from the system, unless the object is a file (unused files are cleared).

As a last step in reducing disk usage OS@P0004 is run over <<pre>pgmlib>> to
reduce the resulting run time system to its minimum size.

The resulting system can only do the following:

- Import files, fields, processes and functions
- Compile processes, functions, and I/O modules
- Execute processes and functions

Name	OS@P0005
Type	Control Language Program
Source provided	Yes

#### **Files Used**

OS@F01 Directory of LANSA Objects affected by OSU

# **Warnings That Apply**

- Remember The Golden Rule.
- Backup the entire LANSA system first.
- <<osulib>> must be in the library list when running this job.
- Run a LANSA REORG before using this utility.
- Once completed the system will be available for import, run and compile only. No other functionality.
- No one should be using the system at time of run.
- This routine does not do any security checking.

#### **Parameters**

No	Type	Len	Dec	Description	
1	Alpha	1		Delete multilingual components (Y/N)	

# **Examples**

Strip system to run, import and compile only, but with multilingual components still available

**CALL OS@P0005 (N)** 

### 3.6 Strip System to Import and Run Only

### **Functional Description**

This program reads the file OS@F01 and according to the flags set in this file for each object, decides whether or not the object is needed for a run only system. If not then object is removed from the system, unless the object is a file (unused files are cleared).

As a last step in reducing disk usage OS@P0004 is run over <<pre>pgmlib>> to
reduce the resulting run time system to its minimum size.

The resulting system can only do the following. . .

- Import files, fields, processes and functions
- Execute processes and functions

Name	OS@P0006
Type	Control Language Program
Source provided	Yes

#### **Files Used**

OS@F01 - Directory of LANSA Objects affected by OSU

# **Warnings That Apply**

- Remember The Golden Rule.
- Backup the entire LANSA system first.
- <<osulib>> must be in the library list when running this job.
- Run a LANSA REORG before using this utility.
- Once completed the system will be available for import and run only. No other functionality.
- No one should be using the system at time of run.

#### **Parameters**

No	Type	Len	Dec	Description

1	Alpha 1		Delete multilingual components (Y/N)
---	---------	--	--------------------------------------

# **Examples**

Strip system to run only with no multilingual components **CALL OS@P0006 (Y)** 

#### 3.7 Rename LANSA Libraries

### **Functional Description**

The LANSA system is locked exclusively.

The <<pgmlib>> and <<dtalib>> are renamed to the specified names.

The system definition data area (DC@A01) is updated to reflect the new names for the <<pre>regmlib>> and <<dtalib>>.

All the LANSA commands are then changed to point to the correct <<pre>cqpmlib>>.

The LANSA system is released.

Name	OS@P0007
Type	Control Language Program
Source provided.	Yes

#### **Files Used**

DC@W12 LANSA System Tables

### Warnings That Apply

- Remember The Golden Rule.
- Backup the system first
- <<osulib>> must be in the library list when running this job.
- The LANSA libraries must not be in the library list of any other job on the system.
- No one should be using the system.
- If the LANSA IFS root directory name is based on the LANSA program library name (which is the default) then you need to run OSU program OSXP0012 to rename the LANSA IFS root directory and update references to the LANSA IFS root directory within LANSA.
- Any software that invokes this LANSA system may need to be changed.
- Any software that uses the old library names may need to be changed.

# **Parameters**

No	Type	Len	Dec	Description
1	Alpha	10		New < <pgmlib>&gt;</pgmlib>
2	Alpha	10		New < <dtalib>&gt;</dtalib>

# **Examples**

Rename the LANSA libraries to LANPGMPROD and LANDTAPROD CALL OS@P0007 (LANPGMPROD LANDTAPROD)

#### 3.8 Rename Partition Libraries

### **Functional Description**

This program receives two library names. The first name is the old partition data or module library, the second is the new name for this library.

Any occurrence of the old library specified in the files mentioned above, are changed to the new library.

Name	OS@P0008
Type	Control Language Program
Source provided	Yes

#### **Files Used**

DC@F02 System Security

DC@F05 Validation Check Directory

DC@F12 File Definition

DC@F14 File Definition List

DC@F15 Logical File Definition

DC@F17 Foreign Key Check Definition

DC@F18 Access Route Definition

DC@F19 Batch Control Logic Definition

DC@F21 Module File Usage

DC@F27 File Version Definition

DC@F45 Virtual Field RPGIII Code File

DC@F46 Partition Identifier

DC@F64 Multilingual Extension of DC@F27

DC@F65 Multilingual Extension of DC@F15

DC@F74 TTS Object Register

DC@F75 TTS Object Event Log

### **Warnings That Apply**

- Remember The Golden Rule.
- <<osulib>> must be in the library list when running this job.
- No one should be using the system at time of run.
- This routine does not do any security checking.
- This routine does not rename the actual library, only the references to it in LANSA internal database.
- Renames any occurrences of this library name in the LANSA internal database.

#### **Parameters**

No	Type	Len	Dec	Description
1	Alpha	10		Old partition library
2	Alpha	10		New partition library

# **Examples**

Rename partition library OLDPARTLIB to NEWPARTLIB

CALL OS@P0008 (OLDPARTLIB NEWPARTLIB)

Rename partition library DEVLANSA to DEVLANDATA

CALL OS@P0008 (DEVLANSA DEVLANDATA)

Rename partition library DEVLIB to PRODLIB

CALL OS@P0008 (DEVLIB PRODLIB)

# 3.9 Restore a Process from Backup Media

### **Functional Description**

The files listed below are restored into temporary files in QTEMP from the specified backup media.

