LANSA for i User Guide

- About this guide
- LANSA Overview
- Fields
- Files
- Processes and Functions
- Housekeeping
- Administration
- Documentor
- LANSA PC Development
- Developer Services
- Configuration Items
- Appendix A. Virtual Fields
- Appendix B. System Data Areas
- Appendix C. Technical Notes

Edition Date October 23, 2014 © LANSA

About this guide

This guide describes the facilities available with LANSA and the LANSA Repository, how to access and how to use them. It does not cover the creation of LANSA's RDML code - this is described in detail in the *LANSA Technical Reference*.

The contents have been grouped into four main areas which represent the four control menus in the LANSA system. These are:

- FIELDS
- FILES
- PROCESSES
- HOUSEKEEPING.

Within these major headings you will find a description of each entry on each LANSA menu in the same sequence as the item appears on the menu.

For every component of the LANSA system, you will find:

- What the component does.
- How to access it.
- When to use it.
- Why you would use it.
- The input requirements and options.
- Examples of using it.

1. LANSA Overview

LANSA is a system that can create, review, change or delete any of the following types of "object":

- **Models**Logical models of the information that your organization stores and uses. Typically fields and files are created as physical representations of a logical data model.
- Fields

The smallest physical element that can be defined. Typically a field is a discrete piece of information such as an invoice number, a date, a part number, etc.

• Files

A set of records. Each record is a group of related fields.

• Processes and their associated Functions

Programs used to define the method used to store, update, delete, display, print and manipulate information stored in files.

1.1 How is LANSA used?

LANSA is normally used in the following way:

- Optionally, a logical model of the information that you need to handle is defined and refined, until it represents a fair and accurate representation of your data requirements. This logical model may then be converted into a set of fields and files that represent a physical implementation of it.
- Fields, files and processes are set up (or have been created automatically from a logical model) to store and process the information your company or organization needs.
- These are then used to input, store, maintain and retrieve computerized company or organization records. The level of knowledge required to use LANSA varies according to which task is being performed:
 - To set up a logical data model you will require specialized data modeling skills and an ability to think in abstract and pure terms about the type of information that is being processed.
 - To set up fields, files and processes you will require a sound knowledge of most of the components of LANSA (except for logical data modeling) and some experience in data processing at the programming level.
 - To use LANSA to maintain company records you will need very little knowledge of LANSA. The main skills required relate to more general data processing requirements such as keyboard skills, a general understanding of computer systems and a knowledge of the organization's own business applications. The attainment of these skills is beyond the scope of this guide.

1.2 The Components of LANSA

There are three basic "object" types: fields, files and processes, that you normally manipulate when using LANSA. These three objects are reflected in the major developer menus in the system.



Examples of the screens represented are given in 1.5 The Major LANSA Developer Menus.

There is another way to access the LANSA development environment. It is called "Advanced access" and uses "Work with" style screens. Before using Advanced access please note that:

- You should first be familiar and comfortable with accessing LANSA in the normal manner.
- The normal method of access is used in all LANSA guides and tutorials. When you understand a facility in normal mode, the use of it in advanced mode is easy and implicit.

You access Advanced mode access using the DEVELOPER(*ADVANCED) parameter on the LANSA command.

When this parameter is used the developer menus are structured like this:



Refer to 1.6 The Major LANSA Developer(*Advanced) Screens for examples of the DEVELOPER(*ADVANCED) "Work with" style screens.

1.3 Accessing LANSA

LANSA can be invoked from command entry display by entering the command:

LANSA

If the system responds with a message that says "Command LANSA *LIBL not found." then re-enter the command as:

<pgm lib>/LANSA

where **<pgm lib>** is the name of the LANSA program library. Normally this is DC@PGMLIB but it may be different on your system. Refer to What is a LANSA system? in the *Introduction to LANSA for i* for more details.

Once invoked, the LANSA command will automatically establish the environment it needs to operate and present the MAIN SYSTEM MENU. LANSA can also be invoked by using the CALL command:

CALL LANSA

or

CALL <pgm lib>/LANSA

Either of these methods can be used to invoke LANSA from your own application menus.

In addition, the command LANSA and the program LANSA have parameters that allow you to pass information into the LANSA system. Refer to Using a Process or Function from Your Own Application for more details.

Note: If you wish to access the demonstration or training LANSA partition then you should also include parameter PARTITION(DEM) on the command LANSA or program call to LANSA. Alternatively, you could change the current partition to "DEM" from the Housekeeping Menu after invoking LANSA. For details, refer to Change the System Partition Being Worked With.

1.4 LANSA Command

The LANSA command is used to invoke LANSA, control how the LANSA environment is set up and pass information into the LANSA system.

By specifying certain values with this command, the behavior of LANSA can be tailored to suit your installation practices.

Optional		
LANSA REQUEST MENU>		
RUN		
SUBMIT		
EXPORT		
IMPORT		
REORG		
LICENSE		
IMPORTMSGS		
EXPORTMSGS		
LCOE		
FIELDS		
FILES		
PROCESSES		
PCMAINT		
PCEXPORT		
VERIFYFRT		
STRLISTEN		
CONFIGURE		
ABOUT		
X_RUN		
STRMON		
> PROCESS process name>		
> FUNCTION function name>		
> PARM01 parameter number 1>		
> PARM02 parameter number 2>		
> PARM03 parameter number 3>		

> PARM04 parameter number 4>
> PARM05 parameter number 5>
> PARM06 parameter number 6>
> PARM07 parameter number 7>
> PARM08 parameter number 8>
> PARM09 parameter number 9>
> PARM10 parameter number 10>
> PARTITION partition identifier>
> LANGUAGE language code>
> TASK ID task identifier>
> PC_TYPE N> B W P
> DEVELOPER *NO> *YES *ADVANCED N Y A
> ALLOW_MSGS *NO> *YES N Y
> DATE_SRCE *SYSTEM>

*JOB
S
J
> BDEBUG N>
Y
> BDEBUG_DEV device name>
> BDEBUG-MSG message queue
> X_RUNADPRM additional X_RUN parameters

1.4.1 Parameters

REQUEST

This is the action requested to be performed within LANSA. The action requested can be up to 10 characters in length. The actions that may be requested include:

- **MENU**: Specifies that the Main System Menu (Advanced if DEVELOPER(*ADVANCED) is specified) is to be presented on entry to LANSA. This is the default action, and the one most likely to be used by developers.
- **RUN**: Specifies that a process or a function is to be used. This allows a process or a function to be called without going through the normal LANSA menus.

Refer to Using A Process Or Function Directly From Your Own Application for more details.

- **SUBMIT:** Specifies that a process is to be submitted to batch. This allows a process to be submitted without going through the normal LANSA menus. Refer to Submitting A Process To Batch Directly From Your Own Application for more details.
- **EXPORT:** Specifies that an IBM i export list is to be exported directly. This allows an export list to be exported without going through the normal LANSA menus.

Refer to Direct Exporting of Objects in a List for more details.

It is not recommended that this facility be used interactively (i.e. directly from a workstation) because of the relatively large number of database I/Os performed.

To use this request, the user profile being used must have access to *Work with Lists of Objects to Be Exported* in the Housekeeping Menu. Refer to User Access to the LANSA System for information.

• **IMPORT:** Specifies that the import facility is to be used directly. This allows an import to be done without going through the normal LANSA menus.

Refer to Direct Importing of Objects for more details.

It is not recommended that this facility be used interactively (i.e. directly from a workstation) because of the relatively large number of database I/Os performed.

To use this request, the user profile being used must have access to *Import*

objects into this Partition in the Housekeeping Menu. Refer to User Access to the LANSA System for information.

• **REORG:** Specifies that the re-organization facility be executed directly. This allows a LANSA re-organization to be run without going through the normal LANSA menus.

Refer to Directly Invoking The Re-organization Facilityfor more details. It is not recommended that this facility be used interactively (i.e. directly from a workstation) because of the relatively large number of database I/Os performed.

To use this request, the user profile being used must have access to *Submit job to reorg LANSA internal database* in the Housekeeping Menu. Refer to User Access to the LANSA System for information.

- **LICENSE or LICENCE:** Specifies that the LANSA License Facility be invoked. This cannot be accessed via normal LANSA menus. Refer to *The LANSA License Facility* in the *Installing LANSA on IBM i Guide* for more details.
- EXPORTMSGS: Specifies that developer messages are to be exported. This cannot be done via normal LANSA menus. Refer to Export Developer Message for more details. It is not recommended that this facility be used interactively (i.e. directly from a workstation) because of the relatively large number of database I/Os performed.
 IMPORTMSGS: Specifies that developer messages are to be imported.
- IMPORTMSGS: Specifies that developer messages are to be imported. This cannot be done via normal LANSA menus. Refer to Import Developer Messages for more details. It is not recommended that this facility be used interactively (i.e. directly from a workstation) because of the relatively large number of database I/Os performed.
- **LCOE:** Specifies that LANSA Open is to be used. This is used internally by LANSA Open and is not usable in a normal LANSA invocation.
- **FIELDS:** Specifies that the Field Control Menu (Work With Fields screen if DEVELOPER(*ADVANCED)) is to be presented on entry to LANSA. To use this request, the user profile being used must have access to *Work with fields, Advanced menu*. Refer to User Access to the LANSA System for information.
- **FILES:** Specifies that the File Control Menu (Work With Files screen if DEVELOPER(*ADVANCED)) is to be presented on entry to LANSA.

To use this request, the user profile being used must have access to the *Work with files, Advanced menu*. Refer to User Access to the LANSA System for information.

• **PROCESSES:** Specifies that the Process Control Menu (Work With Processes screen if DEVELOPER(*ADVANCED)) is to be presented on entry to LANSA.

To use this request, the user profile being used must have access to the *Work with processes*, *Advanced menu*. Refer to User Access to the LANSA System for information.

- **STRMONITOR:** This is no longer available.
- **STRMON:** Specifies that the host monitor for Visual LANSA is to be started. This is used internally by Visual LANSA and is not usable in a normal LANSA invocation.
- **PCMAINT:** Specifies that the PC Maintenance for Visual LANSA is to be started. This allows definition of information for Visual LANSA. Refer to *Maintaining the PC Definition File* for more details.
- **PCEXPORT:** Specifies that a PC export list is to be exported directly. This allows a PC export list to be exported directly without going through the normal LANSA menus.

Refer to *Direct Exporting of Objects in a List* for more details. It is not recommended that this facility be used interactively (that is, directly from a workstation) because of the relatively large number of I/Os performed. To use this request, the user profile being used must have access to the *Work with list of objects to be exported* in the Housekeeping Menu. Refer to User Access to the LANSA System for information.

- **VERIFYFRT:** Causes a Function Routing Table (FRT) to be verified. Refer to What is Function Routing? for more detailed information about this facility.
- **STRLISTEN:** Specifies that the LANSA Communications Extensions Listener Job is to be started. Refer to the *LANSA Communications Setup Guide* for details of how and when to use this option.
- **CONFIGURE:** Specifies that the LANSA Work with Configuration Items Facility be invoked. This cannot be accessed via normal LANSA menus. Refer to the *Working with Configuration Items* for details of how to use this facility.
- **ABOUT:** Causes the details of this LANSA system, such as LANSA version number, OS version number, libraries and IFS to be displayed on the *LANSA*

Configuration Details screen, frequently called *LANSA About*. You can capture this data on a Save file, to send to LANSA Support in the event of a query. The save file is DCXSUPDTA and it will be saved in the LANSA_XXPGMLIB/SUPPORT directory.

This same information can also be captured using a web interface with the URL:

http://123.45.67.89/cgi-bin/lansaweb? procfun+lansaweb+about+dem

In this case the information can be saved to a save file or, if your email is configured, emailed directly to your LANSA support desk.

• **X_RUN:** Specifies that an RDMLX function is to be executed on IBM i.

PROCESS

This parameter is used to specify the name of the process to RUN, SUBMIT or X_RUN. It can be up to 10 characters in length.

FUNCTION

This parameter is used to specify the name of the function to RUN or X_RUN. It can be up to 7 characters in length.

PARM01

This parameter is used to specify parameter number 1. It can be up to 256 characters in length.

If the request is RUN or SUBMIT, this must exactly match the first user parameter defined for the process being run or submitted.

If the request is EXPORT or PCEXPORT, this must be the name of the previously defined list that specifies object(s) to be exported. It can be up to 7 characters in length.

If the request is IMPORT, this must be the device or device file name if importing from magnetic tape or on a disk, otherwise blank. It can be up to 10 characters in length.

If the request is EXPORTMSGS, this must be the name of the file to receive the export developer message header details. It can be up to 10 characters in length.

If the request is IMPORTMSGS, this must be the name of the file to that contains the developer message header details to be imported. It can be up to 10 characters in length.

If the request is STRLISTEN, this can optionally be the special value

*UNLOCK. The special value *UNLOCK tells the Listener Job to leave LANSA resources unlocked. This option reduces the dependency of the Listener Job on the LANSA system from which it is started.

If the request is REORG, this acts as Change Work Spaces parameter and valid values are N, Y and blank. If the value is blank the default value is 'N"

PARM02

This parameter is used to specify parameter number 2. It can be up to 256 characters in length.

If the request is RUN or SUBMIT, this must exactly match the second user parameter defined for the process being run or submitted.

If the request is EXPORT, this must be the device or device file name if exporting on magnetic tape or disk, blank otherwise. It can be up to 10 characters in length.

If the request is PCEXPORT, this must be the CheckOutUsage of R (read only) or M (maintenance).

If the request is IMPORT, this must be save file name if importing from a save file, otherwise blank. Can be a qualified name. It can be up to 21 characters in length.

If the request is EXPORTMSGS, this must be the name of the file to receive the export developer message text details. It can be up to 10 characters in length.

If the request is IMPORTMSGS, this must be the name of the file to that contains the developer message text details to be imported. It can be up to 10 characters in length.

If the request is REORG, this acts as Total FFC parameter and valid values are 1 to 99 and blank. If the value is blank the default value is 99

PARM03

This parameter is used to specify parameter number 3. It can be up to 256 characters in length.

If the request is RUN or SUBMIT, this must exactly match the third user parameter defined for the process being run or submitted.

If the request is EXPORT, this must be the save file name if exporting to a save file, blank otherwise. Can be a qualified name. It can be up to 21 characters in length.

If the request is PCEXPORT, and the CheckOutUsage (PARM02) is M (maintenance), this must be the Task. It can be up to 8 characters in length.

If the request is IMPORT, this must be name of message queue to which messages and inquiries are to be sent. It can be up to 10 characters in length.

If the request is EXPORTMSGS, this must be the name of the library where the exported developer message files will be created. It can be up to 10 characters in length.

If the request is IMPORTMSGS, this must be the name of library that contains the developer message files to be imported. It can be up to 10 characters in length.

If the request is REORG, this acts as Total Compiles parameter and valid values are 1 to 99 and blank. If the value is blank the default value is 20

PARM04

This parameter is used to specify parameter number 4. It can be up to 256 characters in length.

If the request is RUN or SUBMIT, this must exactly match the fourth user parameter defined for the process being run or submitted.

If the request is EXPORT, this must be the file sequence number if exporting to a device (see PARM02). The default is *END if not specified. It can be up to 7 characters in length (numerics).

If the request is PCEXPORT, and the CheckOutUsage (PARM02) is M (maintenance), this must be the PC Name. It can be up to 10 characters in length.

If the request is IMPORT, this must be the source system type (IBM i). It can be up to 6 characters in length.

If the request is EXPORTMSGS, this must be the export start date. This is in CCYYMMDD format. All developer messages created or changed on or after the export start date will be exported. This must be 8 characters in length.

If the request is REORG, this acts as Remove Multilingual support parameter and valid values are N, Y and blank. If the value is blank the default value is N

PARM05

This parameter is used to specify parameter number 5. It can be up to 256 characters in length.

If the request is RUN or SUBMIT, this must exactly match the fifth user parameter defined for the process being run or submitted.

If the request is EXPORT, this is whether or not to omit RDML source code from exported data. (YES or NO). The default is NO if not specified. It can be

up to 3 characters in length.

If the request is IMPORT, this is the file sequence number if importing from a device (see PARM01). The file sequence number can be up to 7 numbers long. The default is *SEARCH. If the import is from an optical device (see PARM01), then this is the optical file name. The optical file name can be up to 80 characters long.

If the request is REORG, this acts as Purge Developer Message starting from date parameter and valid value is a date format in CCYYMMDD and blank. If the value is blank the default value is 00000000. The date must be a valid date, as it is not validated.

PARM06

This parameter is used to specify parameter number 6. It can be up to 256 characters in length.

If the request is RUN or SUBMIT, this must exactly match the sixth user parameter defined for the process being run or submitted.

If the request is EXPORT, this is the languages to be exported. Up to 64 language codes can be specified or *ALL to export all languages. Only applicable to multilingual partitions. It can be up to 256 characters in length.

If the request is IMPORT, this is whether or not to import all languages in the exported data. (Y or N). Only applicable to multilingual partitions. This is 1 character in length.

If the request is REORG, this acts as Delete Deleted Partition Language Extensions and valid values N, Y and blank. If the value is blank the default value is N

PARM07

This parameter is used to specify parameter number 7. It can be up to 256 characters in length.

If the request is RUN or SUBMIT, this must exactly match the seventh user parameter defined for the process being run or submitted.

If the request is EXPORT, this is do not save to media - leave in QTEMP. (YES or NO) Default is NO if not specified. It can be up to 3 characters in length.

If the request is IMPORT, this is the languages to be imported. Up to 64 language codes or *ALL to import all languages in exported data. Only applicable to multilingual partitions. It can be up to 256 characters in length. If the request is REORG, this acts as Reorganize Shared LANSA system files

and valid values N, Y and blank. If the value is blank the default value is N

PARM08

This parameter is used to specify parameter number 8. It can be up to 256 characters in length.

If the request is RUN or SUBMIT, this must exactly match the eighth user parameter defined for the process being run or submitted.

If the request is EXPORT, this is whether or not to include LANSA/Document details in the export. (YES or NO). Default is NO if not specified. It can be up to 3 characters in length.

If the request is IMPORT, this is the default language. Defaults to current language if not specified. Only applicable to multilingual partitions. It can be up to 4 characters in length.

If the request is REORG, this acts as Purge Task History starting from date parameter and valid value is a date format in CCYYMMDD and blank. If the value is blank the default value is 00000000. The date must be a valid date, as it is not validated.

PARM09

This parameter is used to specify parameter number 9. It can be up to 256 characters in length.

If the request is RUN or SUBMIT, this must exactly match the ninth user parameter defined for the process being run or submitted.

If the request is EXPORT, specify in characters 1 - 3 whether or not to omit frameworks and groups. (YES or NO). Defaults to NO if not specified.

If the request is IMPORT, specify, in characters 1-3, whether or not to assign new internal names. (YES or NO). Defaults to YES if not specified.

Use characters 4-6 to delete the \$\$ files. Pass as YES or NO. Defaults to NO if not specified.

If YES, any \$\$ version of the files being imported will be automatically deleted. The file name to be deleted is composed of \$\$ plus the first 8 characters of the file name. Where 9 and 10 character file names are not unique in the first 8 characters, this may result in additional \$\$ files being deleted. (For the processing to complete, these files must be deleted.)

PARM10

This parameter is used to specify parameter number 10. It can be up to 256

characters in length.

If the request is RUN or SUBMIT, this must exactly match the tenth user parameter defined for the process being run or submitted.

If the request is IMPORT, specify in characters 1-10 the name of the library containing export objects. Only applicable for IBM i to S/38 transfer. It can be up to 10 characters in length.

Use characters 11-13 to specify *Allow Name Changes*. Pass as YES or NO. Defaults to NO if not specified.

Use characters 14-16 to specify *Allow Type Changes*. Pass as YES or NO. Defaults to NO if not specified.

Use characters 17 –19 to specify whether or not to omit frameworks and groups. (YES or NO). Defaults to NO if not specified.

PARTITION

This parameter is used to specify the name of the partition to be used. It can be up to 3 characters in length.

If not specified, the partition defaults to the SYS partition.

LANGUAGE

This parameter is used to specify the language to be used. It can be up to 4 characters in length.

Only applicable to multilingual partitions.

If not specified, the language defaults to the partition default language.

TASK_ID

This parameter is used to specify the task identifier to be used for development work. It can be up to 8 characters in length.

PC_TYPE

This parameter is used to specify the type of PC used with the GUI. This is 1 character in length. This is used internally by LANSA and is not usable in a normal LANSA invocation.

DEVELOPER

This parameter is used to specify whether LANSA presents a command line and work with interface. It is up to 9 characters in length.

• ***NO, N:** Specifies that the LANSA menus are presented in the traditional style without a command line.

• ***YES**, **Y**: Specifies that the LANSA menus are presented in the traditional style with a command line.

If access is given to developer services (see ALLOW_MSGS), then the options to execute an IBM i command and execute user defined developer services exit programs will be available.

*ADVANCED, A: Specifies that the LANSA menus are presented in the work with style with a command line.
 If access is given to developer services (refer to ALLOW_MSGS), then the options to execute an IBM i command and execute user defined developer services exit programs will be available.

ALLOW_MSGS

This parameter is used to specify whether or not access is given to developer services. It is up to 4 characters in length.

- ***NO**, **N**: Specifies that no access is given to developer services. This includes the notification of developer messages.
- ***YES**, **Y**: Specifies that access is given to developer services.

DATE_SRCE

This parameter is used to specify where to retrieve the date and date format from. It is up to 7 characters in length.

- ***SYSTEM**, **S**: Specifies that the date and date format are to be retrieved from the system values. This is the default.
- ***JOB**, **J**: Specifies that the date and date format are to be retrieved from the job attributes.

BDEBUG

This parameter is used to specify that batch debug of LANSA RDML functions is required. This is used internally by LANSA and is not useable in a normal LANSA invocation.

BDEBUG_DEV

This parameter is used to specify the name of the display device to be used for debugging batch RDML functions. This is used internally by LANSA and is not useable in a normal LANSA invocation.

BDEBUG_MSG

This parameter is used to specify the name of the message queue to be used to issue messages when using batch debugging. This is used internally by LANSA

and is not useable in a normal LANSA invocation.

X_RUNADPRM

This parameter is used to specify additional parameters for an X_RUN request. These additional parameters must be specified in the X_RUN format, e.g. PRTR=*PATH. Refer to Standard X_RUN Parameters in the *Technical Reference Guide* for details of X_RUN parameters.

For examples, refer to 1.4.2 Execute RDMLX function.

1.4.2 Execute RDMLX function

An RDMLX function may be executed on an IBM i by using either the LANSA command or by calling X_RUN.

LANSA command parameter	X_RUN parameter
PROCESS	PROC=
FUNCTION	FUNC=
PARTITION	PART=
LANGUAGE	LANG=
TASK_ID	TASK=
DATE_SRC	DATS=
BDEBUG	DBUG=
X_RUNADPRM	All other X_RUN parameters may be specified in the LANSA command parameter X_RUNADPRM. They must

be specified in the X RUN format (eg PRTR=*PATH)

Refer to Standard X_RUN Parameters in the *Technical Reference Guide* for details of X_RUN parameters.

Examples

Command to execute RDMLX function CALC in process ORDERX in partition DEX:

```
LANSA X_RUN PROCESS(ORDERX) FUNCTION(CALC) PARTITION(DEX)
```

This is equivalent to:

CALL X_RUN PARM('PROC=ORDERX FUNC=CALC PART=DEX')

To specify additional parameters:

LANSA X_RUN PROCESS(ORDERX) FUNCTION(CALC) PARTITION(DEX) X_RUNADPRM(PRTR=*PATH)

This is equivalent to:

LANSA REQUEST(X_RUN) PARTITION('DEX') X_RUNADPRM('PROC=ORDERX FUNC=CALC PRTR=*PATH')

1.4.3 Error Messages for LANSA

Escape messages

DCM9997 LANSA processing ended in error - see previous messages for cause

Previous messages will be available in the job log for the job.

1.5 The Major LANSA Developer Menus

The first menu that is presented when LANSA is started is the MAIN SYSTEM MENU. This menu controls access to all other components of the system:

Fnn=HELP Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Chg Tsk ID

From this screen it is possible (if you are authorized) to:

- Execute a process interactively (i.e. at your workstation)
- Submit a process for execution in a batch environment
- Display the Field Control Menu
- Display the File Control Menu
- Display the Process Control Menu
- Display the Housekeeping Menu
- Display the Administration Tasks Menu
- Change Current Task ID

• Exit from the system.

Current Task ID

If you have Task Tracking active and you sign on with a Task ID, you will find that the Current Task ID is displayed on every screen. You can change the current Task ID if you are authorized to do so.

The menu that controls the definition and amendment of fields is the Field Control Menu:

pppDICMENU	Field Control Menu	Current Task ID XXXXXXXX			
Enter number of function or place cursor on same line and press Enter					
 1. Create new field definitions 2. Review or change field definitions 3. Review or change field multilingual attributes 4. Review, change or create field validation checks 5. Review, change or create field HELP text 					
6. Print field def	initions stem variable definition				
8. Review or change system variable definitions					
9. Create new multilingual variables 10. Review or change multilingual variables					
11. Create or re-c	reate a field reference fi	le			
Fnn=HELP Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Chg Tsk Id					

Note that only entries which you are authorized to use will appear on your menu. Multilingual entries will only appear in a multilingual partition. The Current Task ID is only displayed if Task Tracking is active. For more information about Task Tracking, refer to 1.11 Task Tracking.

The menu that controls the definition and amendment of files is the File Control Menu:

pppFILMENU File Control Menu Current Task ID XXXXXXXX

Enter number of function or place cursor on same line and press Enter

_ 1. Create a new file definition

2. Review or change a file definition

3. Delete a file definition

- 4. Make new or amended file definition operational
- 5. Print file definitions
- 6. Bulk Load of OTHER Files
- 7. Exit from LANSA

```
Fnn=HELP Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Chg Tsk Id
```

Note that only entries which you are authorized to use will appear on your menu. The Current Task ID is only displayed if Task Tracking is active.

The Bulk Load of OTHER files is optionally available in each partition. To enable this option for a partition, you must use the LANSA Import facility and import from save file LS@BULKLOD.