The records relating to the specified process, in the specified partition, are copied from the temporary files in library QTEMP into the corresponding LANSA data library files.

This effectively redefines the process to LANSA.

**Note:** Only the definition is restored. The process (and its associated functions) will need to be recompiled before being used.

Name	OS@P0009
Type	Control Language Program
Source provided	Yes

#### **Files Used**

DC@F02 System Security

DC@F04 Help Text Storage

DC@F20 Module Definition

DC@F21 Module File Usage

DC@F23 Module Format Definition

DC@F25 Process Field Usage

DC@F29 Function Control Commands

DC@F30 Function Control Parameters

DC@F31 Process Additional Menu Options

DC@F32 Process Parameter Definition

DC@F40 Process / Function Field Definition

DC@F44 Process Attachments

DC@F63 Multilingual Extension of DC@F04

DC@F66 Multilingual Extension of DC@F20

DC@F67 Multilingual Extension of DC@F23

DC@F68 Multilingual Extension of DC@F31

DC@F69 Action Bar Definition

DC@F71 Action Bar Multilingual Definition

DC@F85 Miscellaneous Function Details

### **Warnings That Apply**

- Remember The Golden Rule.
- <<osulib>> must be in the library list when running this job.
- The partition specified must exist.
- The process specified must not exist.
- This routine does not do any security checking.
- Job best run interactively.
- Definition only not compiled objects.

#### **Parameters**

No	Type	Len	Dec	Description
1	Alpha	4		Type of backup media (TAPE, SAVF).
2	Alpha	10		Name of device / save file name.
3	Alpha	10		Save file library.
4	Alpha	3		Name of partition where process resides.
5	Alpha	10		Name of process to be restored.

# **Examples**

Restore process TEST01 from partition DEM from tape device TAP01:

CALL OS@P0009 (TAPE TAP01 ' ' DEM TEST01)

Restore process TEST01 from partition DEM from save file SAVEDATA in library <<comlib>>:

CALL OS@P0009 (SAVF SAVEDATA <<comlib>> DEM TEST01)

# 3.10 Restore a Function's Definition from Backup Media Functional Description

The files listed below are restored into temporary files in QTEMP from the specified backup media.

The records relating to the user specified function, in the specified process and partition, are copied from the temporary files in library QTEMP into the corresponding LANSA data library files.

This effectively redefines the function to LANSA.

**Note:** Only the definition is restored. The function will need to be recompiled before it can be executed.

Name	OS@P0010
Type	Control Language Program
Source provided	Yes

#### Files Used

DC@F02 System Security

DC@F04 Help Text Storage

DC@F20 Module Definition

DC@F21 Module File Usage

DC@F23 Module Format Definition

DC@F25 Process Field Usage

DC@F29 Function Control Commands

DC@F30 Function Control Parameters

DC@F40 Process / Function Field Definition

DC@F63 Multilingual Extension of DC@F04

DC@F66 Multilingual Extension of DC@F20

DC@F67 Multilingual Extension of DC@F23

DC@F85 Miscellaneous Function Details

### Warnings That Apply

- Remember The Golden Rule.
- <<osulib>> must be in the library list when running this job.
- The partition specified must exist.
- The process specified must exist.
- The function specified must not exist.
- This routine does not do any security checking.
- Job best run interactively.
- Definition only not compiled objects.

#### **Parameters**

No	Type	Len	Dec	Description
1	Alpha	4		Type of backup media (TAPE, SAVF).
2	Alpha	10		Name of device / save file name.
3	Alpha	10		Save file library.
4	Alpha	3		Name of partition where process resides.
5	Alpha	10		Name of process to be restored.
6	Alpha	7		Name of function to be restored.

### **Examples**

Restore function FUNC001 from process TEST01 from partition DEM from tape device TAP01

# CALL OS@P0010 (TAPE TAP01 ' ' DEM TEST01 FUNC001)

Restore function FUNC002 from process TEST01 from partition DEM from save file SAVEDATA in library <<comlib>>

# CALL OS@P0010 (SAVF SAVEDATA << comlib>> DEM TEST01 FUN

### 3.11 High Speed IFS Partition Directory Deletion

# **Functional Description**

This program is no longer used.

This program removes the LANSA partition's IFS directory after removing all its contents.

This program is called from OSU program OS@P0003 3.3 High Speed Partition Deletion so you would normally not call this program directly unless you want to only delete the partition's IFS contents.

Name	OSXP0011
Type	ILE C Program
Source provided	Yes

# **Warnings That Apply**

- <<osulib>> and <<pgmlib>> must be in the library list when running this job.
- The partition specified must exist.
- No one should be using the system at the time of deletion.
- This program is best run in batch.
- Program failure will leave system (IFS contents) in a corrupted state.
- To recover from corrupted state (abnormal termination):
  - Review diagnostic messages issued by this program.
  - Correct the source of error.
  - Rerun this program.
- Don't bypass the PC compatibility test when calling this program directly.

#### **Parameters**

1	No	Type	Len	Dec	Description
1		Alpha	3		Identifier of partition to be deleted.
2	2	Alpha	8		Optional "*NOCHECK" to bypass PC compatibility

test. Should not be used if called directly.	
--	--

# **Examples**

Delete the IFS partition directory for partition WRK CALL OSXP0011 ('WRK')

## 3.12 LANSA IFS Directory Root Rename

## **Functional Description**

The LANSA system is locked exclusively.

The new LANSA root directory is created.

The new directory name is set in data area DC@IFS in <<pre>pgmlib>>

The OS/400 MOV command is used to move the directory tree from the old LANSA root directory into the new LANSA root directory.

The old LANSA root directory is removed.

The Java classpath stored in data area LJAVACP in <<pre>pgmlib>> is updated to reflect the changes in the IFS directory name.

The LANSA system is then released.

**Note:** This program is not called from OS@P0007 3.7 Rename LANSA Libraries because the IFS root directory name may not match the program library name.

Name	OSXP0012
Type	ILE C Program
Source provided	Yes

## Warnings That Apply

- <<osulib>> and <<pgmlib>> must be in the library list when running this job.
- Backup the existing LANSA directory tree before running this program and save data areas DC@IFS and LJAVACP from <<ppgmlib>>.
- You must signon with a user profile that has authority to remove the LANSA IFS directory and its contents.
- No one should be using the LANSA system at the time of rename.
- This program is best run in batch.
- Program failure will leave system in a corrupted state.

- To recover from corrupted state (abnormal termination): Restore the saved IFS directory.
   Restore the saved data areas DC@IFS and LJAVACP.
- Don't bypass the PC compatibility test when calling this program directly.

## **Parameters**

No	Type	Len	Dec	Description
1	Alpha	Vary		New LANSA IFS root directory
2	Alpha	8		Optional "*NOCHECK" to bypass PC compatibility test. Should not be used if called directly.

# **Examples**

Rename the LANSA IFS directory to '/newifsdir'

CALL OSXP0012 ('/newifsdir')

### 3.13 Create/Recreate Java Programs in LANSA Classpath

## **Functional Description**

The LANSA system is locked exclusively.

If the first input argument to this program is 'N', it creates the Java programs for each Java jar file defined in the LANSA classpath that is located under the LANSA IFS root directory.

Conversely, if the first input argument is 'Y' then it creates the Java programs for all Java jar files defined in the LANSA classpath, regardless of its location on the IFS.

The LANSA system is then released.

Name	OSXP0013
Type	ILE C Program
Source provided	Yes

## **Warnings That Apply**

- <<osulib>> and <<pgmlib>> must be in the library list when running this job.
- The command to create Java programs (CRTJVAPGM) is resource intensive and can be long running (depending on the number and size of the Java classes included in the jar file). This program is best run in batch and outside production hours.
- You must signon with the user profile that you want to own the Java programs.
- Program failure will leave system in a corrupted state.
- To recover from corrupted state (abnormal termination): Review diagnostic messages issued by this program.
   Correct the source of error.
   Rerun this program.
- Don't bypass the PC compatibility test when calling this program directly.

No	Type	Len	Dec	Description
1	Alpha	1		'N' - Create/recreate Java programs defined in the LANSA Java classpath that are located under the LANSA IFS root directory.
				'Y' - Create/recreate all Java programs defined in the LANSA Java classpath.
2	Alpha	8		Optional "*NOCHECK" to bypass PC compatibility test. Should not be used if called directly.

# **Examples**

Recreate the Java programs defined in the LANSA Java classpath that are located under the LANSA IFS root directory

**CALL OSXP0013 ('N')** 

# 3.14 Lock and/or Unlock a Complete LANSA System

## **Functional Description**

This program receives a lock or unlock request. If the request is 'LOCK', an exclusive lock is placed on the data area DC@A01 in <<pre>pgmlib>>. If the request is 'FREE', then the data area DC@A01 is de-allocated.

Name	OS@P8001
Type	Control Language Program
Source provided	Yes

# **Warnings That Apply**

- The <<pgmlib>> must be in the library list
- No one should be using the system at time of 'LOCK' request.
- Once the system is locked, no LANSA jobs will execute successfully.
- Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	4		'LOCK' or 'FREE' the LANSA system

# 3.15 Retrieve System Details from DC@A01 Data Area

## **Functional Description**

Retrieves the following information from the system definition data area.

- <<pgmlib>>
- <<dtalib>>
- Current Release
- Current PC Level

These values are returned to the calling program.

Name	OS@P8002
Type	Control Language Program
Source provided	Yes

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		< <pgmlib>&gt;</pgmlib>
2	Alpha	10		< <dtalib>&gt;</dtalib>
3	Alpha	3		Current Release
4	Alpha	2		PC Level

# 3.16 Retrieve Partition Details from DC@F46

# **Functional Description**

Retrieves the passed partition's record from DC@F46 to return the partition data library and module library.

Name	OS@P8003
Type	Control Language Program
Source provided	Yes

## **Files Used**

DC@F46 Partition Identifier

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	3		Partition Identifier
2	Alpha	10		Partition data library
3	Alpha	10		Partition module library

# 3.17 Set Library List for Initial System Access

# **Functional Description**

Sets the <<pgmlib>> and the <<dtalib>> to be highest libraries in the library list.

Name	OS@P8004
Type	Control Language Program
Source provided	Yes

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		< <pgmlib>&gt;</pgmlib>
2	Alpha	10		< <dtalib>&gt;</dtalib>

# 3.18 Set Library List for Partition Access

# **Functional Description**

Includes the partition data and module libraries into the library list.