The menu that controls the definition and amendment of processes is the Process Control Menu:

pppPROMENUProcess Control MenuCurrent Task ID XXXXXXEnter number of function or place cursor on same line and press Enter_____1. Create a new process definition2. Review or change a process definition3. Delete a process definition4. Compile a process from new or amended definition

5. Review, change or create process HELP text
 6. Print process definitions
 7. Use a process
 8. Submit a process to batch
 9. Turn LANSA debugging mode on
 10. Turn LANSA debugging mode off
 11. Exit from LANSA

Fnn=HELP Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Chg Tsk Id

Note that only entries which you are authorized to use will appear on your menu. The Current Task ID is only displayed if Task Tracking is active. The menu that controls access to the housekeeping facilities within LANSA is the Housekeeping Menu:

Current Task ID XXXXXX pppHKPMENU Housekeeping Menu Enter number of function or place cursor on same line and press Enter 1. Review access to objects defined within LANSA 2. Review user access to LANSA system 3. Submit job to reorg LANSA internal data base 4. Change partition being worked with 5. Create or change system partition definitions 6. Work with partition multilingual attributes 7. Use the file maintenance utility 8. Work with list of objects to be exported 9. Import objects into this partition 10. Work with Application Template definitions 11. Use the dictionary conversion utility (if installed) 12. Create, review or run an online presentation 13. Work with multilingual development textual data 14. Work with tasks

Fnn=HELP Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Chg Tsk Id

Note that only entries which you are authorized to use will appear on your menu. The Current Task ID is only displayed if Task Tracking is active.

1.5.1 Select Entries from the Menus

Entries from LANSA menus are selected by either entering the number associated with the function you wish to use, or by placing the screen cursor on the same line and pressing enter.

For an example, to choose the "Display Field Control Menu" option from the example menu below:



1.6 The Major LANSA Developer(*Advanced) Screens

The first menu that is presented when LANSA is started using parameter DEVELOPER(*ADVANCED) is the MAIN SYSTEM MENU (ADVANCED). This menu controls access to all other components of the system.

Main System Menu (Advanced) Current Task ID XXXXXXX				
Enter number of function or place cursor on same line and press Enter.				
 1. Work with Fields 2. Work with System Variables 3. Work with Multilingual Variables 4. Work with Files 5. Work with Visual LANSA objects 6. Work with Processes 7. Work with Housekeeping Tasks 8. Work with Administration Tasks 9. Work with System Documentation 10. Work with submitted jobs - Workstation 11. Work with submitted jobs - User 12. Display workstation message queue 13. Exit from LANSA 				
Command :				
Fnn=HELP Fnn=Exit Fnn=Prompt Fnn=Retrieve Fnn=Msgs				

Fnn=Chg Tsk Id Fnn=Cancel

From this screen it is possible (if you are authorized) to:

- Select "Work with Fields"
- Select "Work with System Variables"
- Select "Work with Multilingual Variables"

- Select "Work with Files"
- Select "Work with Processes"
- Display the Housekeeping Menu
- Display the Administration menu
- Change Current Task ID
- Exit from the system

Task Tracking

If you have Task Tracking active and you sign on with a Task ID, you will find that this Current Task ID is displayed on every screen. You can change the current Task ID if you are authorized to do so.

The Task ID locked to an object may be displayed or hidden by toggling a function key.

The screen which controls the definition and amendment of fields using DEVELOPER(*ADVANCED) is the "Work with Fields" screen.

Work with Fields			
Position to Current Task ID XXXXXXX Impact List			
Type option(s) and press Enter. 2=Review/Change definition 4=Delete 5=HELP Text 6=Print 12=Validation 16=Multilingual attributes 17=Last Action Details			
Validation			
Opt Field Description Type Len Dec Dic File			
AB\$OPT Action bar option A 3			
$_$ @@UPID Field update / access identifier P 7 0			
ADDRESS3 State and Country A 25 1			
ADDRESS1 Street No and Name A 25 1			
ADDRESS2 Suburb or town A 25 1			
Command :			
Fnn=HELP Fnn=Exit Fnn=Prompt Fnn=Create Fnn=Retr Fnn=Other Fnn=Dsp Tsk Id Fnn=Cancel Fnn=Msgs Fnn=SortByField			

Note that only options which you are authorized to use will appear. Multilingual options will only appear in a multilingual partition. Impact List is described in **1.6.2 Using an Impact List**. The Current Task ID is only displayed if Task Tracking is active.

The Task ID locked to an object may be displayed or hidden by toggling the Disp/Hid Task ID function key. When displayed, the Task ID replaces the current data to the right of each object's line.

г

You can view the fields by Field name or Description using the SortByDesc/Field function key. The Description used for this sort is the one you enter via the Create or Maintain functions and not the description maintained by the multilingual attribute option.

Work with Fields			
Position to Current Task ID XXXXXXX Impact List			
Type option(s) and press Enter. 2=Review/Change definition 4=Delete 5=HELP Text 6=Print 12=Validation 16=Multilingual attributes 17=Last Action Details			
Validation			
Opt FieldDescriptionType Len Dec Task ID@@UPIDField update / access identifierP70XXXXXXXAB\$OPTAction bar optionA3XXXXXXXADDRESS1Street No and NameA25XXXXXXXADDRESS2Suburb or townA25XXXXXXXADDRESS3State and CountryA25XXXXXXX			
Command :			
Fnn=HELP Fnn=Exit Fnn=Prompt Fnn=Create Fnn=Retr Fnn=Other Fnn=Hid Tsk Id Fnn=Cancel Fnn=Msgs Fnn=SortByDesc			

The screen which controls the definition and amendment of system variables using DEVELOPER(*ADVANCED) is the "Work with System Variables" screen.

Work with System Variables				
Position to Current Task ID XXXXXXX Impact List				
Type options, press Enter				
2=Review/Change 4=Delete 6=Print 17=Last Action Details				
Opt System Variable Description				
*AUTOALP06FCTNO LANSA Function Name				
*AUTOALP09PRONO LANSA Process Name				
*AUTOALP10PRJNO Project Identifier (LANSA Project)				
*AUTOALP10TKNNO Task notes sequence (LANSA Project)				
*AUTOALP10TSKNO Task IDentifier (LANSA Project)				
*BLANK Blank / blanks variable				
*BLANKS Blank / blanks variable				
*CENTURY GREATER Century when date greater than switch				
*CENTURY LESSEQUAL Century when date less / equal to switch +				
Command :				
Enn=HELP Enn=Exit Enn=Prompt Enn=Create Enn=Retrieve Enn=Cance				
Fnn=Msøs Fnn=Dsn Tsk Id Fnn=SortBvDesc				

Note only options which you are authorized to use will appear. Impact List is described in 1.6.2 Using an Impact List. The Current Task ID is only displayed if Task Tracking is active.

You can see/hide the Task ID locked to a variable by toggling the Disp/Hid Task ID function key. The Task ID is displayed to the right of the screen.

You can view the System Variables by System Variable name or Description using the SortByDesc/Variable function key. The Description used for this sort is the one you enter via the Create or Maintain functions and not the description maintained by the multilingual attribute option.

The screen which controls the definition and amendment of multilingual variables using DEVELOPER(*ADVANCED) is the "Work with Multilingual Variables" screen.

Work with Multilingual Variables				
Position to Current Task ID XXXXXXX Impact List				
Type options, press Enter 2=Review/Change 4=Delete 6=Print 17=Last Action Details				
Opt Variable Name Left Aligned Value in Current Language				
*MTXTDDN Down				
*MTXTDEMABWIN02901 Select Department				
*MTXTDEMABWIN05001 Select Section				
*MTXTDEMABWIN07301 Select Employee				
*MTXTDEMCALEN02801 Calendar				
*MTXTDEMCALEN05801 MONDAY				
*MTXTDEMCALEN06001 TUESDAY				
*MTXTDEMCALEN06201 WEDNESDAY				
*MTXTDEMCALEN06401 THURSDAY				
*MTXTDEMCALEN06601 FRIDAY				
*MTXTDEMCALEN06801 SATURDAY	+			
Command :				
Fnn=HELP Fnn=Exit Fnn=Prompt Fnn=Create Fnn=Retrieve Fnn=Cance Fnn=Msgs Fnn=Dsp Tsk Id Fnn=SortByDesc				

Note that only options which you are authorized to use will appear. Impact List is described in 1.6.2 Using an Impact List. The Current Task ID is only
displayed if Task Tracking is active.

The Task ID locked to a variable may be displayed or hidden by toggling the Disp/Hid Task ID function key. The Task ID is displayed to the right of the screen.

You can view the Multilingual Variables by Variable Name or Description using the SortByDesc/Variable function key. The Description used for this sort is the one you enter via the Create or Maintain functions and not the description maintained by the multilingual attribute option.

The screen which controls the definition and amendment of files using DEVELOPER(*ADVANCED) is the "Work with Files" screen.

Work with Files
Position to Current Task ID XXXXXXXX Impact List
Type options, press Enter 2=Review/Change Fields 4=Delete 6=Print 7=Make operational 12=Validation 16=Multilingual attributes 17=Last Action Dets 19=Logical Views 21=Access Routes 23=Vrt Flds 25=Batch Control 27=DataBase attributes 29=Load Other 31=FMU
No. ofOpt FileLibrary DescriptionLogicals_DEPTABDC@DEMOLIB Department code table0_LCOI01DC@DEMOLIB Testing LCOI file1_LCOI02DC@DEMOLIB Large LCOI test file0_LCOI03DC@DEMOLIB LCOI Department File0_LCOI04DC@DEMOLIB LCOI Section File0_PSLMSTDC@DEMOLIB Personnel2
Command :
Fnn=HELP Fnn=Exit Fnn=Prompt Fnn=Add/Create Fnn=Retrieve

Fnn=Bulk load Fnn=Cancel Fnn=Msgs Fnn=Dsp Tsk Id Fnn=SortByDesc

Note only options which you are authorized to use will appear. Multilingual options will only appear in a multilingual partition. Impact List is described in 1.6.2 Using an Impact List. The Current Task ID is only displayed if Task Tracking is active.

By using the Function Key to display the Task ID locked to a file the screen is rewritten with the No. of Logicals field replaced by the Task ID.

The command key Fnn=Bulk load is optionally available in each partition. To enable this option for a partition you must use the LANSA Import facility and import from save file LS@BULKLOD.

You can view the files by File name or Description using the SortByDesc/File function key. The Description used for this sort is the one you enter via the Create or Maintain functions and not the description maintained by the multilingual attribute option.

The screen which controls the definition and amendment of processes using DEVELOPER(*ADVANCED) is the "Work with Processes" screen.

Work with Processes	
Position to Current Task ID XXX	XXXXX Impact List
Type options, press Enter 2=Review/Change Functions 4=Delete 5=HE 10=Use 16=Multilingual attributes 17=Last A 18=Multil special entries	LP text 6=Print 7=Compile Action Dets
Opt Process Description No _ PSLSYS Personnel System _ HELP\$\$AB HELP Pull Down _ LCOISYS LCOI System	o. of Functions 8 6 4

Command : _

Fnn=HELP Fnn=Exit Fnn=Prompt Fnn=Create Fnn=Retr Fnn=Other Fnn=Dsp Tsk Id Fnn=Cancel Fnn=Msgs Fnn=More Optns Fnn=SortByDe

Note that only options which you are authorized to use will appear. Multilingual options will only appear in a multilingual partition. Impact List is described in **1.6.2 Using an Impact List**. The Current Task ID is only displayed if Task Tracking is active.

The Task ID locked to a process may be displayed or hidden by toggling the Disp/Hid Task ID function key. The Task ID is displayed to the right of the screen.

You can view the processes by Process name or Description using the SortByDesc/Process function key. The Description used for this sort is the one you enter via the Create or Maintain functions and not the description maintained by the multilingual attribute option.

The screen which controls the definition and amendment of functions using DEVELOPER(*ADVANCED) is the "Work with Functions" screen.

Work with Functions			
Process : PSLSYS Personnel System Main Menu Position to Current Task ID XXXXXXX Impact List Type options, press Enter			
2=Review/Change RDML 4=Delete 9=Rename 1/=Last Action Details			
Opt Function _ EMPLIST	Description Full Employee Listing		
_ ENROL	Enrol a New Employee		
_ INQUIRE	Browse/Maintain Employee and Skill Files		
_ PHONE	Employee Business Phone Number List		
_ SALARY	Produce Salary Commitment Reports		
_ SEARCH	Perform General Employee Search		

SNAME Telephone Number Search
 WINDOW Dept/Section/Employee Window Enquiry

Command : _____

Fnn=HELP Fnn=Exit Fnn=Prompt Fnn=Create Fnn=Retrieve Fnn=Cance Fnn=Msgs Fnn=Dsp Tsk Id Fnn=SortByDesc

Note only options which you are authorized to use will appear. Multilingual options will only appear in a multilingual partition. Impact List is described in 1.6.2 Using an Impact List. The Current Task ID is only displayed if Task Tracking is active.

The Task ID locked to a function may be displayed or hidden by toggling the Disp/Hid Task ID function key. The Task ID is displayed to the right of the screen.

You can view the functions by Function name or Description using the SortByDesc/Function function key. The Description used for this sort is the one you enter via the Create or Maintain functions and not the description maintained by the multilingual attribute option.

RDMLX functions will be flagged with **x** after the description.

1.6.1 Sort by Description

You can sort by the description of the Work with Fields, Work with System Variables, Work with Multilingual Variables, Work with Files and Work with Processes and Functions screens.

The description that is used for the sort is the description input via Creation or Maintain functions and not the description maintained by the multilingual attribute option.

1.6.2 Using an Impact List

It is possible to work with a subset of objects on the Work with Fields, Work with System Variables, Work with Multilingual Variables, Work with Files and Work with Processes screens.

The Impact List name is entered in the Impact List field. The List must exist and contain entries of the type being worked with. Only those entries which exist in the Impact List and in the repository will be displayed. By this means it is possible to have a list of objects which require changes and to work with just those objects on the Work with screens.

The actions taken against objects when using an Impact List subset are applied to the repository. For example, if using an Impact List and a field is changed, the changes are made to the field in the repository. Any changes or deletions made when using an Impact List are made to the repository. The Impact List is not changed.

When using an impact list, the 1.6.1 Sort by Description facility is not available. Impact List creation and maintenance is performed using the "Work with Impact Lists" on the Housekeeping Menu.

1.6.3 Function Keys

LANSA has several functions which can be performed by using function keys. These are:

Function	Function Key	Description
HELP		Display help text about the current function.
EXIT		Exit from LANSA immediately, or return to the Main System Menu immediately.
CANCEL		Cancel current function.
MESSAGES		Display all messages.
ADD		Add details to those currently displayed.
CHANGE		Change information currently displayed.
DELETE		Delete information currently displayed.
PROMPT		Prompt allowable values for a field

The column headed "Function key" has been left blank because the actual function key number assigned to the function is set during the LANSA installation procedures.

This allows the function key assignments within LANSA to match your installation standards. For instance, some sites use function key 1 for EXIT while others prefer key 3. Likewise, some sites use function key 8 for ADD while others use function key 6.

All LANSA screens show the function keys that can be used on lines 23 and 24 of the screen like this:

F1=Help F3=Exit F12=Cancel F14=Messages

This example shows function keys 1, 3, 12 and 14 assigned to the functions Help, Exit, Cancel and Messages respectively. If your function key assignments

are different, then this will be reflected in the display.

It should be noted that only the currently usable function keys are ever shown on any screen. For example, the ADD, CHANGE and DELETE function keys have no use on any of the system menus and thus are not displayed.

1.7 Messages and the Help Key

Messages are used extensively throughout the LANSA system. It is worthwhile spending some time understanding exactly how they are displayed and used.

Details are provided in these topics:

- 1.7.1 Messages
- 1.7.2 The HELP Key
- 1.7.3 Display Second Level Message Text
- 1.7.4 Display the Online Guide and Search the Dictionary
- 1.7.5 The Extended HELP Facility
- 1.7.6 Display User Defined HELP Text

1.7.1 Messages

When message(s) have been issued, the first message is always displayed on line 22 of the current screen (above the function key line).

Job 123456/QUSER/INVOICE submitted to job queue QGPL/QBATCH F1=Help F3=Exit F12=Cancel F14=Messages

A plus sign indicates that more messages follow this message.

If the plus sign appears then there are two ways of viewing the following message(s).

- The **first** is to move the screen cursor down to line 22 of the screen (the same line as the message) and then use the ROLL UP and ROLL DOWN keys to scroll backwards and forwards through all the messages. The last message can be identified by the absence of the "+" sign (no more messages follow).
- The **second** method is to use the **MESSAGES** function key. When used, a display like the example below will overlay the bottom of the current screen. Note that the "+" sign may still be present if more messages exist than can be displayed. In this case, use the ROLL UP and ROLL DOWN keys to scroll through all the messages.

CUSADD01 Enter New Customer Details

Cust no A6747 Cust name ACME ENGINEERING INCORPORATED Address 121 Smith Street

Note:

- When using the MESSAGES function key, not only are the messages issued by LANSA displayed, but also any messages that are currently on the message queue of the workstation at which you are working.
- To view the second level text of any message, position the screen cursor anywhere on the same line as the message and press the HELP key.
- To complete the review of messages press the ENTER/REC ADV key. The full screen that was in use when the messages function key was used will then be restored and normal processing can be resumed.
- While messages are being reviewed it is not possible to change any information shown anywhere on the screen.

1.7.2 The HELP Key

The HELP key has 3 uses within LANSA:

- **To display the "second level text" of any message**Second level text is available for every message and usually describes in more detail the cause or consequence of a message.
- **To display information about LANSA itself** This facility allows access to the LANSA Online Guide or the definition of any field or file in the LANSA data dictionary.
- **To display user written HELP text** This facility displays any HELP text that was provided to LANSA regarding the user defined fields or processes defined under LANSA.

1.7.3 Display Second Level Message Text

Second level message text can be displayed at any time by placing the screen cursor on the same screen line as the message and pressing the HELP key.

This applies equally to a message on line 22 of any display, or the special display that results from using the MESSAGES function key (previously described).

For example the second level text of a message like this on line 22 of a display could be viewed by positioning the screen cursor onto line 22 and pressing the HELP key:

+

Field name and alias name cannot be the same F1=HELP F3=Exit F12=Cancel F14=Messages

The resulting second level text display would look like this:

SECOND LEVEL MESSAGE DISPLAY

Msg ID: DCM0464 Sev: 00 Type: INFO 05/01/87 12:08:14 Job: User: Nbr: From pgm: DC@P9002 Inst: 0000 To pgm: DC@P1001 Inst: Field name and alias name cannot be the same

All field names and all alias names must be unique in combination. This means that no field can have the same name as another field or The same name as the field's alias name. Likewise a field's alias Name cannot be the same as another field's name or another field's Alias name. Change the alias name you have used, remove the alias Name altogether or change the field name.

1.7.4 Display the Online Guide and Search the Dictionary

If the HELP key (or the HELP function key) is pressed at any time while working within LANSA (rather than within a user application created by LANSA) and the screen cursor is NOT on the same line as a message then a screen showing related subjects will be displayed.

The list of subjects initially displayed will be "related" to the task being performed when the HELP key was used.

The example below shows what may result if the HELP key was used when reviewing or creating a new field definition:

SUBJECTS LANSA Online Guide - Selected Subjects

Type action next to each desired subject, then press Enter.

1=Display 5=Print

Action Subject

- _ Defining new fields in the LANSA dictionary
- _ Amending existing fields in the LANSA dictionary
- _ Impact of field definition changes on file definitions
- _ Impact of field definition changes on process definitions
- _ Creating a field reference file

Fnn=Ex HELP Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Contents

Working from the LANSA Online Guide's Selected Subjects screen you can:

- Invoke the extended HELP facility to search the LANSA data dictionary or the Online Guide's table of contents. Use the "Ex HELP" function key to invoke the "Extended HELP Facility". Refer to 1.7.5 The Extended HELP Facility for details.
- Display the entire table of contents of the Online Guide. To do this use the "Contents" function key. The resulting display will be identical to the one above, except that the title will indicate "All Subjects" and the entire table of

contents will be accessible.

- Display subject(s) from the Online Guide. Select the desired subject(s) from the list on display by entering a "1" beside the subject(s) in the column headed "Action", and then pressing enter. The full text of each subject chosen will then be displayed.
- Print subject(s) from the Online Guide. Select the desired subject(s) from the list on display by entering a "5" beside the subject(s) in the column headed "Action", and then pressing enter. The full text of each subject chosen will then be printed onto your current output queue.

To return to what was being done before the HELP key was pressed use the "Exit" function key or "Cancel" function key or press enter without selecting any subject(s) to be displayed or printed.

1.7.5 The Extended HELP Facility

The "Extended HELP Facility" is of particular use if you are using the more technical facilities of LANSA.

It is invoked by pressing the HELP function key on any LANSA screen, then using the "Ex HELP" (extended help) function key from the resulting "Selected Subjects" display.

This will cause a display like the example below to be presented to you:

EXTENDHELP Online Guide - Extended HELP Facilities

Enter Search Choice(s) and press the Enter key.

Search the Data Dictionary For fields and files containing : BILL OF LADING B/LADING BLADING

Search the Online Guide Table of Contents for Subjects containing : ABORT SUBSTRING

Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Contents

Working from the Extended HELP Facilities screen you can:

- Cause a search of the data dictionary for fields and files to be performed.
- Cause a search of the Online Guide's Table of contents to be performed.

Note that the search ignores upper and lower case character differences. In the screen example shown above, the user is requesting that the Data Dictionary be searched for occurrences of different strings that involve a "BILL OF LADING" number and that the Online Guide Table of Contents be searched for the strings "ABORT" and "SUBSTRING". You can also:

- Exit from the Online HELP facility and return to what you were doing before you pressed the HELP key by pressing the "Exit" function key.
- Cancel the Extended HELP facility and return to what you were doing before you invoked the extended help facility by pressing the "Cancel" function key.
- Request that the entire Online Guide Table of Contents be displayed by pressing the "Contents" function key.

1.7.6 Display User Defined HELP Text

If the HELP key or the HELP function key is pressed at any time while executing a **user written** LANSA **process** or **function** and the screen cursor is **NOT** on the same line as a message then user written HELP text associated with the process, function, or fields used by the function can be reviewed.

If the user is a LANSA developer, then both user written **technical and user** HELP text will be displayed. Otherwise, only user written **user** HELP text will be displayed.

Exactly what level of HELP text is displayed depends upon what the user was doing, and where the screen cursor was, when the HELP key was pressed. For instance:

- If the process main menu was on the screen then the HELP text associated with the **process** will be displayed initially.
- If a function screen format (resulting from an RDML DISPLAY or REQUEST command) was on the screen and the cursor was within the boundaries of a field on the screen, then the HELP text associated with the **field** will be displayed initially. The phrase "within the boundaries" means within the field itself or anywhere on its associated label, description or column headings.
- If a function screen format was on the screen and the cursor was not within the boundaries of a field, then the HELP text associated with the **function** will be displayed initially.

To illustrate this point, the following screen format shows what the screen would look like after the user has pressed the HELP function key while displaying a process main menu called CUSTOMERS.



:CUSTOMERS	Work with Customers	:
:	:	
: This is the customer co	ntrol system's main menu. By wor	king from :
: this menu you can perfe	orm just about any action you wish	n on any :
: customer that is recorde	ed in the system. :	C C
:	:	
: The sort of things that y	ou can do include :	
:	:	
: - Enter details of a new	v customer into the system	:
: - Review or change just	st about any detail of an existing cu	istomer :
: - Review or change the	e credit rating of a customer	•
: - Purge customers who	have not ordered any stock within	the :
:	:	
•	:	
:F2=Ex HELP F3=Exit	F12=Cancel F11=Index F14=Me	essages :
•	:	

Note: The user defined process level HELP text overlays the lower portion of the process menu in a "window". The advantage of the window over a full screen presentation is that it visually reminds the user what he/she was doing when the HELP function key was used.

While the HELP text window is on display you can only work with the facilities available within in it, these include:

- Invoke the extended HELP facility. Use the function keyed identified as "Ex HELP". Normally this function key is only available when reviewing the HELP text associated with a field. If used, the HELP text associated with the current process or function will be displayed.
- Exit from the HELP facility and return to what was being done when the HELP key was first used. To do this use the function key identified as "Exit".
- Cancel what is being done at the moment and return to what was being done previously. To do this use the function key identified as "Cancel".
- Cause the current process's HELP Index to be displayed. To do this use the

function key identified as "Index".

If the "Contents" function key was used on this display, a new window would overlay the lower portion of the screen.

Enter number of function required or place cursor on same line 1. Input new customer details 2. Review or change existing customer details 3. Review customer credit ratings 4. Purge old customer details
 1. Input new customer details 2. Review or change existing customer details 3. Review customer credit ratings 4. Purge old customer details HELP_TABLE Table of Contents : :Enter number of desired topic or put cursor on same line and press: :enter. : :
:HELP_TABLE Table of Contents : :Enter number of desired topic or put cursor on same line and press: : :enter. : :1. Process - Work with Customers : :1. Process - Work with Customer details : :1. Process - Work wor change existing customer details : :1. Process - Review or change credit ratings :
 4. Function - Review Customer credit ratings 5. Function - Purge old customer details 6. Field - Customer number 7. Field - Customer full name 8. Field - Customer search name 8. Field - Customer search name 1 1

Working from this table of contents, you may elect to:

• Review the HELP text associated with the current process.

- Review the HELP text associated with any of the functions defined within the current process.
- Review the HELP text associated with any field used by the function if a function was in use when the HELP function key was used.

1.8 Command Entry Line

If the LANSA developer screens have been entered with one of:

- DEVELOPER(*YES)
- DEVELOPER(*ADVANCED)
- DEVELOPER(Y)
- DEVELOPER(A)

and the user profile is not limit capability *YES, then a command entry line will be made available towards the bottom of the screen.

Commands entered here can be prompted using the Prompt function key, and previous commands can be retrieved with the Retrieve function key.

Commands are executed using only the normal IBM i authority of the current user profile and its group user profile, if it has one. No IBM i authority is adopted or inherited.

1.9 Select LANSA Objects via the Prompt Key

If the Prompt key is pressed at any time when the cursor is positioned on an object (field, file, process and function) where the prompt key has been enabled (indicated by a "+" at the end of the field) the "Selecting a LANSA object" facility will be invoked.

Depending on the type of object for which the prompt selection facility is requested, a pop-up window will be displayed enabling you to:

1.9.1 Select a Field to Work With

1.9.2 Select a File to Work With

1.9.3 Select a Process to Work With

1.9.4 Select a Function to Work With

These object prompt selection options allow you to search for a specific object and return the selected object to the program from which the prompt key selection facility was invoked.

Each of the four object prompt selection options is described in the following sections.

1.9.1 Select a Field to Work With

When a search has been requested for a field the Select Field to Work with screen will be displayed.

Note: The Current Task ID is only displayed if Task Tracking is active.

Use the Dsp/Hid Tsk ID function key to display the Task ID locked to a field. The Task ID replaces the display on the right of the screen.

From this screen you can nominate the field required in 3 different ways:

- **In full.** Enter the full name of the field required.
- **Partially.** Enter as much as desired of the partial name of the field. A search is made for all fields which generically match the name specified and the resulting list is displayed for subsequent selection.
- Leave blank. In this case a list of all fields in the data dictionary is displayed. The desired field can be selected from the list.

When you press Enter, a screen showing a list of fields is displayed.



From the Select Field to Work with screen you can:

- Alter the field name at the top right of the screen. The change can be to specify a full field name, a partial field name or a change to blanks. In all cases another search is made for the field required.
- **Select** one of the fields from the list by positioning the cursor on the required field and pressing Enter.