Name	OS@P8005
Type	Control Language Program
Source provided	Yes

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		Partition data library
2	Alpha	10		Partition module library

# **3.19 Copy Records for One Partition Only**

# **Functional Description**

Copies <<dtalib>> records to temporary files in QTEMP.

Name	OS@P8006
Type	Control Language Program
Source provided	Yes

## **Files Used**

All files in the <<dtalib>> are used by this job.

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	3		Partition Identifier
2	Alpha	10		File name
3	Alpha	10		Field name
4	Alpha	10		Source library
5	Alpha	10		Target library
6	Alpha	3		Equal or not equal
7	Alpha	8		Member option
8	Alpha	4		Create file

# 3.20 Copy Records for All Partitions

# **Functional Description**

Copies the records for all partitions to temporary files in QTEMP.

Name	OS@P8007
Type	Control Language Program
Source provided	Yes

## **Files Used**

All files in the <<dtalib>> are used by this job.

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		File name
2	Alpha	10		Source library
3	Alpha	10		Target library
4	Alpha	8		Member option
5	Alpha	4		Create file

# 3.21 Check Release and PC Level Compatibility

## **Functional Description**

Compares the LANSA system release and PC level to the levels supported by the Open Systems Utilities. These levels are stored in data area OS@A01 in <<osulb>>.

If the levels do not correspond, OSU will send a message to the system operator stating that OSU and LANSA are incompatible.

The user then has the option to continue or end the job. Unless you are absolutely sure that it is safe to continue with the job even though the levels are incompatible, end the job at this point.

Name	OS@P8008
Type	Control Language Program
Source provided	Yes

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	3		Release Level
2	Alpha	2		PC Level

# 3.22 Copy Records for a Process

# **Functional Description**

This routine is passed all Parameters specifying the copy criteria. That is, file name, source library, target library, where parameter (INCREL) details.

These parameters are used to build a specific copy command that is used for copying process details.

Name	OS@P8009
Type	Control Language Program
Source provided	Yes

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		File name
2	Alpha	10		Source Library
3	Alpha	10		Target Library
4	Alpha	256		Where parameter of copy (INCREL)

# 3.23 Build List of Help Pointers for a Process

# **Functional Description**

This routine builds a list of up to 50 help pointers for a process. This is done by processing the file DC@F20.

Name	OS@P8010
Type	Control Language Program
Source provided	Yes

#### **Files Used**

DC@F20 Module definition

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		Process
2	Alpha	3		Partition ID
3	Alpha	500		List of help pointers

# 3.24 Build List of Help Pointers for a Function

# **Functional Description**

This routine builds a list of up to 50 help pointers for a function. This is done by processing the file DC@F23.

Name	OS@P8011
Type	Control Language Program
Source provided	Yes

#### **Files Used**

DC@F23 Module format Definition

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		Process
2	Alpha	7		Function
3	Alpha	3		Partition ID
4	Alpha	500		List of help pointers

# 3.25 Update a Partition's Libraries

# **Functional Description**

This routine reads a file sequentially. During the read it looks for a certain library name and replaces it with the new library name. The changed record is then updated.

Name	OS@P8012
Type	RPG Program
Source provided	Yes

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		Old library name
2	Alpha	10		New library name
3	Numeric	15	5	Position of library in file

# 3.26 Copy Selected Records

## **Functional Description**

This routine is passed all Parameters specifying the copy criteria. That is, source file name, source library, target file name, target library, where parameter (INCREL) details.

These parameters are used to build a specific copy command that is used for copying details.

Name	OS@P8013
Type	Control Language Program
Source Provided	Yes

#### **Files Used**

All files in the <<dtalib>> are used by this job.

## Warnings that apply

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		Source File name
2	Alpha	10		Source Library
3	Alpha	10		Target File name
4	Alpha	10		Target Library
5	Alpha	256		Where parameter of copy (INCREL)

# 3.27 Update Security File Partition Libraries

# **Functional Description**

This routine reads file DC@F02 sequentially. During the read it looks for a certain library name and replaces it with a new library name. The changed record is then updated.

Name	OS@P8014
Туре	RPG Program
Source provided	Yes

#### **Files Used**

DC@F02 System security

# **Warnings That Apply**

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		Old library name
2	Alpha	10		New library name

# 3.28 Copy All Records

# **Functional Description**

This routine is passed the Parameters specifying the copy criteria. That is, source file name, source library, target file name and target library.

These parameters are used to build a specific copy command that is used for copying a file.

Name	OS@P8015
Type	Control Language Program
Source Provided	Yes

#### **Files Used**

All files in the <<dtalib>> are used by this job.

## Warnings that apply

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		Source File name
2	Alpha	10		Source Library
3	Alpha	10		Target File name
4	Alpha	10		Target Library

## 3.29 Re-assign or Zero RDML Audit Tokens

## **Functional Description**

This routine re-assigns or zeroes RDML audit tokens. It is used in High Speed Partition Import, Restore a Process from Backup Media & Restore a Function's Definition from Backup Media. It is passed the Parameters specifying the temporary Function Control Command file and temporary Miscellaneous Function Details file, and uses program DC@P8197 to retrieve / store RDML Audit information in the Miscellaneous Function Details File in the LANSA Data library.

Name	OS@P8016
Type	RPGIV Program
Source Provided	Yes

#### **Files Used**

DC@F29 Function Control Commands

DC@F85 Miscellaneous Function Details

# Warnings that apply

• Do not call directly. This is a utility program for the OS@P00nn series programs in this OSU package.

No	Type	Len	Dec	Description
1	Alpha	10		Temporary Function Control Command File name
2	Alpha	10		Temporary Function Control Command Library
3	Alpha	10		Temporary Miscellaneous Function Details File name
4	Alpha	10		Temporary Miscellaneous Function Details Library

# 3.30 Service Program for OSXP802X Series

# **Functional Description**

This service program groups common functions and IFS access application programming interfaces used by other programs in this OSU package.

Name OSXS802X

**Type** ILE C Service Program

Source Provided Yes

# **Warnings that apply**

• Do not call directly. This service program is used by other programs in this OSU package.

## 4. Direct Calling LANSA Processes and Functions

## What Is Direct Calling?

Direct calling is the method used to execute LANSA processes and functions without using the LANSA command or program.

This means that a user defined 3GL program written in RPG, CL, COBOL, PL/1 etc. can be used to directly invoke a LANSA process or function.

## **Prerequisites**

If you use Direct Calling using these guidelines you must have a sound working knowledge of the LANSA product.

If you use this method to run LANSA applications, be very careful. Be absolutely sure of what you are doing. Do not bypass any part of these guidelines because they don't look important or seem pointless.

#### Disclaimer

All the information about directly invoking LANSA processes and functions has been tested and verified to the best of our ability. However, no guarantee or warranty is expressed or implied due to the many site defined variances that occur between LANSA installations.

The testing of applications invoked in this way is the responsibility of the user site.

## 4.1 Technical Considerations and Warnings

- Processes must be compiled. It is impossible to invoke a LANSA process using this method if the OS/400 object for the process does not exist.
- No library list manipulation is performed. The library list must be correct. That is, these libraries must be included in the library list:
- LANSA Program Library
- LANSA Data Library
- Partition module library
- Partition data library (optionally)
- Any LANSA functions that are executed using this method must be created with FUNCTION OPTIONS(\*DIRECT).
- These procedures are not available for processes and/or functions that require parameters, other than working lists or data structures.
- In the case of multilingual partitions, the user defined setup program will also need to perform any overrides for message files.
  - That is, LANSA Message file DC@M01 may need to be overridden to the appropriate version. (e.g. French may use DC@M01FRA). Refer to the *LANSA Multilingual Application Design Guide* for further information about the type of processing that is necessary.
- It is not recommended that Function Routing be used with this method of running LANSA applications. When LANSA is invoked normally by use of the LANSA command, the Function Routing Table is loaded during the LANSA initiation and is available until LANSA is terminated. If you use this method of calling LANSA applications, the Function Routing Table will not have been loaded for you.
- All steps and guidelines must be considered when using this method to execute LANSA processes or functions.
- All examples provided are in RPG/400 or Control Language. This is not a restriction. User defined programs that use this method to execute LANSA applications can be written in any language available in OS/400.
- These procedures are not available for RDMLX functions. To achieve the equivalent functionality, code a "wrapper" RDML function that uses the same FUNCTION statement and simply calls the RDMLX function. If the

original RDMLX function is using the RCV\_DS to receive a data structure, refer to Note 3 in 4.6.1 Data Structures.

# **Warnings:**

- No security checking is performed.
- The environment for the application is your responsibility. LANSA no longer has any control, your user defined 3GL program is in control.

# 4.2 Determining the i5/OS Object Name of a Process & Function Process i5/OS Object Names

The OS/400 object name of a process can be determined by displaying the contents of a data area with the same name as the process that is to be directly invoked, in the partition module library.

For example, the process to be invoked is PSLSYS. It resides in partition DEM and the partition module library is DC@DEMOLIB.

### DSPDTAARA DC@DEMOLIB/PSLSYS

This will return a screen similar to this:

#### Display Data Area

System: SYDASD25

Data area . . . . : PSLSYS

Library . . . . : DC@DEMOLIB

Type . . . . : \*CHAR

Length . . . : 100

Text . . . . :

Value

Offset \*..+..1...+...2...+...3...+...4...+...5

0 'CP@D00001YYNNNNNN

50 ' ' '

The process object name is positioned in bytes 2 thru 9. LANSA allows recursive calling of processes therefore a two digit suffix (01 thru 09) is added to this process name when it is created.

In this case the object name is "P@D00001" and the process's OS/400 object name is:

#### P@D0000101

If the application demanded that this particular process be called recursively, then subsequent calls would be to programs:

## P@D0000102 -> P@D0000109

**Note:** This name only applies to this process in this partition on this machine. It will vary between CPUs and partitions - such as when it is exported/imported between partitions, LANSA systems, or sites.

## **Function OS/400 Object Names**

The object names of LANSA functions are much easier to determine than process object names.

LANSA functions created with the FUNCTION OPTIONS(\*DIRECT) option have the following naming convention:

@FFFFFF

where:

@ is the prefix FFFFFFF is the function name

That is, the object name is the function name with an "@" symbol as its prefix. For example, if a function EMPLIST is defined and created with FUNCTION OPTIONS(\*DIRECT) specified, its OS/400 object will be @EMPLIST.