Use the Dsp/Hid Tsk ID function key to display the Task ID locked to a field. The Task ID replaces the display on the right of the screen.

1.9.2 Select a File to Work With

When a search has been requested for a file, the Select File to Work with screen is displayed.

Note: The Task ID is only visible if Task Tracking is active.

By using the Function Key to display the Task ID locked to a file the screen is amended with the Task ID displayed on the right of the screen.

From the Select File to Work with you can nominate the file required in 3 different ways:

- **In full.** Enter the full name of the file required.
- **Partially.** Enter as much as desired of the partial name of the file. A search is made for all files which generically match the name specified and the resulting list is displayed for subsequent selection.
- Leave blank. In this case a list of all files in the LANSA system is displayed. Select the desired file from the list.

When you press Enter, a screen showing a list of files is displayed.

.....

Select File to Work with Current Task ID XXXXXXXX :
Enter full or partial name of the file to search for or :
leave blank to select allOE :
File Library Description :
OEHEADER SALESLIB Sales Order Entry Header File :
OEDETAIL SALESLIB Sales Order Entry Detail File :
OEREPSC TABLELIB Sales Rep's Commission File :
Bottom :
Fnn=HELP Fnn=Cancel Fnn=Messages Fnn=Dsp Tsk Id :

From the Select File to Work with you can:

- Alter the file name at the top right of the screen. The change can be to specify a full file name, a partial file name or a change to blanks. In all cases another search is made for the file required.
- **Select** one of the files from the list by positioning the cursor on the required file and pressing Enter.

Use the Dsp/Hid Tsk ID function key to display the Task ID locked to a file. The Task ID replaces the display on the right of the screen.

1.9.3 Select a Process to Work With

When a search has been requested for a process the Select Process to Work with screen is displayed.

: Select Process to Work with : Enter full or partial name of the pr	 Current Task ID X ocess to search for	XXXXXXXX : or :	
	••••	- •	
•	:		
:	:		
	•		
•	•		
•	•		
:	:		
:	:		
: Fnn=HELP Fnn=Exit Fnn=Chg	Tsk Id Fnn=Cance	Fnn=Messages	:
•	•••••		

Note: The Current Task ID is only displayed if Task Tracking is active.

By using the Function Key to display the Task ID locked to a process the screen is amended with the Task ID displayed on the right of the screen.

From the Select Process to Work with you can nominate the process required in 3 different ways:

- **In full.** Enter the full name of the process required.
- **Partially.** Enter as much as desired of the partial name of the process. A search is made for all processes which generically match this name and the resulting list is displayed for subsequent selection.
- Leave blank. In this case a list of all processes in the LANSA system is displayed. Select the desired function from the list.

When you press Enter, a screen showing a list of processes is displayed.

.....

: Select Process to Work with Current Task ID XXXXXXXX :
: Enter full or partial name of the process to search for or :
: leave blank to select all
: Process Description :
: SALESYS Sales System :
: STOCKSYS Stock Control System :
: :
: :
: Bottom :
: Fnn=HELP Fnn=Exit Fnn=Dsp Tsk Id Fnn=Cancel Fnn=Messages :
·

From the Select Process to Work with you can:

- Alter the process name at the top right of the screen. The change can be to specify a full process name, a partial process name or a change to blanks. In all cases another search is made for the process required.
- **Select** one of the processes from the list by positioning the cursor on the required process and pressing Enter.

1.9.4 Select a Function to Work With

When a search has been requested for a function the Select Function to Work with screen is displayed.



From the Select Function to Work with you can nominate the function required in 3 different ways:

- **In full.** Enter the full name of the function required.
- **Partially.** Enter as much as desired of the partial name of the function. A search is made for all functions which generically match the name specified and the resulting list is displayed for subsequent selection.
- **Leave blank.** In this case a list of all functions in the LANSA system is displayed. Select the desired functions from the list.

When you press Enter, a screen showing a list of functions is displayed.

Note: The Current Task ID is only displayed if Task Tracking is active. Use the Function Key to display the Task ID locked to a function. The Task ID displayed on the right of the screen.

[:] Select Function to Work with Current Task ID XXXXXXXX :

From the Select Function to Work with you can:

- Alter the function name at the top right of the screen. The change can be to specify a full function name, a partial function name or a change to blanks. In all cases another search is made for the function required.
- **Select** one of the functions from the list by positioning the cursor on the required function and pressing Enter.

1.10 Last Action Details

Last action details are the date, time and user that last performed particular development actions against LANSA objects. Only actions taken via LANSA facilities after this feature is installed are captured.

- 1.10.1 What Actions are captured?
- 1.10.2 What Object Types are captured?
- 1.10.3 Online Access to Last Action Details
- 1.10.4 Last Action Details On Reports

1.10.1 What Actions are captured?

These are the actions for which details are captured:

Action	Abbreviation
Create	CR
Modify	MD
Compile	СМ
Export to IBM i	AE
Export to Workstation	WE
Check Out	СО
Use to Export	EX
Import	IM
Check In	CI
Export via PCMAINT	PE
Reorganize	RO
Use to Produce Report	PR
Task Tracking	TT

1.10.2 What Object Types are captured?

These are the LANSA object types for which details are captured:

LANSA Object Type Action Abbreviations

Field	CR, MD, AE, WE, CO, IM, CI, TT
System Variable	CR, MD, AE, WE, CO, IM, CI, TT
Multilingual Variable	CR, MD, AE, WE, CO, IM, CI, TT
File	CR, MD, CM, AE, WE, CO, IM, CI, TT
Process	CR, MD, CM, AE, WE, CO, IM, CI, TT
Function	CR, MD, CM, AE, WE, CO, IM, CI, TT
Template	CR, MD, AE, WE, CO, IM, PE, TT
Partition	CR, MD, PE
Task	CR, MD, PE
Export List	CR, MD, AE, EX, IM
The LANSA System	CR, MD, PE, RO
PC Definition	CR, MD, PE
PC User	CR, MD, PE
LANSA Document	CR, MD, PR

1.10.3 Online Access to Last Action Details

Last action details for LANSA objects is accessible online via either option 17 from the relevant Work With screen or if there is no Work With screen, via function key 17 from the relevant Display screen.

Last action details for LANSA object types: field, system variable, multilingual variable, file, process and function, is only accessible online via the advanced development menus.

In either case a screen similar to the following will be displayed:

DC@P820201 Review Last Action Details

Partition: XXX Object:XXXXXXXXX Extension: XXXXXXXXX Type: 2

Created by XXXXXXXXX on dd/mm/ccyy at hh:mm:ss Modified by XXXXXXXXXX on dd/mm/ccyy at hh:mm:ss Compiled by XXXXXXXXXX on dd/mm/ccyy at hh:mm:ss Exported to IBM i by XXXXXXXX on dd/mm/ccyy at hh:mm:ss Exported to Workstation by XXXXXXXXX on dd/mm/ccyy at hh:mm:ss Checked out by XXXXXXXXX on dd/mm/ccyy at hh:mm:ss Used to Export by XXXXXXXXX on dd/mm/ccyy at hh:mm:ss Imported by XXXXXXXXX on dd/mm/ccyy at hh:mm:ss Checked in by XXXXXXXXX on dd/mm/ccyy at hh:mm:ss Exported via PCMAINT by XXXXXXXXX on dd/mm/ccyy at hh:mm:ss Reorganised by XXXXXXXXX on dd/mm/ccyy at hh:mm:ss Used to Produce Report by XXXXXXXX on dd/mm/ccyy at hh:mm:ss Lock to task XXXXXXX by XXXXXXXXX on dd/mm/yy at hh:mm:ss Checked in by XXXXXXXXXX on dd/mm/ccyy at hh:mm:ss Locked to Task ID XXXXXXX by XXXXXXXXX on dd/mm/ccyy at hh:mn using XXXXXX

Fnn=HELP Fnn=Exit Fnn=Cancel Fnn=Messages F22=ShowLong

Together, the Partition, Object, Extension and Type define which LANSA object

for which Last Action Details are being displayed.

The body of the screen shows the formatted Last Action Details for actions which have last been carried out against the LANSA object since the Last Action Details feature was loaded onto this LANSA system.

The date is shown in system format. Both the date and time are shown with system defined separators.

From this screen it is possible to:

Show Long Name for the object by pressing F22. Refer to 1.13 Long Names for information.
1.10.4 Last Action Details On Reports

Last action details for LANSA objects is printed on existing reports in a format very similar to that shown via online access.

It appears on the reports printed for these LANSA object types:

- Field
- System Variable
- Multilingual Variable
- File
- Process
- Function
- Task

1.11 Task Tracking

In order to keep track of development within LANSA, task tracking is used. Task identifiers can be allocated to individual tasks, developers, or products (groups of objects) at partition level.

A developer chooses the task to work on when logging on to LANSA and the Task ID is displayed on all the relevant LANSA windows.

An object that is locked out by one task cannot be used by another task. "Objects" include:

- Fields
- Visual LANSA components
- Files
- Functions
- Processes
- Application templates
- System variables
- Multilingual variables.

You can review a history of all work performed on an object using the Task Identifier and you can use the task identifier to export all objects worked on for a task.

Task tracking can, and should be, tailored to your site's specific needs. You can decide how much control you want to have over the development effort, how much administration overhead you are willing to bear, how big is your team and whether development needs to be controlled by individual developer, by product or by both.

You must use Task Tracking when developing with Visual LANSA. This means that before you start your Visual LANSA development, you must first set up your Task Tracking requirements using LANSA on IBM i. When developing with Visual LANSA, in addition to locking by task, Task Tracking will also lock objects by PC Name on IBM i.

You will find a description of task tracking and how to set up your Task Identifiers in Task Tracking in *Housekeeping*.

1.12 Expedited Program Change (EPC) Log

LANSA upgrades are issued as EPCs (Expedited Program Changes).

The steps required and any pre-requisite EPCs are supplied with each EPC.

If an EPC requires another EPC to be loaded first, the load process will check for the existence of previously required EPCs. If a pre-requisite EPC is not present, the EPC loading procedure will fail.

You can see which EPCs have already been loaded using the F10=EPC Log function on the *LANSA Configuration Details* screen. You access the *LANSA Configuration Details* screen using the ABOUT parameter on the 1.4 LANSA Command. The EPC details can be displayed in EPC Number or Date/Time sequence and can be printed.

1.13 Long Names

Several different LANSA object types are able now to be referred to in RDMLX partitions by a name that is longer than the traditional ten characters. These object types are:

- Fields
- Files
- Logical views
- VL Components
- Processes
- Functions

Long names can only be given to these LANSA objects in Visual LANSA.

Further Information

1.13.1 Online Access to Long Names

1.13.1 Online Access to Long Names

The Long name for a LANSA objects is accessible online via F22=ShowLong on the Review Last Action Details screen for the object or from various screens in Housekeeping.

Once F22=ShowLong is pressed for an object, a window similar to the following will be displayed:

If F22=ShowLong is pressed for an object type that does not support long names, an error message stating this is returned to the screen.

If the object has not been given a long name, the Display Long Name screen is still displayed, but with the long name displayed as blank.

2. Fields

This chapter describes these topics:

2.1 The Field Control Menu

2.2 Select Field(s) to Be Worked With

2.3 Create New Field Definitions

2.4 Review or Change Field Definitions

2.5 Field Multilingual Attributes

2.6 Field Rules/Triggers

2.7 Range of Values Rule

2.8 List of Values Rule

2.9 Code / Table File Lookup Rule

2.11 Complex Logic Rule

2.12 Date Format / Range Rule

2.13 Triggers

2.14 Field Help Text

2.15 Print Field Definitions

2.16 System Variables

2.17 Print System Variable Definitions

2.18 Multilingual Variables

2.19 Print Multilingual Variable Definitions

2.20 Field Reference File

2.1 The Field Control Menu

Access to the field control components of LANSA is from the field control menu. The field control menu is accessed from the main system menu.



Enter number of function or place cursor on same line and press Enter.

- _ 1. Create new field definitions
 - 2. Review or change field definitions
 - 3. Review or change field multilingual attributes
 - 4. Review, change or create field validation checks
 - 5. Review, change or create field HELP text
 - 6. Print field definitions
 - 7. Create new system variable definitions
 - 8. Review or change system variable definitions
 - 9. Create new multilingual variables

- 10. Review or change multilingual variables
- 11. Create or re-create a field reference file
- 12. Exit from LANSA

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

2.2 Select Field(s) to Be Worked With

Many of the options that can be invoked from the field control menu will request the name(s) of the fields that you wish to work with (i.e. fields to be reviewed/changed, printed, etc). In all cases this field selection process is done in exactly the same way.

Initially a screen is presented that requests the name(s) of the fields to be worked with.

If you have been working with fields previously then the name will be pre-filled with the name of the **last** field you worked with.



Using this display you can nominate the field(s) required in 3 different ways:

- **In full.** Enter the full name of the field required. If the field is found then the option chosen from the field control menu will be directly invoked for the field specified.
- **Partially.** Enter as much as desired of the partial name of the field. A search is made for all fields which generically match the name specified and the resulting list is displayed for subsequent selection.
- Leave blank. In this case a list of all fields in the data dictionary is displayed from which the desired field(s) can be selected.

When you press Enter, a list of fields is displayed.

SELFIELD

Enter full or partial name of the data dictionary field(s) to be worked with or leave blank to select from all fields in dictionary . . . _____

	• .•	ттр
Sel Field De	escription	Type Len Dec
_ DEMAD1	Address line 1	A 25
_ DEMAD2	Address line 2	A 25
_ DEMAD3	Address line 3	A 25
_ DEMCAT	Product category	A 10
_ DEMCSC	Cost centre	A 3
_ DEMCUS	Customer number	A 6
_ DEMDDS	Division name in full	A 30
_ DEMDIV	Division number	A 2
_ DEMDUE	Date order due	P 6
_ DEMLIN	Order line	P 3
_ DEMNAC	Name code	A 6
_ DEMNAM	Name	A 25
_ DEMORD	Order number	A 6
_ DEMPCD	Post code	A 4
Fnn=Help Fnn=	=Exit Fnn=Cancel Fnn=	-Messages
I		0

In the Type column:

RDMLX fields are identified by an X

Visual LANSA components are identified with an *.

Using this display you can:

- **Alter** the field name at the top right of the screen. The change can be to specify a full field name, a partial field name or a change to blanks. In all cases another search is made for the field(s) required.
- **Select** one or more fields from the list by entering any non-blank character beside the field in the column labeled "Sel". If this is done then the option selected from the field control menu will be invoked for the field(s) selected.

2.3 Create New Field Definitions

The steps involved in creating a new field definition are:





2.3.1 Field Creation Options

The field creation options screen is presented first in every field definition cycle.

DC@P100103 Create Field Copy Options

To create this field by copying existing field specify details below:

Either copy field ______ which exists in the LANSA data dictionary, Or copy field ______ from file ______ in lib _____

Otherwise press ENTER to proceed without copying an existing field.

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Using the field creation options screen you can:

- Create a new field by copying the definition of a field already defined in the data dictionary.
- Create a new field by copying the definition of a field from an externally described IBM i file. This is a useful facility for copying fields from your own application field reference file in the LANSA data dictionary.

Note that if either copy option is used the information is only used to "pre-fill" the second screen in the cycle. The new field is not actually created until all processing is completed by the second screen.

2.3.2 Create a New Field Definition

The second screen presented when creating a new field definition looks like this:

DC@P100104 Add Data Dictionary Field
Field name:Type (A,P,S) / Keybd shift : _ / _Length / total digits:Number of decimal positions : _Reference field:
Description (0000) : Label (0000) : Column headings (0000) :
Output attributes list :
Edit code / Edit word
Default value
Optional alias name
System field :
Prompting process/function : /
Initial public access : <u>NORMAL</u> (ALL, NORMAL, NONE)
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen may be "pre-filled" with information. This occurs when:

- another field has been "copied" using options on the first screen.
- you are creating more than one field. Most input fields retain the values used to create the previous field.
- the new field has either been "copied" or refers to another field. The

Extended Options action box will be displayed. This will allow you to select certain copy options. For example, if the option to copy help text is selected, all the help text for the copied or referenced field will be copied to the help text for the new field.

Input Options when Creating a New Field

These input options apply when creating a new field or amending an existing one:

Field Name

Mandatory. If not specified field creation cycle ends.

- First character must be A->Z, \$, or @.
- Subsequent characters must be A->Z, 0-9, \$, @, # or _ (underscore).
- Use of @ anywhere in field name is not recommended.
- Use of # as first character is not allowed.
- No imbedded blanks are allowed in the field name.
- Use alias name for COBOL or PL/1 long names.
- Use of names of more than 6 characters in length is not recommended. Although LANSA will create and use files that contain field names of more than 6 characters in length it may be difficult to use such files in your own RPG application programs. Refer to the RPG *Programmer's Reference Guide*.
- Avoid the use of field names like SQLxxx, as this may cause problems when used in functions that use SQL (Structured Query Language) facilities. (i.e. Command SELECT_SQL.)

Туре

Mandatory unless reference field is specified. Must be A (alphanumeric), S (signed) or P (packed) for RDML fields. Additional field types for RDMLX fields are H (char), Z (DateTime), B (binary), E (date), M (time), I (integer), F (float), 1 (string), 2 (VarBinary), 3 (CLOB) and 4 (BLOB).

Keyboard Shift

Optional. If not specified, defaults to blank for an alphanumeric field and Y for a numeric field. Valid keyboard shifts are:

Keyboard Shift	Data Type		
Blank	Alpha/Numeric		

Х	Alpha
А	Alpha
Ν	Alpha/Numeric
S	Numeric
Υ	Numeric
W	Alpha
Ι	Alpha/Numeric
D	Alpha/Numeric
М	Alpha
J	Alpha (DBCS)
E	Alpha (DBCS)
0	Alpha (DBCS)

Refer to the IBM manual *Data Description Specifications* for details. Position 35 for display files is the entry that should be reviewed.

Length

Mandatory unless reference field is specified.

- Length is number of characters for type A.
- Total number of digits (including decimals) for type P or S.
- For type P this is NOT the same as the field byte length.
- The use of odd numbered lengths is recommended for type P or S.
- For type A must be in range 1 to 256.
- For type P or S must be in range 1 to 30.

Decimal Positions

Mandatory for type P or S unless reference field is specified. Ignored for type A (always set to zero). Must be in range 0 to 9. Must be less than or equal to length (total number of digits).

Reference Field

Optional. This is the name of the field which should be "referred" to, to determine the type, keyboard shift, length, number of decimal positions, default value, edit code, edit word, input attributes and output attributes that are to be used in the definition of this new field.

These values are protected while the reference field is specified and they are automatically updated if the reference field is changed. Initially the prompt process/function is inherited but you can change these details if required.

The reference field must already exist in the dictionary before you can enter it here.

If the flag *IMPREFFLDNOPROP is in system data area DC@OSVEROP, the input and output attributes will not be protected but will still be updated if the reference field is changed. This is to provide compatibility with previous versions. For more information, refer to Version Dependency Data Area DC@OSVEROP.

Reference fields are mostly used to specify an attribute that is common to many similar fields. For example, a shipping company deals with many different types of ports, so they could define a field called PORT as an alphanumeric field of 5 characters.

Subsequent port fields such as PORTL (port of loading) or PORTD (port of discharge) would "refer" to PORT. If the definition of PORT was changed from 5 to 6 characters, both PORTL and PORTD would be changed automatically.

Description

Optional. If not specified, field name is used as default. Use of CUA standards for identification are recommended.

Label

Optional. If not specified, field name is used as default. Use of CUA standards for identification are recommended.

Column Headings

Optional. If not specified, field name is used as default. Use of CUA standards for identification are recommended.

Output Attributes List

Optional. If not specified defaults to either alpha (type A) or numeric (type P or S) system default values. Further information on output attributes is contained in DEFINE Parameters and OVERRIDE Parameters in the Technical Reference.

Valid output attributes are:

Attribute	Description / Comments	Α	Р	S
GRN	Display with color green.	Y	Y	Y
WHT	Display with color white.	Y	Y	Y
RED	Display with color red.	Y	Y	Y
TRQ	Display with color turquoise.	Y	Y	Y
YLW	Display with color yellow.	Y	Y	Y
PNK	Display with color pink.	Y	Y	Y
BLU	Display with color blue.	Y	Y	Y
BL	Display blinking.	Y	Y	Y
CS	Display with column separators.	Y	Y	Y
HI	Display in high intensity.	Y	Y	Y
ND	Non-display (hidden field).	Y	Y	Y
SREV	Store in reverse format. This is a special attribute provided for bidirectional languages. Refer to the <i>LANSA Multilingual Application Design Guide</i> .	Y	Ν	N
SBIN	Store in binary format. This special attribute is provided for fields that need to contain embedded packed or signed fields. Refer to the Use of Hex Values, Attributes, Hidden/Embedded Decimal Data in the LANSA Application Design Guide.	Y	Ν	Ν
Urxx	User Defined Reporting Attribute. Provides access to DDS statements for printer files. Refer to User Defined Reporting Attributes.	Y	Y	Y
HIND	HINDI Numerics. Display using HINDI numerals. Refer to Hindi Numerics for details.	N	Y	Y
CBOX *	Check Box	Y	Ν	N
RBnn *	Radio Button	Y	Ν	N
PBnn *	Push Button	Y	Ν	Ν

Attributes marked with * represent the field with the corresponding GUI WIMP construct. Refer to Guidelines, Hints and Tips in *GUI WIMP Constructs* for more information.

In partitions that comply with SAA/CUA guidelines, these attributes may be used as well (and are in fact preferred to those in the previous list):

Attribute	Description / Comments
ABCH	Action bar and pull-down choices
PBPT	Panel title
PBPI	Panel identifier
PBIN	Instructions to user
PBFP	Field prompt/label or description details
PBBR	Brackets
PBCM	Field column headings
PBGH	Group headings
PBNT	Normal text
PBET	Emphasized text
PBEN	Input capable field (normal)
PBEE	Input capable field (emphasized)
PBCH	Choices shown on menu
PBSC	Choice last selected from menu
PBUC	Choices that are not available
PBCN *	Protected field (normal)
PBCE *	Protected field (emphasized)
PBSI	Scrolling information

PBSL	Separator line
PBWB	Pop-up window border
FKCH	Function key information

*Normally only PBCN and PBCE would be specified as output attributes. Refer to the IBM i SAA/CUA Implementation in the *LANSA Application Design Guide* for details of these attributes.

Note that only one color can be specified for a field. Use of colors may affect other attributes. Refer to IBM manual *Data Description Specifications* for details. Keywords that should be reviewed are COLOR and DSPATR.

Input Attributes List

Optional. If not specified defaults to either alpha (type A) or numeric (type P or S) system default values. Further information on input attributes is contained in DEFINE Parameters and OVERRIDE Parameters in the *Technical Reference*.

Attribute	Description / Comments	Α	Р	S
AB	Allow to be blank.	Y	Y	Y
ME	Mandatory entry check required.	Y	Y	Y
MF	Mandatory fill check required.	Y	Y	Y
M10	Modulus 10 check required.		Y	Y
M11	Modulus 11 check required.		Y	Y
VN	Valid name check required.	Y		
FE	Field exit key required.	Y	Y	Y
LC	Lowercase entry allowed. If you do NOT set this attribute, refer also to <i>PC Locale uppercasing</i> <i>requested</i> in Review or Change a Partition's Multilingual Attributes	Y		
RB	Right adjust and blank fill.	Y	Y	Y

Valid input attributes are:

RZ	Right adjust and zero fill.	Y	Y	Y
RL	Move cursor right to left.	Y	Y	Y
RLTB	Tab cursor right/left top/bottom Valid in SAA/CUA partitions only. Affects all screens in function.	Y	Y	Y
GRN	Display with color green.	Y	Y	Y
WHT	Display with color white.	Y	Y	Y
RED	Display with color red.	Y	Y	Y
TRQ	Display with color turquoise.	Y	Y	Y
YLW	Display with color yellow.	Y	Y	Y
PNK	Display with color pink.	Y	Y	Y
BLU	Display with color blue.	Y	Y	Y
BL	Display blinking.	Y	Y	Y
CS	Display with column separators.	Y	Y	Y
HI	Display in high intensity.	Y	Y	Y
ND	Non-display (hidden field).	Y	Y	Y
RA	Auto record advance field	Y	Y	Y
SREV	Store in reverse format. This is a special attribute provided for bidirectional languages. Refer to the <i>LANSA Multilingual Application Design Guide</i> .	Y	N	Ν
SBIN	Store in binary format. This special attribute is provided for fields that need to contain embedded packed or signed fields. Refer to the Use of Hex Values, Attributes, Hidden/Embedded Decimal Data in the LANSA Application Design Guide.	Y	Ν	Ν
HIND	HINDI Numerics. Display using HINDI numerals. Refer to Hindi Numerics for details.	N	Y	Y
CBOX	Check Box	Y	Ν	Ν
RBnn	Radio Button	Y	Ν	Ν

PBnn	Push Button	Y	Ν	Ν
DDxx	Drop Down. Represents the field with the corresponding GUI WIMP construct. Refer to Guidelines, Hints and Tips in GUI WIMP Constructs for more information.	Y	Ν	Ν

In partitions that comply with SAA/CUA guidelines these attributes may be used as well (and are in fact preferred to those described above).

Attribute	Description / Comments
ABCH	Action bar and pull-down choices
PBPT	Panel title
PBPI	Panel identifier
PBIN	Instructions to user
PBFP	Field prompt/label or description details
PBBR	Brackets
PBCM	Field column headings
PBGH	Group headings
PBNT	Normal text
PBET	Emphasized text
PBEN *	Input capable field (normal)
PBEE *	Input capable field (emphasized)
PBCH	Choices shown on menu
PBSC	Choice last selected from menu
PBUC	Choices that are not available
PBCN	Protected field (normal)
PBCE	Protected field (emphasized)

PBSI	Scrolling information
PBSL	Separator line
PBWB	Pop-up window border
FKCH	Function key information

* Normally only PBCN and PBCE would be specified as output attributes. Refer to the IBM i SAA/CUA Implementation in the LANSA Application Design *Guide* for details of these attributes. Note also that only one color can be specified for a field. Use of colors may affect other attributes. Refer to IBM manual Data Description Specifications for details. Keywords that should be reviewed are CHECK, COLOR and DSPATR.

Edit Code

Optional. However, use of edit codes for all numeric fields (type S or P) is strongly recommended. Edit codes supported under this version of LANSA are shown in Standard Field Edit Codes in the *Technical Reference Guide*.

Edit Word

Optional. Use of edit words should only be attempted by experienced users as the validity checking done by LANSA is unsophisticated. Invalid edit words may pass undetected into the system and cause subsequent failures when attempting to create database files or compile programs.

Note: Edit word processing involving floating currency symbols is handled differently by the operating system for screens and reports. If this problem occurs it is best overcome by the use of a virtual field for report production and only using the real field for screen work.

Refer to IBM manual *Data Description Specifications* for details. See keyword EDTWRD.

Default Value

Optional. If not specified, it defaults to *BLANKS (blanks) for alphanumeric fields (type A) and *ZERO (zero) for numeric fields (type P or S). Default value specified can be:

• A system variable such as *BLANKS, *ZERO, *DATE or any other

specifically defined at your installation.

- An alphanumeric literal (in quotes) such as 'STATUS', 'JONES' or 'Australia'.
- A numeric literal such as 1, 10.43, -1.341217.

Optional Alias Name

Optional. If specified it must NOT:

- Be the same as any other field's name (including the field that is being created or amended).
- Be the same as any other field's alias name.

In other words, all field names and all alias names must form a unique list of names. The alias name facility is provided primarily for installations that use the COBOL or PL/1 programming languages. Refer to the specific IBM supplied program reference manuals for the use of the ALIAS keyword. Name must conform to field naming conventions like the field name. COBOL or PL/1 language naming conventions are NOT checked.

System Field

Mandatory, but always pre-filled to NO. Allowable values are:

YES: This field is a system field and cannot be deleted from the data dictionary unless this field is specifically changed to value "NO". This value is used by LANSA to identify all the required LANSA system fields so that they will not be accidentally deleted from the data dictionary. It is also used to identify all the LANSA system fields that may be copied when creating a new LANSA system partition.

NO: This field is not a system field and can therefore be deleted from the data dictionary if there is no reason for the delete operation being refused (e.g. field is used in a file definition, etc).

Prompting Process/Function

Optional. If specified, these values nominate an RDML process and function that should be invoked to handle a "prompt request" made against the field being defined or changed. When specifying the name associated with a prompting process and function it is recommended that the process name be nominated as *DIRECT. This indicates to the prompt control procedures that the nominated function can be called in "direct" mode, without having to go through the associated process "controller".

Using *DIRECT has positive performance benefits, but when a prompting function is to be invoked this way it must use the FUNCTION

OPTIONS(*DIRECT) command. Refer to CALL Comments / Warnings and FUNCTION Examples in the *Technical Reference Guide* for details of direct mode invocation of functions. A "prompt request" is made against a field when the user positions the screen cursor into a field, on its label, or on one of its column headings, and then uses the PROMPT function key. Normally the prompt function key is F4, but it may be assigned differently on your system. When a reference field has been specified, initially the prompt process/function is inherited from the referenced field, but you can change it if required.

If the reference field's prompt process/function is changed, any of the fields referring to it which have the same prompt process/function (before the referenced field is changed) also have their prompt process/function updated.

For technical details of Prompt_Key Processing please refer to the *Technical Reference Guide*. For examples of prompting processes and functions please refer to What Happens When the PROMPT Key is Used in the *LANSA Application Design Guide*.

Initial Public Access

Mandatory, but always pre-filled to NORMAL. Allowable values are:

NORMAL: Other users can use this field, but cannot change or delete its definition in/from the data dictionary.

ALL: Other users can use this field and can also change or delete its definition in/from the data dictionary.

NONE: Other users cannot use this field, nor can they review, change or delete its definition in/from the data dictionary.

Output Stamping Attributes

The following stamping attributes can be used (in the data dictionary only) to indicate that certain fields in file definitions should be automatically "stamped" during INSERT and/or UPDATE operations:

Attribute	Description / Comments		Р	S
USRC	When a field with this attribute is in a file it should always contain the user profile name of the user that created the record.		Ν	Ν
	A user profile name may be up to 256 characters in length if *LONG_USER_AUDIT is enabled when the file is compiled, otherwise it may be up to 10 characters in length. If *LONG_USER_AUDIT is enabled, the user name will be the audit user (up to 256 characters) if the SET_SESSION_VALUE USER_AUDIT has been set, otherwise the current authenticated user name is used if available, otherwise the current user name is used. If *LONG_USER_AUDIT is not enabled then the user name will be the audit user (up to 10 characters) if the SET_SESSION_VALUE USER_AUDIT has been set, otherwise the current user name is used. If running a function through SuperServer, USRC contains the user name used to connect to the server. Refer to USCR Examples to see how this			
	name can be used.			
USRU	When a field with this attribute is in a file, it should always contain the user profile name of the user that last updated the record or be blank, if the record has never been updated. A user profile name may be up to 256 characters	Y	Ν	Ν
	in length if *LONG_USER_AUDIT is enabled when the file is compiled, otherwise it may be up			

User and Job Stamping Attributes

	to 10 characters in length. If *LONG_USER_AUDIT is enabled then the user name will be the audit user (up to 256 characters) if the SET_SESSION_VALUE USER_AUDIT has been set, otherwise the current authenticated user name is used if available, otherwise the current user name is used. If *LONG_USER_AUDIT is not enabled then the user name will be the audit user (up to 10 characters) if the SET_SESSION_VALUE USER_AUDIT has been set, otherwise the current user name is used. The user name will be the audit user if the SET_SESSION_VALUE USER_AUDIT has been set, otherwise the current user name is used.			
USRX	When a field with this attribute is in a file it should always contain the user profile name of the user that either created or last updated the record. A user profile name may be up to 256 characters in length if *LONG_USER_AUDIT is enabled when the file is compiled, otherwise it may be up to 10 characters in length. If *LONG_USER_AUDIT is enabled then the user name will be the audit user (up to 256 characters) if the SET_SESSION_VALUE USER_AUDIT has been set, otherwise the current authenticated user name is used if available, otherwise the current user name is used. If *LONG_USER_AUDIT is not enabled then the user name will be the audit user (up to 10 characters) if the SET_SESSION_VALUE USER_AUDIT has been set, otherwise the current authenticated user name is used. If *LONG_USER_AUDIT is not enabled then the user name will be the audit user (up to 10 characters) if the SET_SESSION_VALUE USER_AUDIT has been set, otherwise the current user name is used. The user name will be the audit user if the SET_SESSION_VALUE USER_AUDIT has been set, otherwise the current user name is used.	Y	N	Ν
JNMC	When a field with this attribute is in a file it	Y	Ν	Ν

	should always contain the name of the IBM i job that was used to create the record. A job name may be up to 10 characters in length			
JNMU	JNMU When a field with this attribute is in a file it should always contain the name of the IBM i job that was last used to update the record, or, be blank if the record has never been updated.		Ν	Ν
JNMX	NMX When a field with this attribute is in a file it should always contain the name of the IBM i job that was used to either create or last update the record.		Ν	Ν
JNRC	When a field with this attribute is in a file it should always contain the number of the IBM i job that was used to create the record. A job number may be up to 6 characters in length	Y	Ν	Ν
JNRU	When a field with this attribute is in a file it should always contain the number of the IBM i job that was last used to update the record, or, be blank if the record has never been updated.	Y	Ν	Ν
JNRX	When a field with this attribute is in a file it should always contain the number of the IBM i job that was used to either create or last update the record.	Y	Ν	Ν

Time and/or Date Stamping Attributes

Attribute	Description / Comments	Α	Р	S
TIMC	When a field with this attribute is in a file it should always contain the time that the record was created. All time stamps are 6 digits long in the format HHMMSS.	Y	Y	Y
TIMU	When a field with this attribute is in a file it should always contain the time that the record	Y	Y	Y

	was last updated, or, contain zeroes if the record has never been updated.			
TIMX	When a field with this attribute is in a file it should always contain the time that the record was created or the time that it was last updated.	Y	Y	Y
DATC	When a field with this attribute is in a file it should always contain the date that the record was created. All DAT date stamps are 6 digits long in system date format.	Y	Y	Y
DATU	When a field with this attribute is in a file it should always contain the date that the record was last updated, or, contain zeroes if the record has never been updated.	Y	Y	Y
DATX	When a field with this attribute is in a file it should always contain the date that the record was created or the date that it was last updated.	Y	Y	Y
TDSC	When a field with this attribute is in a file it should always contain the time/date that the record was created. All TDS time/date stamps are 12 digits long in the format HHMMSSDDDDDD where HHMMSS is the time and DDDDDD is the date in system date format.	Y	Y	Y
TDSU	When a field with this attribute is in a file it should always contain the time/date that the record was last updated, or, contain zeroes if the record has never been updated.	Y	Y	Y
TDSX	When a field with this attribute is in a file it should always contain the time/date that the record was created or the time/date that it was last updated.	Y	Y	Y
CDTC	When a field with this attribute is in a file it should always contain the date the record was created. All CDT date stamps are 8 digits long in system date format.	Y	Y	Y

CDTU	CDTU When a field with this attribute is in a file it should always contain the 8 digit date that the record was last updated, or contain zeroes if the record has never been updated.		Y	Y
CDTX	CDTX When a field with this attribute is in a file it should always contain the 8 digit date that the record was created or last updated.		Y	Y
TCDC	FCDC When a field with this attribute is in a file it should always contain the time/date the record was created. All TCD time/date stamps are 14 digits in format HHMMSSDDDDDDDD where HHMMSS is the time and DDDDDDDD is the date in system format.		Y	Y
TCDU	When a field with this attribute is in a file it should always contain the time/date that the record was last updated, or contain zeroes if the record has never been updated.	Y	Y	Y
TCDX	When a field with this attribute is in a file it should always contain the time/date that the record was created or last updated.	Y	Y	Y
YMDC	When a field with this attribute is in a file it should always contain the date that the record was created. All YMD dates are 6 digits in YYMMDD format.	Y	Y	Y
YMDU	When a field with this attribute is in a file it should always contain the date that the record was last updated, or contain zeroes if the record has never been updated.	Y	Y	Y
YMDX	When a field with this attribute is in a file it should always contain the date the record was created or last updated.	Y	Y	Y
TYDC	When a field with this attribute is in a file it should always contain the time/date that the record was created. All TYD time/dates are 12	Y	Y	Y

	digits in HHMMSSYYMMDD format.			
TYDU	When a field with this attribute is in a file it should always contain the time/date that the record was last updated, or contain zeroes if the record has never been updated.	Y	Y	Y
TYDX	YDX When a field with this attribute is in a file it should always contain the time/date that the record was created or last updated.		Y	Y
CYDC	When a field with this attribute is in a file it should always contain the date that the record was created. All CYD dates are 8 digits in YYYYMMDD format.	Y	Y	Y
CYDU	When a field with this attribute is in a file it should always contain the date that the record was last updated, or contain zeroes if the record has never been updated.	Y	Y	Y
CYDX	When a field with this attribute is in file it should always contain the date that the record was created or last updated.	Y	Y	Y

Process and Function Stamping Attributes

Attribute	Description / Comments	A	Р	S
FUNC	When a field with this attribute is in a file it should always contain the name of the RDML function that was used to create the record. A function name may be up to 7 characters in length.	Y	Ν	Ν
FUNU	When a field with this attribute is in a file it should always contain the name of the RDML function that was last used to update the record or be blank if the record has never been updated.	Y	Ν	Ν
FUNX	When a field with this attribute is in a file it	Y	Ν	Ν

	should always contain the name of the RDML function that was used to either create or last update the record.			
PROC	When a field with this attribute is in a file it should always contain the name of the process that was used to create the record. A process name may be up to 10 characters in length.	Y	Ν	Ν
PROU	When a field with this attribute is in a file it should always contain the name of the process that was last used to update the record, or be blank, if the record has never been updated.	Y	Ν	Ν
PROX	When a field with this attribute is in a file it should always contain the name of the process that was used to either create or last update the record.	Y	Ν	Ν

USCR Examples

The current user (USCR) is the user name used to start the LANSA application. Refer to Output Stamping Attributes for further information.

USRC showing the users running locally with ***LONG_USER_AUDIT** off:

Start Form As	John
Windows log in	MYDOMAIN\John
Connect As	N/A
Connect To	N/A
Value	John

Start Form As	Mary
Windows log in	MYDOMAIN\John
Connect As	N/A
Connect To	N/A
Value	Mary

USRC show the users running locally with ***LONG_USER_AUDIT** on:

Note that the value will be truncated if the USRC field is too short.

Start Form As	John
Windows log in	MYDOMAIN\John
Connect As	N/A
Connect To	N/A
Value	John@MYDOMAIN

Start Form As	Mary
Windows log in	MYDOMAIN\John
Connect As	N/A
Connect To	N/A
Value	John@MYDOMAIN

USRC if running via LANSA SuperServer:

Start Form As	Mary
Windows login	MYDOMAIN\John
Connect As	David
Connect To	IBM i
Value	David

Start Form As	John
Windows login	MYDOMAIN\John
Connect As	David
Connect To	IBM i
Value	David

2.3.3 Reference Field Update Confirmation

DC@P100101 **Display Data Dictionary Field** Field name Type (A,P,S) / Keybd shift : B / B Length / total digits :999 Number of decimal p Reference field : Fields to be updated by current changes : : Press enter to confirm, F12 to cancel : Description (: (: FIELD1 First referencing field Label Column headings (: FIELD2 Second referencing field : Third referencing field : FIELD3 Output attributes 1 : :BB BBBB Input attributes li : :BB BBBB Edit code / Edit wo : : Default value : Optional alias name : F1=Help Fnn=Cancel F14=Messages : System field • Prompting process/function : BBBBBBBBBB / BBBBBBB

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

When updating a field which is referred to by at least one other field, a list of the referencing fields is displayed for you to confirm that they are also to be updated. This confirmation is only displayed when one of the values which are inherited by the referencing field is changed. For example, if only the field description is changed the confirmation display would not appear.

- Use the *Enter* key to update the current field and all its referencing fields as shown in the list.
- Use the F12 *Cancel* key to redisplay the current field without updating the current field or its referencing fields.
2.3.4 Extended Copy or Reference Field Options

The Reference or Copy Field Extended Options screen which is presented only if the new field has been successfully created and has been copied from an existing data dictionary field, or refers to an existing data dictionary field, looks like this:

Display Data Dictionary Field DC@P100101 Field name Type (A,P,S) / Keybd shift : B / B Length / total digits :999 Number of decimal p Reference field : Copy Field Extended Options : Description (: Use cursor to select copy option(s): (: Copy help text Label Column headings (: Copy all Rules/Triggers : Selectively copy Rules/Triggers : Copy multilingual definitions : BBBB BBBB Output attributes l : Input attributes li : Then use cursor to initiate action : BBBB BBBB Edit code / Edit wo : Perform selected copy requests : Default value : Cancel request Optional alias name :F1=Help Fnn=Cancel F14=Messages : System field . Prompting process/function : BBBBBBBBBB / BBBBBBB

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

Using the Display Data Dictionary Field screen you can:

- Use the cursor firstly to select any of the displayed options.
- Use the cursor to specify whether to perform the selected copy requests or to cancel the request.

2.3.5 Select Validation Rules for Copy

If the option to selectively copy validation rules is requested, this screen will be displayed:

F12=Cancel

Using this display you can:

- **Note** for fields which are being changed, if any rule already exists for the field being copied to, the message "RULE EXISTS" will be displayed. The rule can still be selected, but it will overwrite the existing rule.
- Cancel by using function key 12.

2.4 Review or Change Field Definitions

The steps involved in the review or change of a field definition are:





2.4.1 Detailed Display of a Field Definition

When field definitions have been chosen for review or change this display is presented for each field chosen:

DC@P100101 **Display Data Dictionary Field** Field name Type (A,P,S) / Keybd shift : B / B Length / total digits : 999 Number of decimal positions : 9 Reference field Description Label Output attributes list : BBBB BBBB BBBB BBBB BBBB BBBB BBBB Input attributes list : BBBB BBBB BBBB BBBB BBBB BBBB BBBB E Default value Optional alias name : BBB System field Prompting process/function : BBBBBBBBBB / BBBBBBB Fnn=Help Fnn=Cancel Fnn=Messages Fnn=References Fnn=Change Fnn=D

Using the Display Data Dictionary Field display you can:

• Use the CHANGE function key to change the field definition. Changes to the field definition must match the criteria described in the section "Creating a new field definition".

Note that if the field has been successfully changed and a reference field exists then the Reference Field Extended Options screen will be presented

for selection of copy options (see section on creating new field definitions for more information).

- Use the REFERENCES function key to display a list of all the fields which refer to this field for their definition. This function key is only displayed when the current field is referred to by at least one other field. When a referenced field is updated this list is displayed for confirmation that all the referencing fields and the current field should be updated.
- Use the DELETE function key to delete the field definition. After the delete key has been used it will be necessary to confirm the delete request by entering YES on This screen:

DC@P100103	Delete Data Dictionary Field
Field name: XXX Description: XXX	XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Confirm this field	is to be deleted.
Delete this field	YES, NO

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

If a Web visual component exists for the field to be deleted, a warning message will be displayed. If the field is deleted, the web details will also be deleted. It should be noted that there are many reasons for a delete operation to be refused. For example, the field is used in a file definition, other fields refer to this field and so on. In all cases the message issued will describe the reasons for the delete request to be rejected.

2.5 Field Multilingual Attributes

In a multilingual partition, field multilingual attributes may be specified for each language that is defined within the partition.

Field multilingual attributes include the field label, description and column headings.

2.5.1 Steps to Review or Change Field Multilingual Attributes

2.5.2 Review or Change a Field's Multilingual Attributes

2.5.1 Steps to Review or Change Field Multilingual Attributes

The steps involved in the review or change of a field's multilingual attributes are:



2.5.2 Review or Change a Field's Multilingual Attributes

When the option to review or change a field's multilingual attributes on the field definition menu is chosen, a screen similar to this will result:

DC@P100701	Multilingual Field - LRTB	
<u>Field : EMPNO</u>		
<u>Description / Label</u> Personeels nummer Personeels nr	<u>Column Headings</u> Person. Nummer	<u>Language</u> (DUT)
Employee Number Employee no	Employ Number	(ENG)
Codice Impiegato N. Impiegato	Numero Impiegato	(ITL)

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

From this screen you can:

- Review the multilingual information associated with a field.
- Change the multilingual information associated with a field. Use the CHANGE key to place the screen in change mode and make the desired changes.

This panel will be repeated for each language group i.e. all LRTB (Latin languages), DBCS (Ideographic or Double Byte Character Set languages) and RLTB (Bi-directional languages).

Input Options

These input options apply to changing a field's multilingual attributes:

Description

Optional. If not specified, field name is used as default.

Label

Optional. If not specified, field name is used as default.

Column Headings

Optional. If not specified, field name is used as default. All column headings should be centered in the length of the longest column heading for all languages. Refer to the *LANSA Multilingual Application Design Guide* for further details.

2.6 Field Rules/Triggers

The types of rules that can be worked with are:

- A range of values rule: allows a field to be checked against various ranges of values. For instance, an Australian post code should be in one of the ranges 2000 to 2999, 3000 to 3999, etc.
- A list of values rule: allows a field to be checked against a list of values. For instance, an Australian state should be in the list QLD, NSW, VIC, etc.
- A code/table/file lookup rule: allows a field to be "looked up" in a code file or table. For instance, a product number may be looked up in the product master file to check that it is a valid number.
- A simple logic rule: allows simple conditions to be evaluated to check a field. For instance, "item weight must be less than (item volume * 10.643)" may be a check used in an order entry system.
- A complex logic rule: allows complex validation checking to be performed by your own LANSA functions or 3GL application programs. For instance, the validation of a "due date" may be done via a function or program that can account for public holidays, weekends, etc.
- A date format/range rule: allows a field to be validated as a date in a certain format (DDMMYY,MMDDYY,etc) and in a certain range. For instance, a "date order due" may have to be in format DDMMYY and no more than 90 days into the future.
- **Trigger:** allows a function to be linked to a field at either dictionary or file level which will be "triggered" under the conditions specified in the field trigger itself.

2.6.1 Rule and Trigger Levels and Hierarchy

Rules and triggers exist in a 3 level hierarchy:



- **Dictionary level rules/triggers** always apply to a field under all circumstances. For instance, a dictionary level rule that says a post code must be in the range 2000 to 2999 will ensure that no matter when or how the post code field is used it will ALWAYS have to be in the range 2000 to 2999.
- **File level rules/triggers** apply only to the field within the file. For instance, FILE1 may have a rule that says field DATDUE must be in the format DDMMYY, however FILE2 may have a check that says field DATDUE must be in the format YYMMDD. This may be perfectly valid.
- **Process level rules** apply only to the field as used within a process. For instance, process ORDERS may add the rule that field DATDUE (used in preceding example) must be within the next 30 days. This check will be done in addition to the file level checks associated with file FILE1.

When working with a field at any level of the hierarchy you CANNOT VIOLATE a rule or trigger that exists at a higher level in the hierarchy.

The level in the hierarchy you are working at is easy to remember:

- If you are working from the field control menu then you are working at the DICTIONARY level.
- If you are working from the file definition menu (i.e. working with a file definition) then you are working at the FILE level.
- If you are working from the process definition menu (i.e. working with a

process or function definition) then you are working at the PROCESS level.

2.6.2 Diagram of the Field Rule and Trigger Processes

This diagram represents the processing flow used when working with **DICTIONARY LEVEL, FILE LEVEL or PROCESS LEVEL** field rules and triggers.





2.6.3 Effect of the Rule and Trigger Hierarchy

This table shows the effect of the rule and trigger hierarchy on the types of action (Display or Amend) that you can take when working with rules and triggers at each level:

Working Level	DICTIONARY RULE/TRIGGER	FILE RULE/TRIGGER	PROCESS RULE
Dictionary	Display, Amend		
File	Display	Display, Amend	
Process	Display	Display	Display, Amend

2.6.4 How to Create, Review or Change a Field Rule/Trigger

The steps involved in reviewing or changing field rules/triggers at the DICTIONARY LEVEL are:

- Choose the "Review, change or create Field Rules and Triggers" option on the field control menu.
- Select the field(s) for which the rules or triggers are to be reviewed, changed or created using one of the methods previously described.
- The resulting display is then repeated for each chosen field. It shows all **DICTIONARY LEVEL** rules and/or triggers currently in existence for the field. Working from this display you can: Add new rules or triggers to the field.
- Review existing rules or triggers.
- Change existing rules or triggers.
- Delete existing rules or triggers.

2.6.5 Display All Existing Rules and Triggers

When working with field rules and triggers the first screen always displays all the rules and triggers that currently exist for the chosen field.

Remember that only the DICTIONARY LEVEL rules and triggers are displayed when working at the dictionary level. Other rules and triggers may exist at the FILE or PROCESS level, but these are not displayed.

DC@P700101 Select Rule / Trigger

- Usage Sel Seq Description Type Source A/C/D **** DICTIONARY RULES **** _ 10 Check not blank. LIST OF VALUES DATA DICTIONARY Y Y Y
- 20 Check valid number range RANGE OF VALUES DATA DICTIONARY *** DICTIONARY TRIGGERS ***
- _ 10 Print field value TRIGGER DATA DICTIONARY

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note that file name is displayed only when working at file level.

Working from the Select Rule / Trigger screen you can:

- Add a new rule or trigger by using the ADD function key. The resulting popup is described in the next section.
- Select an existing rule or trigger for review by entering any non-blank character beside the desired item in the column labeled "Sel". On the subsequent displays the rule or trigger can be reviewed in full, changed or deleted. These are described in detail in following sections.

2.6.6 Add a New Rule or Trigger

A pop-up will appear when the ADD function key is used from the previously described display of all existing rules and triggers.

Select Rule / Trigger DC@P700101 File : XXX (XXXX) Field :XXX : Add Rule / Trigger : (XXXX) Usage : Use cursor to make a selection. : Sel Seq Description : Range of Values Rule : ce A/C/D**** DICTION : List of Values Rule 10 Check not bl : File Entry Lookup Rule : TIONARY Y Y Y 20 Check valida : Evaluate Expression for Rule : TIONARY Y Y Y : Call User Program for Rule : : Date Range/Format Rule : : Add a Trigger : Cancel Add Request : Fnn=Help Fnn=Messages : •

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Reposition the screen cursor, within the pop-up, on the same line as the type of rule (or a trigger) you wish to add, and then press the enter key.

Once a type of rule (or a trigger) has been selected the next display will ask for a detailed specification of the rule or trigger. These displays are described in detail in these sections.

2.7 Range of Values Rule

This display results when:

- An existing range of values rule has been chosen for review.
- A new range of values rule is to be added.

DC@P700103 Range of Values Rule
File: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Order to process / source 999 / DATA DICTIONARY User description of rule
Use rule when performing ADD CHG DLT Check if field is in range to to
or in range : to
or in range : to
or in range :to
or in range :toto
or in range :to
If field is in range of values above : NEXT
Else field is NOT is range of values above : ERROR
Specify error message number, file and library :
Or text :
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note: The file name is only displayed when working at file level. Change and delete command keys are only enabled when displaying an existing check. If an existing rule is being reviewed then the CHANGE key can be used to place the screen in change mode and the required changes made.

If an existing rule is being reviewed then the DELETE key can be used to delete

the rule. Note that the delete is immediate. No confirmation is required.

If a new rule is being added then the screen will be presented initially with default values. Change as required and press Enter to complete specification of the rule.

Input Options

Order to Process

Mandatory, but always pre-filled to <highest order number + 10>. Rules are performed (and displayed) in the sequence of these order numbers (within the validation level). Order numbers must be unique within the validation level.

For instance these entries would be performed in the sequence RULE04, RULE01, RULE02, RULE05, RULE03:

Rule	Level	Order To Process
RULE01	Dictionary	100
RULE02	Dictionary	200
RULE03	File	20
RULE04	Dictionary	50
RULE05	File	10

Source

Output field. Indicates the source or level at which the rule applies.

User Description of Rule

Mandatory. Enter a brief description of the rule to aid other users in understanding its purpose.

Use Rule When Performing

At least one entry required. Pre-filled to ADD and CHG. Specifies "when" the rule is to be performed. Allowable values are:

- ADD When information is added (inserted) to the database.
- ADDUSE When information is added, and the field is actually specified/used in the INSERT command being executed.

- CHG When information is changed (updated) in the database.
- CHGUSE When information is changed, and the field is actually specified/used in the UPDATE command being executed.
- DLT When information is deleted (removed) from the database.

Most commonly used entries are ADD, CHG and CHGUSE. Use of DLT by itself is a common and a very powerful rule mechanism. If ADDUSE is specified, ensure that the default value of the field is a valid database value. Use caution when specifying CHGUSE with a rule that involves multiple fields, because the check will only be done when the field linked to the rule is specified on an UPDATE command, and not done when it is omitted, regardless of whether or not any of the other fields referenced in the rule are specified.

Check If Field Is in Range

At least one range is required. A maximum of 20 ranges can be specified (use roll keys to input). A "from" and "to" value is required for each range specified. Range values specified can be:

- An alphanumeric literal (in quotes) such as 'STATUS', 'JONES'
- A numeric literal such as 1, 14.23, -1.141217.
- Another field name such as CUSTNO, INVNUM, etc.
- A system variable name such as *BLANKS, *ZERO, *DATE or any other system variable defined at your installation.
- A process parameter such as *UP01, *UP02, etc.