## **4.3 Calling Processes**

A LANSA process is called with five parameters similar to this:

#### In RPG/400

```
CALL 'PRONAME'
PARM DC@IDS
PARM DC@EDS
PARM PRNMIN
PARM PR@IDS
PARM DUMMY1
```

## **In Control Language**

```
CALL PGM(PRONAME) +
PARM(&DC@1DS &DC@EDS &PRNMIN &PR@IDS @DUMMY1)
```

where:

PRONAME Is the process object name

DC@IDS System information A(1024)

DC@EDS Extra System information A(2500)

PRNMIN Process name information A(14)

PR@IDS Process Information A(5000)

DUMMY1 Dummy parameter A(1)

For information about these parameters, refer to 4.5 Access to LANSA Data Structures.

# **4.4 Calling Functions**

A LANSA function has three standard parameters similar to this:

#### In RPG/400

CALL 'FUNNAME'
PARM DC@IDS
PARM DC@EDS
PARM PR@IDS

## **In Control Language**

CALL PGM(FUNNAME) + PARM(&DC@IDS &DC@EDS &PR@IDS)

where:

FUNNAME Is the function object name

DC@IDS System information A(1024)

DC@EDS Extra System information A(2500)

PR@IDS Process Information A(5000)

For information about these parameters, refer to 4.5 Access to LANSA Data Structures.

#### 4.5 Access to LANSA Data Structures

When LANSA invokes a process or a function, it passes some data structures that contain information that is vital for the function to execute as expected. Information such as this:

- Partition information (command key descriptions, data and module library names, SAA/CUA information etc)
- Multilingual information
- Environment information (OS/400 Version/Release)
- LANSA system information

In order for a user defined 3GL program to successfully invoke a LANSA process or function directly this information needs to be available. Refer to 4.5.1 Setting up the Required System Information.

**Note:** Some sample program code is shipped with the LANSA software and this can be used to set up this required system information. For the location of this sample code refer to 4.6.3 Data Structures and Working Lists.

### 4.5.1 Setting up the Required System Information

Shipped with the LANSA product are two source members, UD@CALL1 and UD@CALL2. These are available in file DC@F28 in the LANSA Data Library. The objects are not shipped.

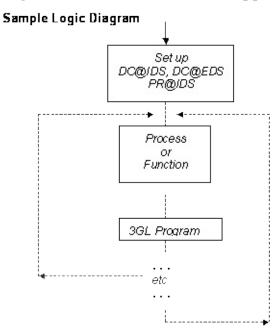
The shipped programs can be used to set up the DC@IDS and DC@EDS information. However, the user defined 3GL program that calls the LANSA process or function will have to supply the PR@IDS - Process Information. (A sample set up for PR@IDS is supplied at the end of this chapter.)

To automatically load the DC@IDS and DC@EDS system information in the user defined 3GL program that will invoke the LANSA process or function, include a call to programs UD@CALL1 and UD@CALL2 respectively .

# Tip

The user defined 3GL program should only load the required information for this type of processing once within a job. Once the system information is loaded it can be passed between all programs within the job.

The following diagram shows a possible logic flow of an application that uses user defined 3GL programs to execute a LANSA application.



# 4.5.2 Source Members Shipped with LANSA

## Data Structures (DC@IDS, DC@EDS & PR@IDS)

- DC@IDS & DC@EDS System Information
   The /COPY source member DC@ISPEC and DC@ESPEC are supplied in file DC@F28 in the LANSA Data Library for RPG/400, and DC@DSPEC for RPGIV.
- PR@IDS Process Information
   The /COPY source members PR@ISPEC and PR@ESPEC are supplied in file DC@F28 in the LANSA Data Library for RPG/400 and PR@DSPEC for RPGIV.

#### **Programs**

- Program UD@CALL1 The source member for UD@CALL1 is supplied in file DC@F28 in the LANSA Data Library.
- Program UD@CALL2
   The source member for UD@CALL2 is supplied in file DC@F28 in the LANSA Data Library.

**Note:** Only the program source of UD@CALL1 and UD@CALL2 is supplied. The program objects will need to be created before they can be executed.

# **4.6 Calling Functions and Passing Data Structures, Working and Exchange Lists**

Other parameters are included when using advanced programming techniques that include passing data structures and/or working lists.

- 4.6.1 Data Structures
- 4.6.2 Working Lists
- 4.6.3 Data Structures and Working Lists
- 4.6.4 Exchange Lists

## **4.6.1 Data Structures**

For example, a function that receives a data structure would be called with a parameter list similar to this:

Data structure received:	PSLMST	
Data structure elements:	EMPNO	A(5)
	SURNAME	A(20)
	GIVENAME	A(20)
	BIRTHDT	A(6)
	SALARY	S(9,2)

#### **In RPG/400**

CALL 'FUNNAME'

PARM DC@IDS
PARM DC@EDS
PARM PR@IDS
PARM PSLMST

# **In Control Language**

CALL PGM(FUNNAME) +
PARM(&DC@IDS &DC@EDS &PR@IDS &PSLMST)

#### where:

FUNNAME	Is the function object name	
DC@IDS	System information A(1024)	
DC@EDS	Extra System information A(2500)	
PR@IDS	Process Information A(5000)	
PSLMST	Passed data structure (Note 1) A(Note 2)	

**Note 1:** The 'Passed data structure' parameter is repeated for each of the Passed data structures.

**Note 2:** The size of the 'Passed data structure' parameter in this scenario could be 60 or 64 bytes. That is, 60 bytes if the PSLMST data structure is a file defined externally to LANSA (OTHER) or 64 bytes if it is a LANSA defined file. The extra 4 bytes in the LANSA file is the @@UPID field used by all files created using LANSA. Each of the fields in the data structure (file) contribute to the aggregate length of the 'Passed data structure'.