Range values are checked for type and length compatibility. System does NOT check "from" value is less than "to" value.

If Field Is in Range of Values Above

Mandatory. Pre-filled to NEXT. Specifies what is to happen if the field is found to be in one of the ranges specified. Allowable values are:

- NEXT Field is "okay". Proceed to next rule for this field.
- ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Else Field Is NOT in Range of Values Above

Mandatory. Pre-filled to ERROR. Specifies what is to happen if the field is NOT found to be in one of the ranges specified. Allowable values are:

NEXT Field is "okay". Proceed to next rule for this field.

ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Error Message Number, File and Library

Optional. Specify either error message number, file and library or error message text, (refer text) but not both. Error message files and error message numbers are a native part of the IBM i operating system. Refer to the IBM supplied *Control Language Reference Manual* for details. CL commands involving message files include CRTMSGF and ADDMSGD.

You can directly edit the message details from this screen. Enter as much of the message details as is known and use the function key labeled "Work Msgd" (Work Message Description). This will cause a WRKMSGD command to be executed, using as much of the supplied message details as is possible. This facility will allow you to create or edit the message details. Upon completion of the WRKMSGD command, this screen will be redisplayed, unchanged, to allow you to proceed.

Do **not** store user defined messages in, or modify "shipped" messages in, the LANSA message file DC@M01 via this or any other message file editing facility. It is regularly replaced by new version or PC (program change) installations.

Text

Optional. If you do not wish to use an error message file (refer to Error Message Number, File and Library) to store the message text, then you may enter the text of the error message to be displayed directly. If this facility is used then the message will have no second level text associated with it. Refer to Messages and the Help Key.

If neither an error message number, file and library nor error message text is specified LANSA will insert a default error message number, file and library as the error message. These default messages are "general purpose" and do not provide much detail about the specific cause of the error.

2.7.1 Examples

These examples illustrate the use of the range of values rule facility:

Example 1

Field being checked:

Name	Туре	Len	Dec
POSTCD	А	4	

Range of values:

	From Value	To Value	Comments
	'2000'	'2900'	Check post code is in NSW, VIC or QLD. Post
Or	'3000'	'3900'	code is alpha.
Or	'4000'	'4900'	

Example 2

Field being checked:

Name	Туре	Len	Dec
POSTCD	S	4	0

Range of values:

	From Value	To Value	Comments
Or	2000 3000	2900 3900	Check post code is in NSW, VIC or QLD. Post code is numeric.

Example 3

Field being checked:

Name	Туре	Len	Dec
AMOUNT	Р	5	2

Range of values:

From Value	To Value	Comments
0.01	999.99	Check is AMOUNT positive.

Example 4

Field being checked:

Name	Туре	Len	Dec
AMOUNT	Р	5	2

Range of values:

From Value	To Value	Comments
-999.99	-0.01	Check is AMOUNT negative.

Example 5

Field being checked:

Name	Туре	Len	Dec
PERIOD	А	2	

Range of values:

From Value	To Value	Comments
*LASTPER	*NEXTPER	Check PERIOD using site defined system variables *LASTPER and *NEXTPER.

2.8 List of Values Rule

This display results when:

- An existing list of values rule has been chosen for review.
- A new list of values rule is to be added.

DC@P700104 List of Values Rule
File: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Order to process / source <u>999</u> / DATA DICTIONARY User description of rule
Use rule when performing <u>ADD</u> <u>CHG</u> <u>DLT</u> Check if field is in list :
+
If field is in the list specified above : <u>NEXT</u> Else field is NOT in the list specified above : <u>ERROR</u>
Or text :
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note: File name is only displayed when working at file level. Change & Delete function keys are only enabled when an existing check is displayed.

If an existing rule is being reviewed then the CHANGE key can be used to place the screen in change mode and the required changes made.

If an existing rule is being reviewed then the DELETE key can be used to delete

the rule. Note that the delete is immediate. No confirmation is required.

If a new rule is being added then the screen will be presented initially with default values. Change as required and press Enter to complete specification of the rule.

Input Options

These input options apply to a list of values rule:

Order to Process:

Mandatory, but always pre-filled to <highest order number + 10>. Validation rules are performed (and displayed) in the sequence of these order numbers (within the validation level). Order numbers must be unique within the validation level.

For instance the entries below would be performed in the sequence RULE04, RULE01, RULE02, RULE05, RULE03.

Rule	Level	Order To Process
RULE01	Dictionary	100
RULE02	Dictionary	200
RULE03	File	20
RULE04	Dictionary	50
RULE05	File	10

Source

Output field. Indicates the source or level at which the rule applies.

User Description of Rule

Mandatory. Enter a brief description of the rule to aid other users in understanding its purpose.

Use Rule When Performing

At least one entry required. Pre-filled to ADD and CHG. Specifies "when" the rule is to be performed. Allowable values are:

ADD When information is added (inserted) to the database.

- ADDUSE When information is added, and the field is actually specified/used in the INSERT command being executed.
- CHG When information is changed (updated) in the database.
- CHGUSE When information is changed, and the field is actually specified/used in the UPDATE command being executed.
- DLT When information is deleted (removed) from the database.

Most commonly used entries are ADD, CHG and CHGUSE. Use of DLT by itself is a common and a very powerful check mechanism. If ADDUSE is specified, ensure that the default value of the field is a valid database value. Use caution when specifying CHGUSE with a rule that involves multiple fields, because the check will only be done when the field linked to the rule is specified on an UPDATE command, and not done when it is omitted, regardless of whether or not any of the other fields referenced in the rule are specified.

Check If Field Is in List

At least one entry in the list is required. A maximum of 50 entries can be specified (use roll keys to input). Values specified can be:

- An alphanumeric literal (in quotes) such as 'STATUS', 'JONES'
- A numeric literal such as 1, 14.23, -1.141217.
- Another field name such as CUSTNO, INVNUM, etc.
- A system variable name such as *BLANKS, *ZERO, *DATE or any other system variable defined at your installation.
- A process parameter such as *UP01, *UP02, etc.

Values are checked for type and length compatibility.

If Field Is in List Specified Above

Mandatory. Pre-filled to NEXT. Specifies what is to happen if the field is found to be in the list specified. Allowable values are:

- NEXT Field is "okay". Proceed to next rule for this field.
- ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Else Field Is Not in the List Specified Above

Mandatory. Pre-filled to ERROR. Specifies what is to happen if the field is

NOT found to be in the list specified. Allowable values are:

NEXT Field is "okay". Proceed to next rule for this field.

ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Error Message Number, File And Library

Optional. Specify either error message number, file and library or error message text (described below), but not both. Error message files and error message numbers are a native part of the IBM i operating system. Refer to the IBM supplied *Control Language Reference Manual* for details. CL commands involving message files include CRTMSGF and ADDMSGD.

You can directly edit the message details from this screen. Enter as much of the message details as is known and use the function key labeled "Work Msgd" (Work Message Description). This will cause a WRKMSGD command to be executed, using as much of the supplied message details as is possible. This facility will allow you to create or edit the message details. Upon completion of the WRKMSGD command, this screen will be redisplayed, unchanged, to allow you to proceed.

Do **not** store user defined messages in, or modify "shipped" messages in, the LANSA message file DC@M01 via this or any other message file editing facility. It is regularly replaced by new version or PC (program change) installations.

Text

Optional. If you do not wish to use an error message file (described above) to store the message text, then you may enter the text of the error message to be displayed directly. If this facility is used then the message will have no second level text associated with it. Refer to the section on Messages and the Help Key.

If neither an error message number, file and library nor error message text is specified LANSA will insert a default error message number, file and library as the error message. These default messages are "general purpose" and do not provide much detail about the specific cause of the error.

2.8.1 Examples

These examples are provided to illustrate the use of the list of values rule facility:

Example 1

Field being checked:

Name	Туре	Len	Dec
STATE	А	3	

List of values:

List Of Values	Comments
'NSW'	Check for valid Australian state mnemonic.
'QLD'	
'VIC'	

Example 2

Field being checked:

Name	Туре	Len	Dec
NAME	А	7	

List of values:

List Of Values	Comments
*BLANKS	A blank name is an error. Reverse the default error logic to get ERROR if in list, NEXT if not in list.

Example 3

Field being checked:

Name	Туре	Len	Dec
COMPNO	Р	1	0

List of values:

List Of Values	Comments
1	Company number must be 1, 2 or 3.
2	
3	

Example 4

Field being checked:

Name	Туре	Len	Dec
YEAR	Α	2	

List of values:

List Of Values	Comments
*LASTYEAR *THISYEAR	Year must equal one of the site defined system variables *LASTYEAR, *THISYEAR or *NEXTYEAR
*NEXTYEAR	

2.9 Code / Table File Lookup Rule

This display results when:

- An existing code/table file lookup rule has been chosen for review.
- A new code/table file lookup rule is to be added.

DC@P700105 Code /Table File Lookup				
File: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX				
Order to process / source <u>999</u> / DATA DICTIONARY User description of rule				
Use rule when performing <u>ADD</u> <u>CHG</u> <u>DLT</u>				
Lookup code table / file :				
using key field(s) : $XXXXXXXXX$				
+				
If a "key match" is found in code file : <u>NEXT</u>				
Else a "key match" is NOT found in code file : <u>ERROR</u>				
Or text :				

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note: File name only displayed when working at file level. Change & Delete function keys only enabled when displaying an existing check.

If an existing rule is being reviewed then the CHANGE key can be used to place the screen in change mode and the required changes made. If an existing rule is being reviewed then the DELETE key can be used to delete the rule. Note that the delete is immediate. No confirmation is required.

If a new rule is being added then the screen will be presented initially with default values. Change as required and press Enter to complete specification of the rule.

Input Options

These input options apply to a code/table file lookup rule:

Order to Process:

Mandatory, but always prefilled to <highest order number + 10>. Validation rules are performed (and displayed) in the sequence of these order numbers (within the validation level). Order numbers must be unique within the validation level.

For instance, in the following example, they would be performed in this sequence: RULE04, RULE01, RULE02, RULE05, RULE03.

Rule	Level	Order To Process
RULE01	Dictionary	100
RULE02	Dictionary	200
RULE03	File	20
RULE04	Dictionary	50
RULE05	File	10

Source

Output field. Indicates the source or level at which the rule applies.

User Description of Rule

Mandatory. Enter a brief description of the rule to aid other users in understanding its purpose.

Use Rule When Performing

At least one entry required. Prefilled to ADD and CHG. Specifies "when" the rule is to be performed. Allowable values are:

ADD	When information is added (inserted) to the database.
ADDUSE	When information is added, and the field is actually specified/used in the INSERT command being executed.
CHG	When information is changed (updated) in the database.
CHGUSE	When information is changed, and the field is actually specified/used in the UPDATE command being executed.
DLT	When information is deleted (removed) from the database.

Most commonly used entries are ADD, CHG and CHGUSE. Use of DLT by itself is a common and a very powerful check mechanism. If ADDUSE is specified, ensure that the default value of the field is a valid database value. Use caution when specifying CHGUSE with a rule that involves multiple fields, because the check will only be done when the field linked to the rule is specified on an UPDATE command, and not done when it is omitted, regardless of whether or not any of the other fields referenced in the rule are specified.

Lookup Code/Table File

Mandatory. Specifies the physical or logical file that is to be used for checking. The full or partial name of the file can be entered. If a partial name is supplied a list of all files that have a generically identical name is displayed. The required file can be chosen from this list.

Using Key Field(s)

At least 1 key field is required. Specifies the key that is to be compared with the keyed index of the file looking for a "match" (i.e. to find a record with an identical key in the file). Initially, the field is pre-filled to name of the field that is being checked. It can be changed to any other value.

Key values specified can be:

- An alphanumeric literal (in quotes) such as 'STATUS', 'JONES'
- A numeric literal such as 1, 14.23, -1.141217.
- Another field name such as CUSTNO, INVNUM, etc.
- A system variable name such as *BLANKS, *ZERO, *DATE or any other system variable defined at your installation.
- A process parameter such as *UP01, *UP02, etc.

Key values are checked for type and length compatibility. The entire key list supplied is checked for compatibility with the actual key(s) of the file. The key
list specified can be a full or partial key to the file. A warning is issued if a partial key list is specified.

If a "Key Match" Is Found in Code File

Mandatory. Prefilled to NEXT. Specifies what is to happen if one or more records in the file have a key identical to the key(s) supplied. Allowable values are:

NEXT Field is "okay". Proceed to next rule for this field.

ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Else a "Key" Match Is NOT Found in Code File

Mandatory. Prefilled to ERROR. Specifies what is to happen if NO records in the file have a key identical to the key(s) supplied. Allowable values are:

NEXT Field is "okay". Proceed to next rule for this field.

ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Error Message Number, File and Library

Optional. Specify either error message number, file and library or error message text (described below), but not both. Error message files and error message numbers are a native part of the IBM i operating system. Refer to the IBM supplied *Control Language Reference Manual* for details. CL commands involving message files include CRTMSGF and ADDMSGD.

You can directly edit the message details from this screen panel. Enter as much of the message details as is known and use the function key labeled "Work Msgd" (Work Message Description). This will cause a WRKMSGD command to be executed, using as much of the supplied message details as is possible. This facility will allow you to create or edit the message details. Upon completion of the WRKMSGD command, this screen will be redisplayed, unchanged, to allow you to proceed.

Do **not** store user defined messages in, or modify "shipped" messages in, the LANSA message file DC@M01 via this or any other message file editing facility. It is regularly replaced by new version or PC (program change) installations.

Text

Optional. If you do not wish to use an error message file (described above) to store the message text, then you may enter the text of the error message to be displayed directly. If this facility is used then the message will have no second level text associated with it. Refer to the section on Messages and the Help Key.

If neither an error message number, file and library nor error message text is specified LANSA will insert a default error message number, file and library as the error message. These default messages are "general purpose" and do not provide much detail about the specific cause of the error.

2.9.1 Examples

These examples are provided to illustrate the use of the code/table file lookup rule facility:

Example 1

Field being checked:

Name	Туре	Len	Dec
PRODNO	А	10	

File details:

Name	Actual Keys	Keys Supplied	Comments
PRODMST	Product number	PRODNO	Check product is in product master

Example 2

Field being checked:

Name	Туре	Len	Dec
TAXCDE	А	3	

File details:

Name	Actual Keys	Keys Supplied	Comments
TAXTABL	Tax code	INCOME	Check TAXCDE is a valid income tax code using alpha literal 'INCOME' in

type		the key list.
Tax code	TAXCDE	

Example 3

Field being checked:

Name	Туре	Len	Dec
PARTNO	Р	7	0

File details:

Name	Actual Keys	Keys Supplied	Comments
INVENT	Part number Warehouse number	PARTNO	Check PARTNO exists. This is a "partial" key validation check.

Example 4

Field being checked:

Name	Туре	Len	Dec
INVNUM	А	8	

File details:

Name	Actual	Keys	Comments

	Keys	Supplied	
INVNUM	Invoice number	INVNUM	Check INVNUM does not already exist. Reverse default error logic to get ERROR if key match is found, NEXT if key match is NOT found.

2.10 Simple Logic Rule

This display results when:

- An existing simple logic rule has been chosen for review.
- A new simple logic rule is to be added.

DC@P700106 Simple Logic Rule
File: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Order to process / source <u>999</u> / DATA DICTIONARY User description of rule
Use rule when performing <u>ADD</u> <u>CHG</u> <u>DLT</u> Check field by evaluating the truth of the condition :
Ifthe condition above is found to be true: NEXTElsethe condition above is found to be false: ERRORSpecify error message number, file and library :
Or text :

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note that file name is only displayed when working at file level. Change and Delete function keys are only enabled when displaying an existing check. If an existing rule is being reviewed then the CHANGE key can be used to place the screen in change mode and the required changes made.

If an existing rule is being reviewed then the DELETE key can be used to delete the rule. Note that the delete is immediate. No confirmation is required.

If a new rule is being added then the screen will be presented initially with default values. Change as required and press Enter to complete specification of the rule.

Input Options

These input options apply to a simple logic rule:

Order to Process:

Mandatory, but always prefilled to <highest order number + 10>. Validation rules are performed (and displayed) in the sequence of these order numbers (within the validation level). Order numbers must be unique within the validation level.

For instance:

Rule	Level	Order To Process
RULE01	Dictionary	100
RULE02	Dictionary	200
RULE03	File	20
RULE04	Dictionary	50
RULE05	File	10

would be performed: RULE04, RULE01, RULE02, RULE05, RULE03.

Source

Output field. Indicates the source or level at which the rule applies.

User Description of Rule

Mandatory. Enter a brief description of the rule to aid other users in understanding its purpose.

Use Rule When Performing

At least one entry required. Prefilled to ADD and CHG. Specifies "when" the rule is to be performed. Allowable values are:

ADD	When information is added (inserted) to the database.
ADDUSE	When information is added, and the field is actually specified/used in the INSERT command being executed.
CHG	When information is changed (updated) in the database.
CHGUSE	When information is changed, and the field is actually specified/used in the UPDATE command being executed.
DLT	When information is deleted (removed) from the database.

Most commonly used entries are ADD, CHG and CHGUSE. Use of DLT by itself is a common and a very powerful check mechanism. If ADDUSE is specified, ensure that the default value of the field is a valid database value. Use caution when specifying CHGUSE with a rule that involves multiple fields, because the check will only be done when the field linked to the rule is specified on an UPDATE command, and not done when it is omitted, regardless of whether or not any of the other fields referenced in the rule are specified.

Check Field by Evaluating the Truth of the Condition

Mandatory. An expression that can be evaluated must be specified. The expression provided is evaluated and produces a "true" or "false" result. This is used in conjunction with other details provided to either proceed to the next rule, issue an error message or accept the field unconditionally.

Components of the expression can be:

- An alphanumeric literal such as 'NSW', NSW, 'Balmain' or BALMAIN.
- A numeric literal such as 1, 14.23, -1.141217.
- Another field name such as #CUSTNO, #INVNUM, etc.
- A system variable name such as *BLANKS, *ZERO, *DATE or any other system variable defined at your installation.
- A process parameter such as *UP01, *UP02, etc.

Note that alphanumeric literals do NOT have to be in quotes when used in an expression. Quotes are only required when the alphanumeric literal contains lowercase characters. If no quotes are used the alpha literal is converted to uppercase. Thus BALMAIN = balmain = Balmain = balMAIN, however, Balmain does not equal 'Balmain'.

Note also that field names must be preceded by a # (hash) symbol when used in expressions. This allows LANSA to differentiate between fields and alphanumeric literals. For instance the expression CNTRY = AUST does not indicate which of the components is the field and which is the alphanumeric literal. The correct format is #CNTRY = AUST or #CNTRY = 'AUST'.

Operator	Description	
(Open bracket	
)	Close bracket	
+	Add	
-	Subtract	
/	Divide	
*	Multiply	
=	Compare equal	
^=	Compare not equal	
<	Compare less than	
<=	Compare less than or equal to	
>	Compare greater than	
>=	Compare greater than or equal to	
*EQ	Compare equal	
*NE	Compare not equal	
*LT	Compare less than	
*LE	Compare less than or equal to	
*GT	Compare greater than	
*GE	Compare greater than or equal to	
AND	And	
OR	Or	

Valid expression operators are:

*AND	And
*OR	Or

Expression evaluation is left to right within brackets, so use brackets whenever in doubt as to the order in which the expression will be evaluated.

Expression components are checked for type and length compatibility. The syntax of the expression is checked. It must also be remembered that when evaluating numeric variables the leading variable will be used to determine the decimal precision. Error messages are issued that indicate any problems found when attempting to evaluate the expression.

If the Condition Above Is Found to Be True

Mandatory. Prefilled to NEXT. Specifies what is to happen if the expression is found to be "true" after evaluation. Allowable values are:

NEXT Field is "okay". Proceed to next rule for this field.

ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Else the Condition Above Is Found to Be False

Mandatory. Prefilled to ERROR. Specifies what is to happen if the expression is found to be "false" after evaluation. Allowable values are:

NEXT Field is "okay". Proceed to next rule for this field.

ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Error Message Number, File and Library:

Optional. Specify either error message number, file and library or error message text (described below), but not both. Error message files and error message numbers are a native part of the IBM i operating system. Refer to the IBM supplied *Control Language Reference Manual* for details. CL commands involving message files include CRTMSGF and ADDMSGD.

You can directly edit the message details from this screen. Enter as much of the message details as is known and use the function key labeled "Work Msgd"

(Work Message Description). This will cause a WRKMSGD command to be executed, using as much of the supplied message details as is possible. This facility will allow you to create or edit the message details. Upon completion of the WRKMSGD command, this screen will be redisplayed, unchanged, to allow you to proceed.

Do **not** store user defined messages in, or modify "shipped" messages in, the LANSA message file DC@M01 via this or any other message file editing facility. It is regularly replaced by new version or PC (program change) installations.

Text

Optional. If you do not wish to use an error message file (described above) to store the message text, then you may enter the text of the error message to be displayed directly. If this facility is used then the message will have no second level text associated with it. Refer to the section on Messages and the Help Key.

If neither an error message number, file and library nor error message text is specified LANSA will insert a default error message number, file and library as the error message. These default messages are "general purpose" and do not provide much detail about the specific cause of the error.

2.10.1 Examples

These examples illustrate the use of the simple logic rule facility:

Check Required

Actual Expression Used

Field VALUE must be greater than zero

Field STATE must be NSW, VIC or QLD.

Field WEIGHT must be zero if field MEASUR is not zero.

Field WEIGHT must be less than field MEASUR multiplied by 10.462

#VALUE > 0

- or #VALUE *GT 0
- (#STATE = NSW) OR or (#STATE = VIC) (#STATE = QLD)
- (#MEASUR ^= 0) AND or (#WEIGHT = 0)
 - (#MEASUR *NE 0) *AND (#WEIGHT *EQ 0)
- #WEIGHT < (#MEASUR * or 10.462)
 - #WEIGHT *LT (#MEASUR * 10.462)

2.11 Complex Logic Rule

This display results when:

- An existing complex logic rule has been chosen for review.
- A new complex logic rule is to be added.

DC@P700107 Complex Logic Rule
File: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Order to process / source <u>999</u> / DATA DICTIONARY User description of rule Use rule when performing ADD CHG DLT Check by calling program Type FUN/3GL with standard parameters : < return code >
<pre>< name of field being checked ></pre>
+
Ifthe program gives a "good" return code: NEXTElsethe program gives a "bad" return code: ERRORSpecify error message number, file and library :

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note: File name is only displayed when working at file level. Change & Delete function keys are only enabled when an existing check is displayed.

If an existing rule is being reviewed then the CHANGE key can be used to place the screen in change mode and the required changes made.

If an existing rule is being reviewed then the DELETE key can be used to delete

the rule. Note that the delete is immediate. No confirmation is required.

If a new rule is being added then the screen will be presented initially with default values. Change as required and press Enter to complete specification of the rule.

Input Options

These input options apply to a complex logic rule:

Order to Process:

Mandatory, but always prefilled to <highest order number + 10>. Validation rules are performed (and displayed) in the sequence of these order numbers (within the validation level). Order numbers must be unique within the validation level.

Rule	Level	Order To Process
RULE01	Dictionary	100
RULE02	Dictionary	200
RULE03	File	20
RULE04	Dictionary	50
RULE05	File	10

For instance:

would be performed: RULE04, RULE01, RULE02, RULE05, RULE03.

Source

Output field. Indicates the source or level at which the rule applies.

User Description of Rule

Mandatory. Enter a brief description of the rule to aid other users in understanding its purpose.

Use Rule When Performing

At least one entry required. Prefilled to ADD and CHG. Specifies "when" the rule is to be performed. Allowable values are:

ADD When information is added (inserted) to the database.

- ADDUSE When information is added, and the field is actually specified/used in the INSERT command being executed.
- CHG When information is changed (updated) in the database.
- CHGUSE When information is changed, and the field is actually specified/used in the UPDATE command being executed.
- DLT When information is deleted (removed) from the database.

Most commonly used entries are ADD, CHG and CHGUSE. Use of DLT by itself is a common and a very powerful check mechanism. If ADDUSE is specified, ensure that the default value of the field is a valid database value. Use caution when specifying CHGUSE with a rule that involves multiple fields, because the check will only be done when the field linked to the rule is specified on an UPDATE command, and not done when it is omitted, regardless of whether or not any of the other fields referenced in the rule are specified.

Check by Calling Program

Mandatory. Specify the name of the LANSA function or 3GL program to be called. Note that no check is done for the existence of the function/program.

Note: The function/program should be able to be found in the user's library list at the time the rule is to be performed.

Туре

Specifies whether the field is to be checked by calling a LANSA function or a 3GL program. Note that additional parameters cannot be specified if a validation function is being specified. See the **FUNCTION** command in the *Technical Reference Guide* for design constraints on validation functions.

With Standard Parameters

Output fields. For information purposes only. All 3GL programs called as part of a complex logic rule must have a least 3 standard parameters. These are:

Name	Description
Return code	Alphanumeric length 1. Returned by the program as '1' (good return) or '0' (bad return). Used by the program to indicate to LANSA the success or failure of the complex logic rule.
Name of	Alphanumeric length 10. Passed to the program. Contains the name (as opposed to the value) of the field that is passed in the

field	third parameter.
Value of field	Length and type depend upon the data dictionary definition of the field. Alphanumeric fields are passed with same type and length as their data dictionary definition. All numeric fields (type P or S) are passed as packed (type P) and the same length and number of decimal positions as their data dictionary definition. Note that the value of the field is passed in a work area, thus it is not possible to change the value of the field by changing the value of the parameter in the validation program.

Note that LANSA does not check that the 3GL program being called has the correct parameter list. This is a user responsibility.

And Additional Parameters

Optional. Additional parameters may be passed to the program. Additional parameters may be:

- An alphanumeric literal (in quotes) such as 'STATUS', 'JONES'.
- A numeric literal such as 1, 14.23, 1.141217.
- Another field name such as CUSTNO, INVNUM, etc.
- A system variable name such as *BLANKS, *ZERO, *DATE or any other system variable defined at your installation.
- A process parameter such as *UP01, *UP02, etc.

The type and length of the parameter(s) passed depends upon the type and length of the parameter value supplied.

For alphanumeric fields (alpha literals, alpha fields, alpha system variables or alpha process parameters) the parameter is passed as alpha (256) with the parameter value left aligned into the 256 byte parameter.

For numeric fields (numeric literals, numeric fields, numeric system variables or numeric process parameters) the parameter is passed as packed 15 with the same number of decimal positions as the parameter value. For numeric literals this means the same number of decimal positions as specified in the literal (e.g.: 1.12 will be passed as packed 15,2. 7.12345 will be passed as packed 15,5. 143 will be passed as packed 15,0. etc). For all other types of numeric parameters this means the same number of decimal positions as their respective definitions.

As with the standard parameters, the actual value is passed in a work area so it is not possible to change the value of a field by changing the parameter value in the validation program.

Note 1: Additional parameters may only be used when a 3GL program is called, and cannot be used when a validation function is called.

Note 2: LANSA does not check that the 3GL program being called has the correct parameter list. This is a user responsibility.

If the Program Gives a Good Return Code

Mandatory. Prefilled to NEXT. Specifies what is to happen if the return code parameter is returned as '1' (a "good" return). Allowable values are:

NEXT Field is "okay". Proceed to next rule for this field.

ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Else the Program Gives a Bad Return Code

Mandatory. Prefilled to ERROR. Specifies what is to happen if the return code parameter is returned as **something other than '1'** (a "bad" return). Allowable values are:

- NEXT Field is "okay". Proceed to next rule for this field.
- ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Error Message Number, File and Library

Optional. Specify either error message number, file and library or error message text (see following), but not both. Error message files and error message numbers are a native part of the IBM i operating system. Refer to the IBM supplied *Control Language Reference Manual* for details. CL commands involving message files include CRTMSGF and ADDMSGD.

You can directly edit the message details from this screen. Enter as much of the message details as is known and use the function key labeled "Work Msgd" (Work Message Description). This will cause a WRKMSGD command to be executed, using as much of the supplied message details as is possible. This facility will allow you to create or edit the message details. Upon completion of the WRKMSGD command, this screen will be redisplayed, unchanged, to allow

you to proceed.

Do **not** store user defined messages in, or modify "shipped" messages in, the LANSA message file DC@M01 via this or any other message file editing facility. It is regularly replaced by new version or PC (program change) installations.

Text

Optional. If you do not wish to use an error message file to store the message text, then you may enter the text of the error message to be displayed directly. If this facility is used then the message will have no second level text associated with it. Refer to the section on Messages and the Help Key.

If neither an error message number, file and library nor error message text is specified LANSA will insert a default error message number, file and library as the error message. These default messages are "general purpose" and do not provide much detail about the specific cause of the error.

2.11.1 Examples

These examples are provided to illustrate the use of the complex logic rule facility:

Check Required	Program Specified	Additional Parameters
Pass field DATDUE to program DATECHECK for validation.	DATECHECK	none
Pass field DATDUE to program INVCHECK for validation. Also pass fields INVNUM and COMPNO.	INVCHECK	INVNUM COMPNO
Pass field TAXCDE to program TAXCHECK for validation. Also supply tax scale 7 as a parameter with 3 decimal positions.	TAXCHECK	7.000 or +7.000 or 07.000

2.12 Date Format / Range Rule

This display results when:

- An existing date format/range rule has been chosen for review.
- A new date format/range rule is to be added.

DC@P700108 Date Format / Range Rule			
File: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX			
Order to process / source <u>999</u> / DATA DICTIONARY User description of rule Use rule when performing <u>ADD</u> <u>CHG</u> <u>DLT</u> Validate date in format <u>DDMMYY</u>			
Number of days allowed into the past <u>9999999</u>			
Number of days allowed into the future <u>9999999</u>			
If field passes date format/range check : NEXT Else field fails data format/range check : ERROR Specify error message number, file and library :			

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note that file name is only displayed when working at file level. Change and Delete function keys are only enabled when an existing check is displayed.

If an existing rule is being reviewed then the CHANGE key can be used to place the screen in change mode and the required changes made.

If an existing rule is being reviewed then the DELETE key can be used to delete

the rule. Note that the delete is immediate. No confirmation is required.

If a new rule is being added then the screen will be presented initially with default values. Change as required and press Enter to complete specification of the rule.

Input Options

These input options apply to a date format/range rule:

Order to Process

Mandatory, but always prefilled to <highest order number + 10>. Rules are performed (and displayed) in the sequence of these order numbers (within the validation level). Order numbers must be unique within the validation level. For instance:

Rule	Level	Order To Process
RULE01	Dictionary	100
RULE02	Dictionary	200
RULE03	File	20
RULE04	Dictionary	50
RULE05	File	10

would be performed: RULE04, RULE01, RULE02, RULE05, RULE03.

Source

Output field. Indicates the source or level at which the rule applies.

User Description of Rule

Mandatory. Enter a brief description of the rule to aid other users in understanding its purpose.

Use Rule When Performing

At least one entry required. Prefilled to ADD and CHG. Specifies "when" the rule is to be performed. Allowable values are:

ADD When information is added (inserted) to the database.

- ADDUSE When information is added, and the field is actually specified/used in the INSERT command being executed.CHG When information is changed (updated) in the database.CHCUSE When information is changed and the field is actually.
- CHGUSE When information is changed, and the field is actually specified/used in the UPDATE command being executed.
- DLT When information is deleted (removed) from the database.

Most commonly used entries are ADD, CHG and CHGUSE. Use of DLT by itself is a common and a very powerful check mechanism. If ADDUSE is specified, ensure that the default value of the field is a valid database value.

Use caution when specifying CHGUSE with a rule that involves multiple fields, because the check will only be done when the field linked to the rule is specified on an UPDATE command, and not done when it is omitted, regardless of whether or not any of the other fields referenced in the rule are specified.

Validate Date in Format

Mandatory. Initially set to SYSFMT. It specifies the format that the date field should be in.

Valid format types in this version of LANSA are:

SYSFMT	Operating system date format (from QDATFMT)
SYSFMT8	8 digit date in operating system format.
DDMMYY	Day month year format
MMDDYY	Month day year format
YYMMDD	Year month day format
DDMMYYYY	Day month century year format
MMDDYYYY	Month day century year format
YYYYMMDD	Century year month day format
YYYYDDMM	Century year day month format
YYMM	Year month format
YYYYMM	Year month format with 4 digit year
MMYY	Month year format

MMYYYY	Month year fo	rmat with 4 digit year
	wionui ycai io	illiat with + digit year

For example, to satisfy each format type, 28th October 1986 must be entered as:

SYSFMT	281086 (Usual format for Australia and Europe)
SYSFMT	102886 (Usual format for USA)
SYSFMT8	28101986 (Usual format for Australia and Europe)
SYSFMT8	10281986 (Usual format for USA)
DDMMYY	281086

- MMDDYY 102886
- YYMMDD 861028
- DDMMYYYY 28101986
- MMDDYYYY 10281986
- YYYYMMDD 19861028
- YYYYDDMM 19862810
- YYMM 8610
- YYYYMM 198610
- MMYY 1086

MMYYYY 101986

Note: The client's date format will be automatically passed to the server. If the client and server date formats are different (e.g. MDY vs DMY), the server will automatically return data in the client's format.

The client's format can be changed from the default by specifying the x_run parameter DATF=. For more information, please refer to Standard X_Run Parameters in the *Technical Reference Guide*.

If client and server date formats are different (such as between USA and UK clients), date format validation rules specifying exact formats will fail. For example, DDMMYY may be returned as MMDDYY. Where clients need to use different date formats, date format SYSFMT is recommended.

Number of Days Allowed into the Past

Mandatory. Prefilled to 9999999. Specifies the lower limit of the date range rule.

Number of Days Allowed into the Future

Mandatory. Prefilled to 9999999. Specifies the higher limit of the date range rule.

The use of the "days into the past" and "days into the future" range limit values can be illustrated with a time line:



If Field Passes Date Format/Range Rule

Mandatory. Prefilled to NEXT. Specifies what is to happen if the field is found to be in the required date format and passes the range test. Allowable values are:

NEXT Field is "okay". Proceed to next rule for this field.

ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Else Field Fails Date Format/Range Rule

Mandatory. Prefilled to ERROR. Specifies what is to happen if the field is not in the required date format or fails the range test. Allowable values are:

NEXT Field is "okay". Proceed to next rule for this field.

ERROR Field is in error. Issue error message described below.

ACCEPT Field is okay. Bypass all other rules for this field.

Error Message Number, File and Library

Optional. Specify either error message number, file and library or error message text (described below), but not both. Error message files and error message numbers are a native part of the IBM i operating system. Refer to the IBM supplied *Control Language Reference Manual* for details. CL commands involving message files include CRTMSGF and ADDMSGD.

You can directly edit the message details from this screen. Enter as much of the message details as is known and use the function key labeled "Work Msgd" (Work Message Description). This will cause a WRKMSGD command to be executed, using as much of the supplied message details as is possible. This facility will allow you to create or edit the message details. Upon completion of the WRKMSGD command, this screen will be redisplayed, unchanged, to allow you to proceed.

Do **not** store user defined messages in, or modify "shipped" messages in, the LANSA message file DC@M01 via this or any other message file editing facility. It is regularly replaced by new version or PC (program change) installations.

Text

Optional. If you do not wish to use an error message file (described above) to store the message text, then you may enter the text of the error message to be displayed directly. If this facility is used then the message will have no second level text associated with it. Refer to the section on Messages and the Help Key.

If neither an error message number, file and library nor error message text is specified LANSA will insert a default error message number, file and library as the error message. These default messages are "general purpose" and do not provide much detail about the specific cause of the error.

Note: All dates must have a four character year so that accurate comparisons and calculations can be performed. Where a two character year (eg. DDMMYY, YYMMDD, MMYY) is supplied the century value is retrieved from the system definition data area. The year supplied is compared to a year in the data area, if the supplied year is less than or equal to the comparison year then the less than year is used. If the supplied year is greater than the comparison year then the greater than year is used.

2.12.1 Examples

These examples are provided to illustrate the use of the date format/range rule:

Description Of Check Required	Date Format	Days Into Past	Days Into Future
Check field DATDUE is in format DDMMYY	DDMMYY	99999999	99999999
Check field DATDUE is in format YYMMDD and is not prior to the current date.	YYMMDD	0	99999999
Check field DATE is in format DDMMYYYY and is within the next 90 days.	DDMMYYYY	0	90
Check field DATE is in format DDMMYY and is in the last 180 days.	DDMMYY	180	0
Check field DATE is in format YYYYMMDD and is in the last week.	YYYYMMDD	7	0
Check field DATE is in format YYYDDMM and is within the next week.	YYYYDDMM	0	7
Check field DATE is in format YYMM.	YYMM	9999999	99999999
Check field DATE is in format MMDDYYYY and is within the next 30 days.	MMDDYYYY	0	30

2.13 Triggers

Γ

This display results when:

- An existing trigger has been chosen for review.
- A new trigger is to be added

DC@P700103	Define / Change a Trigger
<u>File : XXXXXXXX</u> Field : XXXXXXXX	<u>XXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX</u>
Order to process / sour	ce <u>999</u> / DATA DICTIONARY
User description of trig	ger
Trigger Function Name	2 · · · · · · · ·
	Before After
Trigger Point(s)	Open
	Close
	Read
	Insert
	Undate
	Delete
And/Or H	Field Op Field/Literal
Trigger When	
	+

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note that file name is only displayed when working at file level. Change and Delete command keys are only enabled when displaying an existing check. If an existing trigger is being reviewed then the CHANGE key can be used to place the screen in change mode and the required changes made.

If an existing trigger is being reviewed then the DELETE key can be used to delete the trigger. Note that the delete is immediate. No confirmation is required.

If a new trigger is being added then the screen will be presented initially with default values. Change as required and press Enter to complete specification of the trigger.

Input Options

These input options apply to a trigger:

Order to Process

Mandatory, but always prefilled to <highest order number + 10>. Triggers are performed (and displayed) in the sequence of these order numbers (within the trigger level). Order numbers must be unique within the trigger level.

For instance:

Trigger	Level	Order To Process	
TRIG01	Dictionary	100	
TRIG02	Dictionary	200	
TRIG03	File	20	
TRIG04	Dictionary	50	
TRIG05	File	10	

would be performed: TRIG04, TRIG01, TRIG02, TRIG05, TRIG03.

Source

Output field. Indicates the source or level at which the trigger applies.

User Description of Trigger

Mandatory. Enter a brief description of the trigger to aid other users in understanding its purpose.

Trigger Function Name

Mandatory. Enter the name of the trigger function which will be called when the trigger conditions are satisfied. The named function must be defined with the TRIGGER parameter on the FUNCTION command. (Refer to the FUNCTION command in the *Technical Reference Guide* for details).

Trigger Points

At least one point where the function is to be triggered must be entered (by entering a "Y" at the required point).

Trigger When

Optional. If no test conditions are specified the trigger will be unconditionally invoked.

Rules for entering conditions are as follows:

- Operations allowed are:
 - GT Greater than
 - GE Greater than or equal to
 - LT Less than
 - LE Less than or equal to
 - EQ Equal to
 - NE Not equal to
 - REF Refers to
- All operation codes can be suffixed by a "P" indicating that the previous value of the nominated field should be used. Such conditions are only valid for "Update" operations. Also Part 2 of the condition must not be a literal if this type of operation code is used.
- No conditions are allowed if "Open" or "Close" are used.
- The "REF" operation code is only valid for "Insert", "Update" and "Read" operations.
- If the trigger is defined at dictionary level, the conditions can only involve the field to which the trigger is being linked.
- If the trigger is defined at the file level, the conditions can involve any other real or virtual fields in the file to which the trigger is being linked.

Examples

Trigger field:

Name	Туре	Len	Dec
SALARY	Р	12	2

Trigger function:

Name	Function	Trigger Before	Points After	Comments
TRGFN1	Open			Call trigger function TRGFN1
	Close			following an insert or update of field SALARY, or before a
	Read			delete of field SALARY. Note
	Insert		Y	that no conditions have been applied to this trigger.
	Update		Y	11 00
	Delete	Y		

Refer to Trigger Functions for full details and examples

2.14 Field Help Text

HELP text is information that is displayed to the user when the HELP key is used.

Generally Field Help text has these characteristics:

- It is free format. No restrictions usually exist on the content or format of HELP text.
- It relates directly to the action the user was taking at the time the HELP key was pressed. Usually the process or function that the user is using is explained in some detail.
- Quite often the fields on the screen that were on display when the HELP key was pressed are each described in detail. Part of the description usually involves the name and description of the field, any validation rules that apply to its input and comments on the use and interpretation of the field's value.

LANSA automatically controls the handling of the HELP key. In addition LANSA will automatically determine the type of HELP that is required (field, process or function) and automatically display the associated HELP text (if any exists).

LANSA can dynamically, and in the correct language, create the HELP text associated with a field from the dictionary and the rules that it contains. You can turn off this automatic field level help text feature globally, by field, or precede it with your own HELP text.

What LANSA cannot do is create the free format HELP text that is associated with the processes or functions. This is generally the task of the person who creates the process or function.

These sections deal with how HELP text is entered for fields. For details of how HELP text is entered for processes or functions refer to the Process and Function sections.

Once the HELP text has been entered LANSA will automatically display it at the appropriate time.

The actual method used to display HELP text is described in some detail in the Messages and the Help Key.

How to use LANSA's Help text features is described in the following topics:

2.14.1 Steps to Create, Review or Change HELP Text

2.14.2 HELP Text Copy Options

- 2.14.3 Use the IBM Source Entry Utility (SEU)
- 2.14.4 Use Special Characters to Enhance HELP Text
- 2.14.5 Use Substitution/Control Values in HELP Text
- 2.14.6 Substitution/Control Values for Windows-specific HELP Text
- 2.14.7 Use IBM i Office Help Documents
- 2.14.8 Exit from the SEU Utility
- 2.14.9 Use Panel Groups for Help Text
- 2.14.10 Create Panel Groups for HELP Text in LANSA
- 2.14.11 Help Text Considerations when using Panel Groups

2.14.1 Steps to Create, Review or Change HELP Text

The steps involved in the review, change or creation of a fields HELP text are as follows:





2.14.2 HELP Text Copy Options

If no HELP text currently exists for the chosen field, then this format is presented:

DC@P100201 Create Help Text by Copy

No HELP text currently exists for this FIELD Optionally use any one of the options below to initialise the HELP text

Either copy HELP text from field . . .Orcopy HELP text from process . .Orcopy HELP text from process . .Orset up a standard HELP text form by entering YES

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Using this format you can "initialize" the chosen field's HELP text by copying the HELP text from another field, process or function (within a process).

In addition it is possible to initialize the chosen field's HELP text to a "standard form".

The "standard form" that comes with the shipped version of LANSA can be changed. Refer to Help Text Enhancement & Substitution Values in the *Technical Reference Guide* for details.

If you use a "standard form" for the help, you can alter it by selecting from two procedures:

1. Sign on as the LANSA security officer and invoke LANSA. Review or change the HELP text associated with Process STDFORM. 2. Review or change the "standard form" associated with function STDFORM. Function STDFORM is part of process STDFORM.
2.14.3 Use the IBM Source Entry Utility (SEU)

If HELP text exists for the field, or after completion of input on the HELP text copy options screen, a screen like this will be displayed:

Columns . . . : 1 71 Edit QTEMP/HELPWORK Find ... HELPWORK FMT ** ...+...1 ...+...2 ...+...3 ...+...4 ...+...5 ...+...6. 0001.00 \$\$LANGUAGE=ENG 0002.00 0003.00 This field represents the {employee number} assigned to the 0004.00 employee by this company. Each and every employee must have an employee number. On any screen where you are asked to 0005.00 0006.00 specify an employee number you can press the prompt key on 0007.00 employee number to cause a pop-up window to be displayed. 00.8000 You can then search by employee name and select the 0009.00 required employee number.

F3=Exit	F4=Prompt	F5=Refresh
F10=Top	F11=Bottom	F24=More keys

This screen is presented by the SEU (edit source) utility. SEU is provided by IBM and is used to create and manipulate source or text.

SEU is very widely used by IBM i programmers and other users. If you are unfamiliar with the features and functions of SEU refer to the IBM supplied reference manual.

In addition, SEU has online HELP facilities. Once you have started to use SEU press the HELP key and review the extensive HELP text available.

By using SEU the HELP text associated with the field can be reviewed, created or changed. All the features of SEU are available and can be used.

If SEU is not available on your IBM i machine, the help text will be edited by EDTF. Refer to help on this command for features. On exit from the EDTF you will be asked if you wish any changes made to the LANSA help text to be

copied to the LANSA files.

2.14.4 Use Special Characters to Enhance HELP Text

Some special features are provided by LANSA to enhance the display of HELP text. These consist of special characters that can be used as follows:

Special Character Effect

%(percentage)	Display in high intensity
{(left parenthesis)	Underline
@(at)	Display in reverse image
~(accent)	Blink
<pre>}(right parenthesis)</pre>	Revert to normal display mode
\(backslash)	Revert to normal display mode

Some examples of using these special characters are:

HELP Text Input Using SEU Displayed By LANSA As

the%item number\must	the \$#item number#\$ must
the{item number}must	the \$@item number@\$ must
%{VALIDATION RULES\	\$%VALIDATION RULES%\$
%{VALIDATION RULES}	\$%VALIDATION RULES%\$
{%VALIDATION RULES}	\$%VALIDATION RULES%\$

Where multiple enhancement characters are used together with no spaces between them they are "consolidated" into one enhancement character. For instance "%{~" (high intensity, underline, blink) would be consolidated into one enhancement character that is used in the same position as the "~". The "%" and "{" are replaced by blanks. This explains why in the last example above the underline ("{") attribute starts under the "V" in "VALIDATION" and not one character before it.

Note that a display enhancement cannot be continued over multiple lines. Display enhancements are considered to end at the end of the text line, even if a "revert to normal display" character ("\" or "}") is not used:

contact the{Accounts department or

your data entry supervisor for more information. would be displayed as:

contact the \$@Accounts department or_____@\$ your data entry supervisor for more information.

2.14.5 Use Substitution/Control Values in HELP Text

LANSA provides a set of "substitution" and "control" values that be included into HELP text to control and enhance its display to the user.

HELP text substitution/control values are substituted/actioned at execution time (i.e. when the HELP text is being actively displayed to a user).

Note: These values must be entered in uppercase in the HELP text.

Substitution Value	Description, Effects And Comments
\$\$PAGE	Causes the string "Page: 999" to be inserted into the text. Use to indicate the current HELP text page number to the user.
\$\$NEWPAGE	Causes a new HELP text page to be started. A HELP text page is 18 lines long by default. The line containing the \$\$NEWPAGE value is included into the displayed HELP text. The \$\$NEWPAGE value is replaced by blanks.
\$\$TITLE XXXXXXXX	Indicates to LANSA the title that should be associated with the HELP text. Up to 40 characters of title can be specified. LANSA will automatically center the title and convert it to uppercase. The line containing the \$\$TITLE value is NOT included into the displayed help text.
\$\$FLDNAM	Causes the name of the current field to be inserted into the HELP text. Use this value in field level HELP text only.
\$\$FLDDES	Causes the description of the current field to be inserted into the HELP text. Use this value in field level HELP text only.
\$\$COMNAM	Causes the name of the current component to be inserted into the HELP text. Use this value in component level HELP text only.
\$\$COMDES	Causes the description of the current component to be inserted into the HELP text. Use this value in

	component level HELP text only.
\$\$PRONAM	Causes the name of the current process to be inserted into the HELP text. Use this value in process or function level HELP text only.
\$\$PRODES	Causes the description of the current process to be inserted into the HELP text. Use this value in process or function level HELP text only.
\$\$FUNNAM	Causes the name of the current function to be inserted into the HELP text. Use this value in function level HELP text only.
\$\$FUNDES	Causes the description of the current function to be inserted into the HELP text. Use this value in function level HELP text only.
\$\$RIGHT	Causes all manually defined help text to be right aligned when displayed. This support is provided for bi-directional languages.
\$\$NOAUTO	Indicates that automatically generated help text should not be created for this field. Use this option in field level help text only.
\$\$LANGUAGE=XXXX	Delimits the boundary between the help text associated with different languages when working in a multilingual partition. These values are automatically created when editing help text and should not be altered in any way or the help text associated with a language may be corrupted.
\$\$TECH	Indicates that the following text is technical help text. This help will only be displayed to a developer within LANSA.
\$\$USER	Indicates that the following text is user help text. This help will always be displayed within LANSA. Help text will default to user help text if no \$\$TECH value is used.

2.14.6 Substitution/Control Values for Windows-specific HELP Text

These help text substitutions are used to link help for other objects to the automatically generated Index (& Contents) for the current object. They do not produce visible entries in the generated help text.

Note that the Contents are only produced for the new help interface provided with Windows.

Substitution Value	Description, Effects and Comments
\$\$INDEXFLD = <field-name></field-name>	Causes an item to be included in the Index that will display help text for the designated field. The rest of the current line is ignored. The label used in the Index is the field's description.
\$\$INDEXCOM = <object- name>, <component- name></component- </object- 	Causes an item to be included in the Index that will display help text for the designated component. The component is specified using the owning object (e.g. form) and component names, separated by a comma. The rest of the current line is ignored. The label used in the Index is the component's name. If no component name is specified then the index item will display help for the object (e.g. form) itself.
\$\$INDEXPRO = <process- name></process- 	Causes an item to be included in the Index that will display help text for the designated process. The rest of the current line is ignored. The label used in the Index is the process's description.
\$\$INDEXFUN = <process- name>, <function- name></function- </process- 	Causes an item to be included in the Index that will display help text for the designated function. The function is specified using the owning process and function names, separated by a comma. The rest of the current line is ignored. The label used in the index is the function's description.
\$\$ROOT	Causes any following Index substitutions (\$\$INDEX) to be added to the root of the Contents tree-view. Useful for index items that might be applicable to all or a

	number of topics.
\$\$LEAF	Causes any following Index substitutions (\$\$INDEX) to be added below the current topic in the Contents tree- view. This is the default value and only needs to be specified to switch back after \$\$ROOT has been used.
\$\$IMAGE = <filename>, <alternate-text></alternate-text></filename>	A bitmap (*.BMP) image file to be included in the help text at this point. The image is centered in the screen. The default path is the current partition's Execute directory, but any path may be specified. Note: Universal Naming Convention (UNC) paths are not supported. The alternate text is used instead of the image in an interface where images are not displayed, such as a 5250 terminal. If the alternate text is not specified then the filename is used instead.

The characters used to control help text attributes (i.e.Blink, Underline etc.) can be modified to suit your requirements. Currently the attribute values are @, %, \sim , {, }, and \. If these conflict with special characters at your site the attributes can be re-assigned using these keywords.

Note: These keywords must be entered in uppercase.

Keyword	Example	Description
\$\$HI=	\$\$HI=!	High Intensity
		Causes the special character assigned for HIGH INTENSITY display to be re assigned to a user defined character.
\$\$RI=	\$\$RI=[Reverse Image
		Causes the special character assigned for REVERSE IMAGE display to be re assigned to a user defined character.
\$\$BL=	\$\$BL=:	Blink
		Causes the special character assigned for BLINK display to be re assigned to a user defined

		character.
\$\$UL=	\$\$UL=+	Underline
		Causes the special character assigned for UNDER LINE display to be re assigned to a user defined character
\$\$N1=	\$\$N1=*	Normal Display
		Causes the special character assigned for NORMAL display one to be re assigned to a user defined character.
\$\$N2=	\$\$N2=#	Normal Display
		Causes the special character assigned for NORMAL display two to be re assigned to a user defined character

Note: The "HANDS-ON" facility is no longer applicable and will only be invoked if present. New LANSA users should ignore the following text.

2.14.7 Use IBM i Office Help Documents

When these substitution values are used within HELP text, all other substitution values and HELP text will be ignored.

HELP will be presented (via an interface with the IBM i Office System) in the form of *Help Documents*. For information on creating these documents, see IBM's Create a process documentation guide.

Note 1: Office is not supported with OS/400 V5R1 and higher.

Note 2: This facility, is not available when using Visual LANSA.

Substitution Value	Description, Effects And Comments
\$\$HLPDOC=XXXXXXXX	Causes a help document to be displayed from the IBM i Office System. The help document name of up to 12 characters, must be entered directly adjacent to \$\$HLPDOC=. The help document specified must be a resolved document. \$\$HLPDOC= must be used in relation with \$\$HLPFLR=.
\$\$HLPFLR='XXXXXXX'	Causes a help document to be displayed from the IBM i Office System. The help folder name, of up to 63 characters, must be entered directly adjacent to \$\$HLPFLR= and must be enclosed in single quotes. \$\$HLPFLR= must be used in relation with \$\$HLPDOC=.
\$\$HLPLBL=XXXXXXXX	Causes a help document to be displayed from the IBM i Office System, starting at a specific help label position within the help document. The help label, of up tp 10 characters, must be entered directly adjacent to \$\$HLPLBL=. \$\$HLPLBL= must be used in relation with \$\$HLPDOC= and \$\$HLPFLR=. If \$\$HLPDOC= and \$\$HLPFLR= are both specified and \$\$HLPLBL= has not, a help document will be displayed from first help label or the start of the help document.

2.14.8 Exit from the SEU Utility

Once finished with the field's HELP text use function key 3 to exit from SEU. A screen like this will be presented:

Exit

Type choices, press Enter.