**Note 3:** If this is a "wrapper" RDML function that is itself calling another function using PASS\_DS, then it is necessary to refer to all of the fields in the data structure, otherwise their values will not be passed on. A way of doing this would be via the GROUP\_BY RDML command.

# 4.6.2 Working Lists

A function that receives a working list would be called with a parameter list similar to this:

Working list received:	PSLSKL	
Working list elements:	SKILCODE	A(4)
	SKILDESC	A(20)
	COMMENT	A(40)

## **In RPG/400**

CALL	FUNNAME
PARM	DC@IDS
PARM	DC@EDS
PARM	PR@IDS
PARM	PSLSKL
PARM	PSLNUM
<b>PARM</b>	PSLPOS

# **In Control Language**

CALL PGM(FUNNAME) +
PARM(&DC@IDS &DC@EDS &PR@IDS &PSLSKL +
&PSLNUM &PSLPOS)

#### where:

FUNNAME	Is the function object name
DC@IDS	System information A(1024)
DC@EDS	Extra System information A(2500)
PR@IDS	Process Information A(5000)
PSLSKL	Passed working list ( <b>Note 1</b> ) n * A(64) <b>Note 2</b>
PSLNUM	Number of entries in list ( <b>Note 1</b> )

	P(7,0)
PSLPOS	Position in list ( <b>Note 1</b> ) P(7,0)

**Note 1:** The 'Passed working list' parameters are repeated for each passed working list.

**Note 2:** The PSLSKL parameter is defined as the aggregate length of all fields in the working list. The size of the PSLSKL parameter of this field will be 64 bytes. In RPG terms this field is a multiple occurrence data structure where each of the fields are sub-fields of PSLSKL.

The 'n' occurrences is the number of entries in the working list that have been passed.

# 4.6.3 Data Structures and Working Lists

A function that receives both working lists and data structures would be called with a parameter list similar to this:

Working list received:	PSLSKL	
Working list elements:	SKILCODE	A(4)
	SKILDESC	A(20)
	COMMENT	A(40)
Data structure received:	PSLMST	
Data structure elements:	EMPNO	A(5)
	SURNAME	A(20)
	GIVENAME	A(20)
	BIRTHDT	A(6)
	SALARY	S(9,2)

### **In RPG/400**

CALL	FUNNAME
PARM	DC@IDS
PARM	DC@EDS
PARM	PR@IDS
PARM	PSLSKL
PARM	PSLNUM
PARM	PSLPOS
PARM	PSLMST

## **In Control Language**

CALL PGM(PRONAME) +
PARM(&DC@IDS &DC@EDS &PR@IDS &PSLSKL +
&PSLNUM &PSLPOS &PSLMST)

where:	
--------	--

FUNNAME	Is the function object name	
DC@IDS	System information A(1024)	
DC@EDS	Extra System information A(2500)	
PR@IDS	Process Information A(5000)	
PSLSKL	Passed working list ( <b>Note 1</b> ) n * A(64)	
PSLNUM	Number of entries in list ( <b>Note 1</b> ) P(7,0)	
PSLPOS	Position in list ( <b>Note 1</b> ) P(7,0)	
PSLMST	Passed data structure ( <b>Note 2</b> ) A( <b>Note 3</b> )	

**Note 1:** The 'Passed working list' parameters are repeated for each passed working list.

The 'n' occurrences is the number of entries in the working list that have been passed.

**Note 2:** The 'Passed data structure' parameter is repeated for each of the Passed data structures.

**Note 3:** The size of the 'Passed data structure' parameter in this scenario could be 60 or 64 bytes. That is, 60 bytes if the PSLMST data structure is a file defined externally to LANSA (OTHER) or 64 bytes if it is a LANSA defined file. The extra 4 bytes in the LANSA file is the @@UPID field used by all files created using LANSA. Each of the fields in the data structure (file) contribute to the aggregate length of the 'Passed data structure'.

Refer to 4.5.2 Source Members Shipped with LANSA for further information about the standard parameters.

### 4.6.4 Exchange Lists

You will find a thorough explanation of exchange lists with the EXCHANGE in the *LANSA Technical Reference Guide*.

When using exchange lists it is important to remember that extra calls to M@EXCHL are needed when directly calling a function.

The difference between using LANSA to RUN a function and directly calling the function, is that LANSA controls the transfer of the external (3GL) exchange list to the LANSA exchange list and vice versa. When a function is called directly these transfers must be done through additional calls to the M@EXCHL program.

One call to M@EXCHL is done just before the call to the 3GL program to map the external exchange list into the LANSA exchange list. Another is done right after the call to the 3GL program to map the LANSA exchange list back into the external exchange list.

The parameters required for these M@EXCHL calls are:

#### Parm No Type Min Len Max Len Comments

1	Alpha 3	3	'\$AC' : Transfer onto exchange list '\$RC' : Receive from exchange list
2	Alpha 1	1024	DC@IDS - System Information
3	Alpha 1	2500	DC@EDS - Extra System Information
4	Alpha 1	5000	PR@IDS - Process Information

The correct sequence of calls when using exchange lists within a 3GL program to pass field values to a LANSA function is:

- 1. CALL M@EXCHL 'CLR'
- 2. CALL M@EXCHL 'PUT'
- 3. CALL M@EXCHL '\$AC'
- 4. CALL Function (or Process)
- 5. CALL M@EXCHL '\$RC'

6. CALL M@EXCHL 'GET' ...

### 4.7 Examples of Using Direct Calling

The functions and processes used by these examples are a part of the shipped Demonstration Personnel System. All examples execute on an IBM i with a version of OS/400 at V1R3M0 or higher.