Change/create member <u>N</u> Y=Yes, N=No
Member <u>HELPWORK</u> Name
File <u>HELPWORK</u> Name
Library <u>QTEMP</u> Name
Text
Resequence member $\dots $ $Y = Yes, N=No$
Start
Increment
Print member N Y=Yes, N=No
_ ,
Return to editing N Y=Yes, N=No
0 _ ,
Go to member list N Y=Yes, N=No
_ ,
F3=Exit F5=Refresh F12=Cancel

The SEU exit option - change/create member, will be pre-filled to N if you did not change the field HELP text in any way. Otherwise it will be pre-filled to Y. Normally you would use the option "Y" to change/create member and exit from SEU to return to LANSA. Notice also the option to return to editing. Other options can be used if desired, but you may find that any changes made are lost and the field's HELP text remains unchanged.

Note that the member, file and library always contain HELPWORK,

HELPWORK and QTEMP respectively. This applies no matter what field's HELP text is being edited. This is a special temporary source file that LANSA has created in library QTEMP. These values can be changed if required, but again you may find that any changes are lost and the field's HELP text remains unchanged.

If SEU is not available on your IBM i machine, the help text will be edited by EDTF. When finished editing the help text use function key 2 or 3 to save any changes made. Use function key 3 to exit from EDTF.

You will be asked: Do you wish to save the changes made to HELPWORK? Y/N.

Reply Y to save changes to the LANSA Help files.

2.14.9 Use Panel Groups for Help Text

Instead of the LANSA help text display facility, IBM's UIM Panel Groups may be used to present the user defined help text at execution time. With Panel Groups, the window size and positioning on the screen are cursor sensitive. Panel Groups may also increase the performance of the help text presentation.

Panel Groups are created from source written in the UIM tag language (for further details on this language, refer to the IBM supplied *Guide to Programming Application and Help Displays*). LANSA creates this source from the help text entered through the LANSA facilities. When using UIM help panels with LANSA, some special considerations apply and these are listed in 2.14.11 Help Text Considerations when using Panel Groups.

To enable the use of Panel Groups in LANSA, the text string '*UIMHELP' should be entered in data area DC@OSVEROP. Further details on this data area and its use are in the Version Dependency Data Area DC@OSVEROP. Setting the string '*UIMHELP' in DC@OSVEROP causes LANSA to:

- Prompt for the creation of Panel Groups after exit from SEU during a help text edit session.
- Compile processes and functions in a way that they use Panel Groups for help text when executed later. This includes Field, Function and Process level help text.
- Cause export and import jobs to include help text Panel Groups, if they exist, for the objects included in the export list.

The text string '*UIMHELP' must be set at **compile time** if Panel Groups are to be used for help text presentation. A function compiled while the text string IS NOT set will not look for a Panel Group at execution time, even if the string is then set, and a Panel Group exists for the help text. Vice versa, a function compiled while the text string IS set will look for a Panel Group at execution time, even if the string has been removed. If the Panel Group does not exist, a message saying that help text is not available will be displayed by the UIM manager, even if help text would be available through the LANSA facility. To disable the use of Panel Groups by a function, the function must be re-compiled with the string '*UIMHELP' removed from data area DC@OSVEROP.

The text string '*UIMHELP' also must be set at the time that an export or import is run if Panel Groups are to be included. Note that the LANSA help text source itself is still included in the export or import. To revert to the display of the help text using LANSA instead of the Panel Groups, processes and functions can be recompiled with the string '*UIMHELP' removed from DC@OSVEROP.

2.14.10 Create Panel Groups for HELP Text in LANSA

If the text string '*UIMHELP' is set in data area DC@OSVEROP, the Submit Creation of Panel Group screen will be presented after exit from SEU during a help text edit session.

DC@P100203	Submit Creation of Panel Group	
Field CUSTNC	O _ Customer Number	(ENG)

You may now submit a job to create the Help Panel Group for this FIELD To submit the creation of the panel group, fill in the fields below.

Submit this job	NO	YES,NO
Using job name	DC@2	PNLGRC
Job Description	QGPL/0	QBATCH
Job Queue	*JOBD	
Output Queue	*JOBD	

Specify language or *ALL ENG

The name of the panel group (re_)created will be DC@PNLGRC. Use HELP or this message F1=Help F3=Exit F12=Cancel F14=Msgs

The details on this screen can now be changed to meet the requirements, and to submit the job: Submit This Job Job Name Job Description

Job Queue Output Queue

Specify Language or *ALL

This item is shown in multilingual systems only. It is pre-filled with the current language. Any language valid for the partition for which the Panel Group is to be (re-)created, may be entered here. If the special value '*ALL' is entered, a job will be submitted for each partition language, for the creation of the Panel Group.

A message will be displayed at the bottom of the screen, indicating the name of the Panel Group to be created. This name is generated by LANSA and can not be changed. The name of the Panel Group will be the process name for processes and functions, meaning that one Panel Group will exist for each process and all functions within it. For fields, the generated name will be DC@PNLGR_x, where x is the first character of the field name. Hence, one Panel Group will exist for all fields with a certain first character. Help text for fields with a name starting with a special character is included in DC@PNLGR_A.

This means that if help text is created or edited for more than one object, a compile of the Panel Group needs to be submitted only once for a given group of objects (fields with a common starting letter, functions within a process), that is, for the last object in the group that help text was edited for.

The Panel Groups for fields, functions and processes do not need to exist at the time a process or function is compiled. They can be created at any time. If the help key is used at execution time, but the Panel Group for the object does not exist, the UIM manager presents a message in a window, indicating that help text is not available.

2.14.11 Help Text Considerations when using Panel Groups

In general, apart from the considerations below, help text for Panel Groups can be entered in exactly the same way as help text using the LANSA display facility.

Window Size

LANSA determines the UIM help window width for an object (Field, Function or Process) from the longest line entered for the object, including the window title. The user can influence the window width by keeping the lines of text longer or shorter. The window length is determined by the cursor position at execution time and can not be influenced. For consistent help text presentation, site standards for window width should be considered.

Text Formatting

In the LANSA implementation of Panel Groups, text formatting by UIM is disabled, so the text is shown at execution time exactly in the way that it is entered through SEU using the LANSA supplied facilities, as long as the text entered does not exceed a total of 66 characters per line.

Wrapping of Text onto the Next Line

To avoid the wrapping of text onto the next line, the help text entered should have a maximum line length of no more than 66 characters. From position 67 onwards, all positions in the source file should be left blank when editing help text.

Window Title

LANSA automatically designs a window title for Fields, Functions and Processes. For fields, this title consists of the field description, plus the field name if enough space is available (i.e. if the description is no longer than 35 characters, the maximum allowable title length in UIM being 55 characters). For processes and functions, this title consists of the process or function description.

To replace the automatically designed title with a different title, use the \$\$TITLE substitution value described in the section on substitution/control values. The title will be truncated if it exceeds the maximum allowable in the UIM language of 55 characters.

Automatic Help

Automatic help produced by LANSA is recorded at the time the Panel Group is

compiled. The help text presented by a Panel Group will not reflect any subsequent changes in field definitions or validation rules, or any other help text changes, until the Panel Group is re-compiled.

2.15 Print Field Definitions

The steps involved in printing field definition(s) are as follows:



2.15.1 Submit the Print Job to Batch

When the option is used to print field definitions this screen is displayed to enable a batch job that will perform the actual printing to be submitted:

DC@P100501 Print Field Definitions
Submit this job YES YES, NO
Using Job name
Print all fields or only selected fields ALL ALL SEL Print HELP text NO YES, NO Print dictionary validation checks NO YES, NO Print files using fields cross reference NO YES, NO Print processes using fields cross reference NO YES, NO Print fields referring to fields cross reference NO YES, NO Print component details NO YES, NO
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Input Options

These input options apply to printing field definitions: Submit This Job Job Name Job Description Job Queue Output Queue

Print All Fields or Only Selected Fields

Specifies whether all fields in the data dictionary are to be printed, or only fields selected using the standard field selection method previously described.

Mandatory field. Prefilled to SEL. Allowable values are:

SEL Only previously selected fields should be printed.

ALL All fields in the dictionary should be printed.

Print HELP Text

Specifies whether the HELP text associated with the field(s) should be printed. Mandatory. Pre-filled to NO.

Print Dictionary Rules/Triggers

Specifies whether rules and triggers associated with the field(s) should be printed. Mandatory. Pre-filled to NO.

Print Files Using Fields Cross Reference

Specifies whether a cross reference of the files that use the field(s) should be printed. Mandatory. Pre-filled to NO.

Print Processes Using Fields Cross Reference

Specifies whether a cross reference of the processes that use the field(s) should be printed. Mandatory. Pre-filled to NO.

Print Fields Referring to Fields Cross Reference

Specifies whether a cross reference of the fields that refer to the field(s) being reported should be printed. Mandatory. Pre-filled to NO.

Print Component Details

Specifies whether the extended component details of the field(s) being reported should be printed. Mandatory. Pre-filled to NO. For more information about components, refer to the *Visual LANSA Primer*.

Printing of the HELP text, rules and triggers, cross reference and component details can substantially increase both the size of the report and the length of time taken to produce it.

2.16 System Variables

A system variable is a name that is assigned to a commonly used (and often variable) piece of information.

For instance, the current date is a commonly used piece of information that varies every day. It could be assigned the system variable name *TODAYS_DATE.

Once this is created, you can use this system variable name in many places within LANSA:

- As a field's default value in the data dictionary.
- In any of the 6 standard types of field validation rules.
- Just about anywhere in the RDML components of LANSA.

The extensive use of system variables in validation rules has many advantages:

- The validation rules are simpler and easier to understand.
- The validation rules are more powerful and more accurate.
- The validation rules last longer and are easier to maintain.

For instance, if a field named DATDUE is to be validated so that it can only be the current date. The required validation rule could be done in any of these ways:

- As a range of values rule. Validate DATDUE as being in range *TODAYS_DATE to *TODAYS_DATE.
- As a list of values rule. Validate DATEDUE as being in the list *TODAYS_DATE.
- As a simple logic rule. Validate DATDUE by evaluating the expression '#DATDUE = *TODAYS_DATE'.

Many system variables are provided with the shipped version of LANSA. However, the primary use of system variables is to aid in accessing your own company's information. You might create some system variables defined to match your company's specific requirements such as these:

System Variable	Description	Туре	Len	Dec
*PERIOD	Current financial period	Ν	4	0
*LASTPERIOD	Last financial period	Ν	4	0
*NEXTPERIOD	Next financial period	Ν	4	0

*BEGINMONTH	Date of first day of current month	Ν	6	0
*ENDMONTH	Date of last day of current month	Ν	6	0
*SYDOFFICE	Sydney office name and address code	А	7	
*MEBOFFICE	Melbourne office name and address code	А	7	
*COMPANY	Company number for use in G/Ledger	А	2	

Note that the system variables listed above are not supplied with LANSA. They are suggestions for creating your own system variables.

When a system variable is defined to LANSA, a name, description, method of derivation, type and length must be specified.

The method of derivation indicates when the system variable is to be derived or evaluated. It can be specified as STATIC or DYNAMIC.

In addition, the name of a LANSA function or 3GL program that will "evaluate" the system variable must also be specified. It is this function/ program that is important because it provides the "intelligence" required to decide what is the correct value to place in the system variable.

For information about the evaluation programs, refer to System Variable Evaluation Programs in the *Visual* LANSA Developer Guide.

2.16.1 Steps to Create a New System Variable Definition

The steps involved in the creation of new system variable definitions are as follows:



2.16.2 Create a New System Variable

When the option to create a new system variable is chosen from the field control menu this format is displayed until the EXIT or MENU/CANCEL function key is used:

DC@P100303	Add System Variable Definition
System wide variable	name
Description	••••
Method of derivation	<u>STATIC</u> STATIC, DYNAMIC
Data type	<u>ALPHA</u> ALPHA, NUMBER
Length / Total digits	· · · · · · ·
Decimal Positions	·····_
Set value by calling p	rogram Type FUN 3GL
Initial public access .	<u>NORMAL</u> (ALL, NORMAL, NONE)
Fnn=Help Fnn=Exit	Fnn=Cancel Fnn=Messages

Input Options

These input options apply to the creation or amendment of a system variable definition:

System Wide Variable Name

This is the symbolic name of the system variable. It must:

- Begin with a "*" (asterisk).
- At least 4 characters in length.
- Not contain imbedded blanks.
- Not be *ALL, *ALL_REAL, *ALL_VIRT, *DEFAULT, *EXCLUDING, *HIVAL, *INCLUDING, *LOVAL, *NAVAIL, or *NULL as these values are reserved by LANSA.
- Not begin with *MTXT because this prefix is reserved for multilingual variables.

Description

Mandatory. Enter a brief description of what the system variable is or represents to aid other users of the system.

Method of Derivation

Mandatory. Default value is STATIC. Specifies how LANSA is to derive the system variable within a file I/O module or user written function that references it. Allowable values are:

- STATIC The system variable is a static value, therefore its value can be derived once (during program initialisation) by LANSA. Examples of static system variables would include the current job name, the current user name and most probably the current date (providing that applications do not normally span midnight while executing).
- DYNAMIC The system variable is a dynamic value, therefore its value must be derived each and every time it is referenced. Examples of dynamic system variables would include the current time, the current output queue name and library and all user defined system variables that "allocate" values such as the next invoice number, the next batch number, etc.

Note that every time a dynamic system variable is referenced the associated evaluation program is called to "refresh" the system variable. Excessive use of dynamic system variables with complex evaluation programs may degrade LANSA performance.

Data Type

Mandatory. Allowable values are:

- ALPHA System variable is alphanumeric.
- NUMBER System variable is numeric. Use of this option in fact nominates the system variable as a packed decimal variable.

Length/Total Digits

Specifies the number of characters in an alpha system variable. Specifies the total number of digits (including decimals) in a numeric system variable.

- Must be in range 1 to 256 for type ALPHA.
- Must be in range 1 to 30 for type NUMBER.

Decimal Positions

Specifies the number of decimals in a numeric type system variable. Only valid for type NUMBER. Ignored for type ALPHA. Must be in range 0 to 9 and less than or equal to total digits.

Set Value by Calling Program

Specifies the name of the LANSA function or 3GL program that is to be called to set the value of the system variable.

The name of a LANSA function or 3GL program must be specified. Note that if a LANSA function is specified, LANSA checks that the function name does not exceed 7 characters- but does not check that the function actually exists. The function should be able to be located in the user's library list at the time the system variable is evaluated.

Note also that if a 3GL type program is specified, LANSA checks that the name specified is a valid program name, but does not check that the program actually exists.

Note also that the program should be able to be located in the user's library list at the time the system variable is evaluated. For more information refer to System Variable Evaluation Programs in the Visual LANSA Developer Guide.

Туре

Specifies whether the variable is to be evaluated by a LANSA function or a 3GL program.

Initial Public Access:

This is mandatory, but always pre-filled to NORMAL.

Allowable values are:

NORMAL Other users can use this system variable, but cannot change or

delete its definition in/from the data dictionary.

- ALL Other users can use this system variable and can also change or delete its definition in/from the data dictionary.
- NONE Other users cannot use this system variable, nor can they review, change or delete its definition in/from the data dictionary.

2.16.3 Review or Change System Variable Definitions

The steps involved in the review or change of system variable definitions are as follows:



2.16.4 Select the System Variable(s) Required

When you elect to review or change system variable definitions a screen format is presented that requests the name of the system variable(s) to be worked with.

If you have been working with system variables previously then the name will be pre-filled with the name of the last system variable you worked with.

SELSYSVAR Select System Var to Work with Enter full or partial name of the system variable(s) to be worked with or leave blank to select from all system variables . . . ______ Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Using this display you can nominate the system variables(s) required in 3 different ways:

- **In full.** Enter the full name of the system variable required. If the system variable is found then it will be immediately displayed in detail.
- **Partially.** Enter as much as desired of the partial name of the system variable. A search is made for all system variables which generically match the name specified and the resulting list is displayed for subsequent selection.
- Leave blank. In this case a list of all system variables is displayed from which those required can be selected.

When a list of system variables is displayed for selection it will look like the screen following (which resulted from entering partial system variable name *D):

SELSYSVAR Select System Var to Work with

Enter full or partial name of the system variable(s) to be worked with or leave blank to select from all system variables . . . *D

Sel	System variable	Description
_	*DATE	Numeric date in installation format
_	*DATEC	Character date in installation format
_	*DAY	Current day (numeric)
_	*DAYC	Current day (character)
_	*DDMMYY	Numeric date in format DDMMYY
_	*DDMMYYC	Character date in format DDMMYY

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Using this display you can:

- Alter the system variable name at the top right of the screen. The change can be to specify a full system variable name, a partial system variable name or a change to blanks. This results in a new search for the required system variable(s).
- **Select** one or more system variables from the list by entering any non-blank character beside the system variable in the column labeled "Sel". Each selected system variable will then be displayed in detail.

2.16.5 Detailed Display of a System Variable

This screen format is used to display a system variable in detail:

Using this display you can:

- Review the details of the system variable.
- Use the CHANGE function key to change details of the system variable. Note that the name cannot be changed.
- Use the DELETE function key to request delete of the system variable. After it has been used it is necessary to confirm the delete request by entering YES on this screen format:

System wide variable name : XXXXXXXXXXXXXXXXXXXXXXXXXX
Description: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Method of derivation : XXXXXXX STATIC, DYNAMIC
Data type: XXXXXX ALPHA, NUMBER
Length / Total digits: 999
Decimal Positions: 9
Set value by calling program : XXXXXXXX Type XXX FUN
Confirm this system variable is to be deleted
Delete this system variable : YES, NO
WARNING: Deletion of a system variable removes it from ALL partitions
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note:

- Changes to system variable definitions must match the input options described in the section "Creating a new system variable".
- Changes to the type and/or length of a system variable definition may require re-compilation of all file I/O modules and process definitions that reference it. Refer to System Variables in the *Technical Reference Guide* for details.
- Deletion of a system variable definition should be done with extreme care. Deleting a system variable may cause file I/O modules and/or process definitions that reference it to fail. The failure may be during execution and/or during compilation.

2.17 Print System Variable Definitions

The steps involved in printing system variable definition(s) are as follows:


2.17.1 Submit the Print Job to Batch

When the option is used to print system variable definitions the Print System Variable Definition screen is displayed to enable a batch job that will perform the actual printing to be submitted.

DC@P101001 Print System Variable Definition	
Submit this job YES YES, NO	
Using Job name	
Print all system variables or only selected ALL ALL SEL	
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages	

Input Options

These input options apply to printing system variable definitions: Submit This Job Job Name Job Description Job Queue Output Queue Print All System Variables or Only Selected

Specifies whether all system variables are to be printed, or only system

variables selected using the standard selection method previously described. Mandatory field. Pre-filled to SEL. Allowable values are:

- SEL Only previously selected variables should be printed.
- ALL All system variables should be printed.

2.18 Multilingual Variables

A multilingual variable is a text string that changes value according to the language being used.

The following example of a text string means the same thing in different languages, but has a different textual value in each language:

"Locate Employee by Name"

"Ricerca Alfabetica Impiegati"

"Zoek Werknemer op Naam"

If this was a piece of text that you wished to place on a screen, a report, or send in a message, then a multilingual variable (called an *MTXT variable as well) should be used.

By positioning an *MTXT defaulted field on the screen, rather than just text, the values that appear on the screen at execution time will vary according to the language being used.

In many respects *MTXT variables are just like system variables (described in 2.16 System Variables), and they can be used in the same places as system variables:

- As a field's default value in the data dictionary.
- In any of the 6 standard types of field validation rules.
- Just about anywhere in the RDML components of LANSA.

The extensive use of multilingual variables in validation rules has many advantages:

- The validation rules are simpler and easier to understand.
- The validation rules are more powerful and more accurate.
- The validation rules work in multiple languages.

For example, imagine these multilingual variables:

*MTXTMON	"Monday" in all desired languages
*MTXTTUE	"Tuesday" in all desired languages
*MTXTWED	"Wednesday" in all desired languages
*MTXTTHU	"Thursday" in all desired languages
*MTXTFRI	"Friday" in all desired languages

*MTXTSAT	"Saturday" in all desired languages
*MTXTSUN	"Sunday" in all desired languages
*MTXTERR001	"Day of week specified is invalid" in desired languages

Then, in just one list of values rule in the dictionary, a file or an RDML program, you could validate the day of the week (in the current language) and issue an error message (in the current language).

Additionally, as the "dictionary" of *MTXT variables grows, they will become used repeatedly in different applications. For instance *MTXTMON, *MTXTTUE, etc might be used as column headings on screens or reports.

Note: Before attempting to make large scale use of *MTXT variables please develop and use naming standards for them.

2.18.1 Steps to Create a New Multilingual Variable

The steps involved in the creation of new multilingual variable definitions are as follows:



2.18.2 Create a New Multilingual Variable

When the option to create a new multilingual variable is chosen from the field control menu, a format like this example is displayed until the EXIT or CANCEL function key is used:

DC@P100801	Multilingual Variable - GGGG
Multilingual Variable Maximum Length . Initial public access *123 XXXXXXXXX	e *MTXT NORMAL (ALL, NORMAL, NONE) 3 4 5 6 7
XXXXXXXXXX	•
XXXXXXXXXX	•
XXXXXXXXXX	•
Fnn=Help Fnn=Exit	Fnn=Cancel Fnn=Messages

Key in details of the new multilingual variable and press Enter.

Input Options

These input options apply to the creation (or amendment) of multilingual variables:

Special Note: This Facility uses Language "Grouping"

Standard LANSA language group processing means that any screen interaction may have to be repeated for each language group (if languages are used in the

group). The group being processed is shown on the top of the example screen as GGGG, but when in use will have values:

LRTB Left to right languages group.

RLTB Right to left languages group.

DBCS Double byte character set languages group.

Multilingual Variable

Specifies the name of the multilingual variable:

- The first 5 characters are fixed to "*MTXT" and cannot be changed.
- This allows ready identification of multilingual variables.
- The rest of the name must be from 1 to 15 characters, contain no imbedded blanks, conform to IBM i & CPF valid name rules, and be unique within the current partition.
- Use a maximum of 3 characters for function key names as the input field on RDML commands is only 8 long.
- Develop and use naming standards for *MTXT variables.
- In packaged systems, use obscure prefixes to preserve uniqueness.
- Use as default value for fields in the dictionary, rather than direct RDML reference, is preferable when intention is to use as panel or report text. Improves cross referencing capabilities.
- Fields in the dictionary should have a naming standard too. It should equate to the naming standard used for the multilingual variable that is used as its default value.
- Most *MTXT references are "locked in" at compile time. Change of an *MTXT variable will require application recompile. This is where the field/default value/cross reference capability is most useful.
- Exception is use of *MTXT variables as message text. In this situation the derivation is dynamic, so no recompiles are required.

Maximum Length

Specifies the maximum length of multilingual variable.

- Range 1 to 78.
- No value specified should exceed length specified.
- In RLTB languages, length is from the right hand side.

• Includes shift characters in DBCS languages.

Initial Public Access

Mandatory, but always pre-filled to NORMAL. Allowable values are:

NORMAL	Other users can use this multilingual variable, but cannot change
	or delete its definition in/from the data dictionary

- ALL Other users can use this multilingual variable and can also change or delete its definition in/from the data dictionary
- NONE Other users cannot use this multilingual variable, nor can they review, change or delete its definition in/from the data dictionary.

XXXXXXXXXXX

- Keyed from right in RLTB languages.
- Include shift characters in DBCS languages.
- Do not exceed maximum length specified.
- Use upper and lower case characters as required.
- Where applicable, manually centering within the maximum length can be used.
- Center from left (within maximum length) for LRTB and DBCS languages.
- Center from right (within maximum length) for RLTB languages.

2.18.3 Review or Change a Multilingual Variable

The steps involved in the review or change of a multilingual variable are as follows:



2.18.4 Select the Multilingual Variable(s) Required

When you elect to review or change a multilingual variable, a screen is presented that requests that you choose the multilingual variable(s) to be worked with.

If you have been working with multilingual variables previously then the name at the top of the display will be pre-filled with the name of the last multilingual variable you worked with.

DEMDICMENU Field Control Menu

Enter full or partial name of multilingual text variables to be worked with or leave blank to select from all variables . . . <u>*MTXTDEMCAL</u>

Sel Variable Name	Left Ali	gned Value in Current Language
_ *MTXTDEMCALE	N05801	MONDAY
_ *MTXTDEMCALE	N06001	TUESDAY
_ *MTXTDEMCALE	N06201	WEDNESDAY
_ *MTXTDEMCALE	N06401	THURSDAY
_ *MTXTDEMCALE	N06601	FRIDAY
_ *MTXTDEMCALE	N06801	SATURDAY
_ *MTXTDEMCALE	N07001	SUNDAY

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Using this display you can nominate the multilingual variables(s) required in 3 different ways:

- **In full.** Enter the full name of the multilingual variable required. If the multilingual variable is found then it will be immediately displayed in detail.
- **Partially.** Enter as much as desired of the partial name of the multilingual variable. A search is made for all multilingual variables which generically match the name specified and the resulting list is displayed for subsequent selection.

• Leave blank. In this case a list of all multilingual variables is displayed from which those required can be selected.

After one or more multilingual variables have been selected, a detailed display will be presented for each one.

The layout and content of this detailed display is identical to that already described in the section "Creating a new multilingual variable".

Using this detailed display you can:

- Review the detailed definition of the multilingual variable.
- Change the definition of the multilingual variable. Use the CHANGE function key to make the screen input capable. Make the desired changes and press Enter. Validation rules applicable are as for the "Creating a new multilingual variable" section.
- Delete the definition of the multilingual variable. Use the DELETE function key. Enter YES into the confirm delete prompt and press Enter.
- Create a new multilingual variable. Use the CREATE function key and a full creation panel, as described in 2.18.2 Create a New Multilingual Variable, will appear.

2.19 Print Multilingual Variable Definitions

The steps involved in printing multilingual variable definition(s) are as follows:



2.19.1 Submit the Print Job to Batch

When the option is used to print multilingual variable definitions the Print Multilingual Variable Definition screen is displayed to enable a batch job that will perform the actual printing to be submitted.

DC@P101201
Submit this job YES YES, NO
Using Job name
Print all multilingual variables or selectedALL ALL SEL
Select a language to print or *ALL ALL
Sel Language Description
_ ENG English
FRA French
JPN Japanese
_ LLL Test Card Language
Enn-Holn Enn-Ewit Enn-Concol Enn-Massages
rini-neip rini-Exit rini-Cancel rini-Messages

Input Options

These input options apply to printing multilingual variable definitions: Submit This Job Job Name Job Description

Job Queue Output Queue

Print All Multilingual Variables or Selected

Specifies whether all multilingual variables are to be printed, or only multilingual variables selected using the standard selection method previously described.

Mandatory field. Pre-filled to SEL. Allowable values are:

SEL Only previously selected multi variables should be printed.