The examples are provided with the LANSA Product. The source is stored in file DC@F28 in the LANSA Data Library with the following member names:

UD@FUNC1 Call a LANSA Function using RPG/400

UD@FUNC2 Call a LANSA Function using Control Language (CLP)

UD@FUNC3 Call a LANSA Function using ILE RPG

UD@FUNC4 Call a LANSA Function using Control Language (CLLE)

UD@PROC1 Call a LANSA Process using RPG/400

UD@PROC2 Call a LANSA Process using Control Language (CLP)

UD@PROC3 Call a LANSA Process using ILE RPG

UD@PROC4 Call a LANSA Process using Control Language (CLLE)

The source members for these programs for RPG/400(UD@FUNC1) & ILE RPG (UD@FUNC3) can be found in file DC@F28 in the LANSA Data Library.

### 4.7.1 Executing the LANSA INQUIRE Function with RPG

The sample RPG members are an examples of directly calling a LANSA function. They show the basics necessary to set up all relevant information for a function. The function is executed in English.

The programs UD@FUNC1 & UD@FUNC3 call the LANSA function INQUIRE - 'Browse/Maintain Employee and Skill Files' and pass to the function, via the exchange list, a value for the field EMPNO - 'Employee number'.

**Note:** The employee number will only be exchanged if the INQUIRE function has been created with access to 3GL exchange list flag set on.

### 4.7.2 Executing the LANSA INQUIRE Function with CL

The source members for these programs for CLP (UD@FUNC2) & CLLE (UD@FUNC4) are supplied in file DC@F28 in the LANSA Data Library. These examples perform exactly the same processing as the Executing the LANSA INQUIRE Function with RPG sample code, except these programs have been written using Control Language (CLP & CLLE). The function is executed in English.

### 4.7.3 Executing the LANSA PSLSYS Process with RPG

The source members for these programs for RPG/400 (UD@PROC1) & ILE RPG (UD@PROC3) are supplied in file DC@F28 in the LANSA Data Library.

The sample RPG members are an examples of directly calling a LANSA Process. They show the basics necessary to set up all relevant information to run a LANSA Process.

The programs UD@PROC1 & UD@PROC3 call the LANSA Process PSLSYS - 'Personnel System'. The process is executed in English.

### 4.7.4 Executing the LANSA PSLSYS Process with CL

The source members for these programs for CLP (UD@PROC2) & CLLE (UD@PROC4) is supplied in file DC@F28 in the LANSA Data Library.

These examples perform exactly the same processing as the Executing the LANSA PSLSYS Process with RPG sample code, except these programs have been written using Control Language (CLP & CLLE). The process is executed in French.

### 4.7.5 Sample Set up of Process Information - PR@IDS

The following RPG/400 code shows the necessary field settings for the Process Information data structure - PR@IDS.

```
I* Program Information Data Structure
I*
I$PID
       SDS
                        200
I
                 *PROGRAM $PGMNM
I*
C* Clear the exchange list
C*
C
           Z-ADD0
                     PR@NXL
C
                     PR@NXP
           Z-ADD1
C
          MOVEL<function>PR@FUN
                                      Function name
C
           MOVEL<fdesc> PR@FUD
                                     Function
C*
                          description
C
          MOVEL*BLANKS PR@NXT
C
          MOVEL*BLANKS PR@NXD
C
          MOVEL<ptype> PR@TYP
                                     Process type
C*
                          MNU, FUN, CMD,
                          HLP, or EXT
C
          MOVELcess> PR@PRO
                                      Process name
C
          MOVEL<pdesc> PR@PRD
                                      Process
C*
                          description
C
                      PR@RCL
                                   Heavy = N,
          MOVEL'L'
C*
                          LIGHT = L
C
           MOVEL'DIS'
                       PR@MDE
C
           MOVEL*BLANKS PR@VER
C
           MOVELSA@KEH
                          PR@CEX
C
          MOVELSA@KCN
                           PR@CMN
C
                           PR@CDM
          MOVELSA@KMS
C
          MOVELSA@KAD
                           PR@CAD
C
                           PR@CCH
           MOVELSA@KCH
C
                           PR@CDL
           MOVELSA@KDL
                      PR@CU1
C
           MOVEL'00'
C
           MOVEL'00'
                      PR@CU2
                      PR@CU3
           MOVEL'00'
```

```
C
C
          MOVEL'00'
                      PR@CU4
          MOVEL'00'
                      PR@CU5
C
C
C
          MOVE *BLANKS
                         PR@CD1
                         PR@CD2
          MOVE *BLANKS
                         PR@CD3
          MOVE *BLANKS
C
          MOVE *BLANKS
                         PR@CD4
C
          MOVE *BLANKS PR@CD5
C
          MOVEL*BLANKS PR@KEY
C
          MOVEL*BLANKS PR@ELK
C*
C*
C* $PGMNM is the name of this program
C* from the Program Information Data Structure (PIDS)
C^*
C
          MOVEL$PGMNM PR@RMQ
C
          MOVE '01'
                     PR@NIC
C
          MOVEL'Y'
                      PR@RQR
C
          MOVEL'Y'
                      PR@SAA
C
          MOVE 'N'
                     PR@EPR
```