ALL All multi variables should be printed.

Select a Language to print or *ALL

Specifies whether to select a language to print or all the languages.

Optional field. Select from the list below this option by entering any character or select *ALL to print all languages:

Sel

Enter any character in this column to select the language you want to print.

2.20 Field Reference File

A field reference file is a normal IBM i database file that is created for use as a data dictionary.

Typically a field reference file contains the definition of every field used by an installation. The field reference file never actually contains information (records) itself, it is used as a common reference point for the definition of fields used in other "real" database files that are created.

Since field reference files are very widely used, a facility is provided that will create a field reference file from the LANSA data dictionary.

It should be noted that a field reference file created by LANSA has no use whatsoever within LANSA. The facility is provided purely as an aid to applications that are external to LANSA and provides a means of centralizing and standardizing the methods by which data dictionary(s) are maintained on the IBM i.

2.20.1 Steps to Create (or Recreate) a Field Reference File

The steps involved in the creation or re-creation of a field reference file are as follows:



2.20.2 Create (or Re-Create) a Field Reference File

If the option to create or re-create a field reference file is chosen from the field control menu this screen is displayed:

DC@P218001 Create System Field Reference File
Submit this job <u>YES</u> YES, NO
Using Job name
Create field reference file named <u>FLDREF</u> In library named <u>QGPL</u>
Produce source listings <u>NO</u> YES, NO
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Input Options

These input options apply to creating a field reference file: Submit This Job Job Name Job Description Job Queue Output Queue Create Field Reference File Named Specifies the name of the field reference file that is to be created or re-created. Pre-filled to FLDREF. Can be changed as desired. LANSA checks that the name supplied is a valid file name.

In Library Named

Specifies the name of the library in which the field reference file is to be created or re-created. Pre-filled to QGPL. Can be changed as desired. Special value *LIBL (search library list) is NOT acceptable. LANSA checks that name supplied is a valid library name, but does not check that the library actually exists. Specification of a non-existent library will cause the batch job to fail. Error messages in the job log will indicate the cause of the failure.

Produce Source Listings

Indicates whether the DDS (data description specifications) used to create the field reference file should be printed. Allowable values are:

YES Produce the DDS source listings.

NO Do not produce DDS source listings.

LANSA will attempt to delete the named field reference file in the library specified before attempting to create a new field reference file. It may be necessary to authorize the LANSA owner profile to delete any existing field reference file(s).

Creation of a field reference file with a large number of fields (> 1000) may take an extended period of time. In such cases the batch job should be scheduled for overnight or off-peak execution.

3. Files

This chapter describes these topics:

- 3.1 What Is a File and What Is a File Definition?
- 3.2 The File Control Menu
- 3.3 Select File Definition(s) to Work With
- 3.4 Create a New File Definition
- 3.5 File Definition Maintenance
- 3.6 Delete a File Definition
- 3.7 Make a New or Amended File Definition Operational
- 3.8 Print File Definitions
- 3.9 Bulk Load of OTHER files
- 3.10 The File Definition Menu
- 3.11 Fields in the File Definition
- 3.12 Logical Views/Files Associated with File Definition
- 3.13 File Level Rules/Triggers
- 3.14 Access Routes to Other Files
- 3.15 Virtual Fields
- 3.16 Batch Control Logic
- 3.17 Database File Attributes
- 3.18 File Multilingual Attributes
- 3.19 Load the Definition of a File NOT Maintained by LANSA
- 3.20 Tips and Techniques for Defining Files Using LANSA

3.1 What Is a File and What Is a File Definition?

The file control component of LANSA works with file definitions.

It is important that the concept of a file definition and how it relates to the IBM i concept of a file be understood before attempting to use the file control components of LANSA.

A file is a normal IBM i database file in which records can be retrieved, added, changed or deleted. A file may be a physical file or a logical view / file.

To create a file LANSA uses a file definition.

In IBM i terms a file definition contains:

- The fields that make up the format of the file records.
- Logical views of the file that are required. The logical views may contain select/omit criteria.

In addition the LANSA file definition may also contain:

- Validation rules that are to be performed on fields when records are added, updated or deleted.
- Access routes from the current file to other files in the system. This facility allows users to access a complete database without having a detailed knowledge of its structure.
- Predetermined join fields which use the current file's access routes to retrieve information from related files.
- Virtual field definitions. These are fields that are not actually present in the database file but are "derived" from fields that are in it. Hence the name "virtual". These fields appear as if they are in the file, even though they aren't.
- Batch control logic which specifies other files in the system that contain batch, control or summary totals of information in the current file. LANSA will ensure that the summary or control information in other files is automatically maintained.

Some points to note about working with file definitions are:

- The file definition name and actual physical file name are always the same. When LANSA requests that the name of a file definition be supplied, specify the name of the physical file (as opposed to the name of any associated logical views/ files).
- The preceding point applies particularly when adding or amending logical

views / files. When the option to "Review or change a file definition" is chosen from the file control menu, LANSA will request that the name of the file definition to be review / change be specified. Specify the name of the physical file (which is always the same as the file definition) rather than the name of the logical view / file. Once the correct file definition has been chosen options on the file definition menu allow the addition, review or change of the associated logical files.

• To emphasize these two preceding items again. Within the LANSA system logical views / files are considered to be part of the file definition. The file definition contains the attributes of the associated physical file and all associated logical views / files. The file definition name and the associated physical file name are one and the same.

3.1.1 When New File Definition Is Made Operational

When the menu option to make a new file definition operational is taken this occurs:

- 1. An IBM i physical file is created. The format of the records in this file is determined by the fields that are contained in the LANSA file definition.
- 2. If specified in the LANSA file definition one or more IBM i logical files may be created. All logical files will be based on the physical file created in step 1. Refer to 3.1 What Is a File and What Is a File Definition? for details.
- 3. An I/O module will be created. This is an RPG program written and compiled by LANSA. It is called an I/O module because it supports all input and output operations to the physical file and any logical files created in the preceding steps. In addition it contains any validation and batch control logic in the LANSA file definition.

The actual procedures for making a new file definition operational are described in more detail in the following sections.

3.1.2 When Amended File Definition Is Made Operational

When the LANSA file definition has been amended and the menu option to make an amended file definition operational is taken this occurs:

- LANSA compares the amended definition of the file with the associated IBM i physical and logical files that currently exist. During the comparison a list is prepared that details all the actions required to make the physical and logical files identical to the amended file definition. This procedure is analogous to the "set operational" procedures used by many IT installations.
- Working from the list of actions built in the preceding step LANSA then recreates the physical and logical files to match the amended file definition. All of the work is done in temporary areas so if the recreation fails the database will be left in its prior condition and users can continue to use it.
- The type of action taken during a re-creation vary widely with the type of amendment made to the LANSA file definition. This is best illustrated by examples:

Actions Taken During Re-Creation

Type Of

Amendment	
New logical view	New logical view created in temp area
added	New logical view moved to user library.
New field added to file	 New physical file created in temp area New logical views created in temp area New I/O module created in temp area New physical file in temp area loaded from existing physical file. Existing physical file renamed * Existing logical views deleted Existing I/O module deleted New physical file moved to user library New logical views moved to user library New I/O module moved to user library
Validation rules	New I/O module created in temp area
changed	Existing I/O module deleted

Note the action indicated with a "*". The existing physical file is renamed by prefixing it with "\$\$". Thus existing file CUSMST will be renamed \$\$CUSMST before the new version of CUSMST is actually moved into the user library.

This is an important step because it means that the data in the file can never be accidentally lost. However, 2 points should be noted:

- When a file re-creation has completed there are 2 versions of the file in the user library. The new version that has been loaded and the old version (with the "\$\$" prefix). This means that until the "\$\$" version is deleted twice as much disk space is being used.
- The "\$\$" version can be manually deleted (refer to the DLTF delete file command in the IBM supplied *Control Language Reference Manual*) or a request can be set to YES on the Make File Operational screen to delete the \$\$ version of the file during the batch process. Note that the request to delete a \$\$ version of a file will only appear if a \$\$ version of the file being recreated actually exists.

3.1.3 What Is an I/O Module?

I/O modules are RPG programs compiled by LANSA to:

- Support all input and output operations to IBM i database files set up by LANSA.
- Perform file and dictionary level validation rules against fields when adding, changing or deleting information in the files.
- Set field default values when inserting new records into the file and the user has not supplied the field value.
- Execute user defined RPG code to derive any virtual fields in the file definition.
- Automatically maintain batch, control or summary totals in other files.
- Perform data conversions if the data format requested (or supplied) by the user varies from the data format in the file.
- Emulate file I/O operations not supported in the RPG language.
- Provide database security.
- Protect database integrity.
- Provide consistent and easy to use error diagnostics.
- Provide I/O support to the RDML commands used by LANSA.

The use of I/O modules within LANSA "insulates" compiled user defined processes from the database definition and provides some distinct advantages:

- Centralization of validation rules logic. Dictionary and file level validation code is compiled into the I/O module and not repeated in every process that uses the file. Thus the validation rules can be changed without the need to recompile every process that accesses the file.
- True data independence. A file can be changed without the need to recompile every process that uses the file. This includes changing the definition of existing fields in the file as well as adding new fields to the file.
- Faster compiles of user defined processes because there is no direct reference to the external database definition.
- Support for virtual fields. These fields, derived from "real" fields in the file, appear as if they are in the database file, even though they don't really exist.

3.1.4 LANSA and Other System File Definitions

When a file definition is being set up under LANSA you will be requested to nominate who is to maintain the associated physical and logical database file definitions.

Two options are possible:

- The first is to nominate LANSA as the file maintainer. By doing this you are indicating that LANSA should set up and maintain (i.e. create, change and delete) any IBM i physical and logical files associated with the file definition. If this option is used the file must not already exist in the system, because you have elected to have LANSA create the file and maintain it in the future.
- The second is to nominate some OTHER system or mechanism as the file maintainer. In this case you are indicating that the physical file (and any associated logical files) already exist in the system and you only wish to load the definition of the files into LANSA and you will continue to maintain the definition of the file outside of the LANSA system. Once this has been done and the associated I/O module created, the file can be accessed through LANSA just like a normal LANSA maintained file, even though it is not one of the LANSA files in the strict sense.

When the OTHER option is used in a file definition several of the file definition facilities behave in a different manner. For instance:

- The "fields in a file" facility will display the fields in the file definition but will not allow them to be changed. This is because the file definition is not to be maintained (i.e. changed) by LANSA.
- The "logical views" facility will display the logical view definitions but will not allow them to be changed for the same reason as the preceding point. In addition select / omit criteria are not displayed for a logical file maintained by some OTHER system.
- The "make new or amended definition operational" facility will never actually perform an action that modifies the associated physical or logical database files in any way. Only the LANSA created file I/O module will ever be changed.

In all the cases just mentioned, a "reminder" message will appear on line 22/24 when the facility is used. This indicates that the file is maintained by some OTHER system and thus certain actions cannot or will not be taken.

Also when the OTHER option is used in a file definition, binary, date, time and timestamp fields will be supported. As these fields' support only exists for OTHER files, and Visual LANSA does not allow direct porting of OTHER files, this feature will only be supported for the IBM i operating system.

- Any binary field encountered will be enrolled in the LANSA data dictionary as a packed decimal field with the same number of digits and decimals as the source binary field.
- As far as LANSA is concerned the binary field is a packed decimal field at all levels above the I/O module (or I/O module routines in *DBOPTIMIZE or *DBOPTIMIZE-BATCH functions)
- This allows direct support of already existing files containing or keyed by binary fields, but not creation of files through LANSA containing or keyed by binary fields.
- Date, time and timestamp fields (IBM L, T and Z data types) found in the file are enrolled as alphanumeric fields with the same length as required by the formatted field (For example, a date field with *ISO format would be defined with a length of 10 characters).
- When date, time and timestamp fields are added to the LANSA Data Dictionary, default values consistent with those defined by IBM are assigned in the LANSA Data Dictionary. These default values must be reviewed to verify their suitability. Special care must be taken with date fields with 2digit years, as the resulting date may be different to the expected date based on the comparison year for century change within LANSA or IBM threshold year for century change (outside LANSA). For example year 01 may result in year 2001 instead of 1901. The timestamp default value is provided by the system variable *TIMESTAMP_DFT which returns a value of 0001-01-01-00.00.0000000.
- If date fields are used, it is strongly recommended that date formats which include the century be used. If that is not possible, then the LANSA's year for century change and the operating system's year for century change definitions must be consistent to prevent unpredictable results during date comparisons or validations. It must be noted that when a date field using a 2-digit year format (For example date format *JUL) is used in a key list, the sorting sequence is determined by the operating system (For example Julian date 40/001 is January 1st, 1940 and 39/365 is December 31st, 2039).
- Default values are only assigned when equivalent fields are not found during the load of the OTHER file. It is the developer's responsibility to ensure that

existing fields have default values which are valid date, time or timestamp values in the correct format.

• LANSA will handle these fields as alphanumeric fields with the only difference being that the I/O modules (or I/O module routines in *DBOPTIMIZE or *DBOPTIMIZE_BATCH functions) will validate that these fields contain valid date, time or timestamp values, before attempting to add or update records to the OTHER file. The developer must provide the logic (using existing LANSA features for handling dates, such as Built-In Functions) to operate these fields.

3.2 The File Control Menu

Access to the file control components of LANSA always starts from the file control menu. The file control menu is accessed from the main system menu:



pppFILMENU

File Control Menu

Enter number of function or place cursor on same line and press Enter.

- _ 1. Create a new file definition
 - 2. Review or change a file definition
 - 3. Delete a file definition
 - 4. Make new or amended file definition operational
 - 5. Print file definitions
 - 6. Exit from LANSA

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

3.3 Select File Definition(s) to Work With

Many of the options that can be invoked from the file control menu will request the name(s) of the files that you wish to work with (i.e. files to be reviewed/changed, printed, etc). In all cases this file selection method is identical.

Initially a screen is presented that requests the name(s) of the files to be worked with.

If you have been working with files before, then the name will be prefilled with the name of the last file you worked with.

SELFILE	Select File to Work with	
Enter full or partial name of the file definition(s) to be worked with or leave blank to select from all file definitions		
Fnn=Help Fnn=Exi	t Fnn=Cancel Fnn=Messages	

Using this display you can nominate the file(s) required in 3 different ways:

- **In full.** Enter the full name of the file required. If the file is found then the option chosen from the file control menu will be directly invoked for the file specified.
- **Partially.** Enter as much as desired of the partial name of the file. A search is made for all files which generically match the name specified and the resulting list is displayed.
- **Leave blank.** In this case a list of all files in the LANSA system is displayed from which the desired file(s) can be selected.

When a list of files is displayed the display will look like the screen below (which resulted from entering partial file name DEM)

F

SELFILESelect File to Work withEnter full or partial name of the file definition(s) to be we with or leave blank to select from all file definitions D	orked EM
Sel File Library Description _ DEMDIVN DC@DEMOLIB Table of valid compan _ DEMHEAD DC@DEMOLIB Demo Order Header H _ DEMLINE DC@DEMOLIB Demo Order Lines File _ DEMNAME DC@DEMOLIB Demo Names File _ DEMPROD DC@DEMOLIB Product master file	y divisions File O x e x O O x
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages	

An 'O' to the right of the file description indicates that the file has been compiled to make it operational.

An 'x' to the right of the file description indicated that this is an RDMLX file. Using this display you can:

- Alter the file name at the top right of the screen. The change can be to specify a full file name, a partial file name or a change to blanks. In all cases another search is made for the file(s) required.
- **Select** one or more files from the list by entering any non-blank character beside the file in the column labeled "Sel". If this is done then the option selected from the file control menu will be invoked for the file(s) selected.

3.4 Create a New File Definition

The steps involved in the creation of a new file definition are:



VIRTUAL BATCH	DATABASE LOAD
FIELDS CONTROL	ATTRIBUTES "OTHER"
& CODE LOGIC	FILE

A detailed description of the file definition menu and the processes that control the specification of the individual file components follow.

3.4.1 Specify the New File Name

When the option to create a new file definition has been chosen from the file control menu this screen is displayed. It requests that the name and description of the file to be created are input:

CRTFILE	Create New Fil	e Definition	
New file name Library of residence Description Initial public access To be maintained by	···· · · <u>NORMAL</u> ′ <u>LANSA</u>	NORMAL, ALL, NONE LANSA, OTHER	
Fnn=Help Fnn=Exi	··· ··· t Fnn=Cancel F	- Fnn=Messages	
L		0	

After details requested by this screen have been supplied, the file definition menu will be displayed. This allows the specification of the various components of the file definition (i.e. fields, validation rules, access routes, etc).

New File Name

Specifies the name by which the new file is to be known. Must be a valid object name. This name was once restricted to a maximum length of 8 characters but can now be 9 or 10 characters long. It is strongly recommended that 9 and 10 character file names be unique in their first 8 characters. This retains uniqueness in the formation of the \$\$ file name used in making the file operational and importing.

Combination of file name and library name must be unique. The use of the same

file name in different libraries is possible using LANSA. However it is strongly recommended that this procedure is NOT used. Instead it is better to install multiple LANSA systems to service different environments such as development and production. Refer to the *Installing LANSA on IBM i Guide* for more details.

Library of Residence

Specifies the library in which the new file is to reside. The library specified must exist and the user must be authorized to use it. This is checked by LANSA. When the library name is combined with the file name it must form a unique name for the file.

Avoid creating files that reside in your IBM i "current" library.

This parameter can be pre-filled to a default library value as specified in the current partition's definition.

Description

Specifies the description that is to be associated with this file. This is important as it is more often used than the file name when selecting the file for use in some way. A brief description must be specified.

Initial Public Access

Specifies what access other users are to be given to the file initially. The access to the file can be changed at any time later using the option available on the housekeeping menu. Mandatory. Pre-filled to NORMAL. Allowable values are:

- NORMAL Other users can read, add, change and delete records that are stored in this file, but they cannot modify the definition of the file in any way.
- ALL Other users can read, add, change and delete records that are stored in this file. In addition they can modify or delete the definition of the file.
- NONE Other users cannot access records stored in this file nor can they review or modify the definition of the file.

To Be Maintained By

Specifies who is to create and maintain this file definition. Prefilled to LANSA. Allowable values are:

LANSA Indicates that the actual physical and logical files that result from
this file definition are to be maintained (i.e. created, changed and deleted) by the LANSA system. If this option is used the file must not already exist in the library specified.

OTHER Indicates that the actual physical and logical files associated with this file definition are to be maintained (i.e. created, changed and deleted) by some OTHER system or mechanism. If this option is used the file must already exist in the library specified.

Use this option when you wish to load the definition of an existing file into LANSA, thus making the file accessible through LANSA, but do not wish to have LANSA responsible for the setup and maintenance of the associated physical and logical files.

If this option is used it is possible to use the facility "Load definition of file not maintained by LANSA" from the file definition menu to automatically load the definition of associated physical and logical into the LANSA system. Refer elsewhere for more information about this facility.

Note: You must specify *OTHER_DATETIME and *OTHER_VARCHAR options if the OTHER files that you want to made known to LANSA contain:

- date (L), time (T) or timestamp (Z) fields
- variable length fields.

For more information about these options, refer to OTHER file I/O modules in Compile and Edit Settings in the Review System Settings.

Copy File Named

Optional. Specifies the name of a file definition to be copied to create the definition of this new file. Copied details can be modified from the file definition menu.

Residing in Library

Optional. Specifies the library in which the file definition that is to be copied is located.

3.5 File Definition Maintenance

The steps involved in the review or change of file definition are:





A detailed description of the file definition menu and the processes that control the review/change of the individual file components is given.

3.5.1 Commit or Drop Amendments to a File Definition

If amendments have been made to a file definition, this screen will be presented after the Exit or Cancel function key has been used from the file definition menu:

COMMITAMD Commit File Amendment(s)

Commit amendment(s) YES YES, NO

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Enter YES to commit the file definition amendments to the LANSA system. Enter NO to drop all amendments. If NO is entered the file definition will revert to the form it had prior to the amendments being made.

After amendments have been made to the file definition it is then necessary to recreate the actual IBM i database file(s) from the amended definition. Refer to Making a New or Amended File Definition Operational for a detailed description of how this is done.

3.6 Delete a File Definition

The steps involved in the deletion of a file definition are:



3.6.1 Submit the Job to Delete a File Definition

When the option to delete a file definition has been selected from the file control menu and the file is eligible for deletion a screen like that following will result.

The sample screen shown is for a file CUSMST that has 2 logical views named CUSMSTV1 and CUSMSTV2:

DC@P200202	Delete a File from the System		
File : CUSMST	QGPL	Customer master file	
Submit this job as Using Job name	described	below	
Job description			
Job queue		*JOBD	

Object Lib Type Action to be taken / reason(s) I@CUSMST QGPL I/O Delete I/O module in user library User requested deletion of file definition and data CUSMSTV1 QGPL LGL Delete logical file in user library User requested deletion of file definition and data CUSMSTV2 QGPL LGL Delete logical file in user library User requested deletion of file definition and data CUSMST QGPL PHY Delete physical file in user library User requested deletion of file definition and data

WARNING : File definition and all data will be deleted

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The lower half of the screen details the actions that will be taken by the batch job and the reasons that they will be taken. In this example it can be seen that:

- The RPG program I@CUSMST which is the I/O module for file CUSMST will be deleted first.
- Logical views CUSMSTV1 and CUSMSTV2, which are based on file CUSMST, will be deleted next.
- Physical file CUSMST will be deleted last. Note that this step deletes ALL the DATA in CUSMST as well.

Input Options

Submit This Job Job Name Job Description Job Queue Output Queue

3.7 Make a New or Amended File Definition Operational

The steps involved in making a new or amended file definition operational can be visualized as:



3.7.1 Submit Job to Make File Definition Operational

When the option to "make a new or amended file definition operational" has been selected from the file control menu either of these screens may be presented first.

If LANSA cannot find any "reason" for making the file definition operational again (because it is not a new file definition and it has not been changed) the screen Create/Recreate a file from its definition will be presented first.

```
DC@P200203 Create/Recreate a file from its definition
```


No specific reason can be found to recreate this file. From the details below select the components of the file definition that are to be recreated:

Physical file Yes, No

Logical file(s) / view(s) _ Yes, No

I/O module Yes, No

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen asks that you manually specify the parts of the file that are to be recreated. Generally you would only use this screen if:

- The LANSA product vendor requests that the I/O module be recreated because of changes to the LANSA system.
- The IBM i physical or logical files or the I/O module associated with the file definition are accidentally deleted or become damaged in some way.
- You are re-installing your system after a total system failure and do not have valid backups of the physical and/or logical files.

Note that in the last 2 cases LANSA can recreate the physical and logical files but it cannot recreate the data they contain.

Select the desired part(s) of the file that are to be recreated by entering any nonblank character beside it.

If LANSA can find a "reason" for making the file definition operational, or if you have manually requested that parts of the file be recreated on the preceding screen, the screen Create/Recreate a File from its Definition will be presented.

DC@P200201 Create/Recreate a File from its Definition QGPL File : CUSMST Customer master file Submit this job as described below YES YES,NO Job description <u>*LIBL/QBATCH</u> Produce file and I/O module source listings NO YES, NO Ignore decimal error / Strip debug data options . . . <u>NO/YES</u> YES, NO User program to call at completion Delete \$\$ fileNO YES, NO Object Library Type Action to be taken / reason(s) QGPL PHY Create new physical file in library QTEMP CUSMST Physical file does not currently exist LGL Create new logical file in library QTEMP CUSMSTV1 QGPL Logical file does not currently exist I@CUSMST QGPL I/O Create new I/O Module in library QTEMP I/O module does not currently exist Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The lower half of the screen details the actions that will be taken by the batch job and the reasons that they will be taken. In this example it can be seen that:

- The physical file CUSMST will be created in QTEMP first.
- The logical view CUSMSTV1 will be created in library QTEMP next.
- The I/O module I@CUSMST is then compiled into library QTEMP.
- The list of actions continues. This is indicated by the "+" sign at the bottom right of the screen. If desired use the roll up and roll down keys to scroll backwards and forwards through the list of actions/reasons displayed.

Input Options

Submit This Job

Job Name

Job Description

Job Queue

WARNING: submitted batch job library list. Make sure that the job description being used does not cause the library in which the file resides to be made the IBM i "current library". Jobs submitted to make files operational where the "current library" and the library in which the file resides are the same may fail or produce unexpected results.

Output Queue

Produce File and I/O Module Source Listings

Specifies whether the DDS (data description specification) source listings produced by any CRTPF (create physical file) or CRTLF (create logical file) commands should be printed. In addition it specifies whether the RPG compiler listing of the I/O module program should be printed.

Mandatory. Prefilled to YES. Allowable values are:

YES Produce DDS and RPG compiler listings.

NO Do not produce DDS and RPG compiler listings.

Ignore Decimal Error

Specifies how decimal data errors should be dealt with in the compiled RPG program that results from the "function control commands".

How this feature is implemented depends on the version of RPG code being compiled. For more information on the different versions of RPG code that can be compiled refer, to ILE Implementation.

This facility is provided for compatibility with the operating system and because it is required by some installations. It is strongly recommended that this option is **not** used. Refer to the IBM supplied CL reference manual for details.

Mandatory field. Default value is determined from the system definition data area. Refer to System Definition Data Area DC@A01. Allowable values are:

• If RPG/400 Code Is Being Compiled

YES Use the IGNDECERR(*YES) parameter.

- NO Use the IGNDECERR(*NO) parameter.
- If RPG/IV Code Is Being Compiled

YES Use the FIXNBR(*ZONED) parameter.

NO Use the FIXNBR(*NONE) parameter.

Strip Debug Data Options

Specifies whether or not the RPG I/O module associated with the file definition should have its associated debug symbolic information removed. This information is only required in two situations:

• When attempting to use the IBM supplied debugging aids with the compiled RPG I/O module.

Since these situations are relatively rare, the default for this field is YES (debugging information should be stripped). By using this option, the size of the compiled I/O module will typically be reduced by 40 to 60%. This size reduction has no bearing on execution speed, just on the size of the compiled object.

To enable RPG I/O modules to be debugged, use option NO (debug information should not be stripped).

User Program to Call at Completion

Specifies the qualified name of a user defined program that should be called after the file has been successfully created or recreated. Typically this facility is used to execute a program that initializes a new file or modifies the data in a file that has been amended.

Optional. If used the program name must be specified as a standard object name (i.e. <program>.<library>) Special value *LIBL (search library list) is acceptable as the name of the library that is specified. LANSA does not validate the name or format of the program specified in any way.

When the program is called, it is passed two parameters:

Туре	Length	Dec	Description
А	10		Name of physical file created or recreated.
A	10		Name of library in which physical file resides.

If the user program fails the create / recreate job will end abnormally. Messages on the job log will indicate the cause of the failure. However, the file will have been successfully created or recreated and will be usable. In this case correct the problem in the user program and independently resubmit for execution.

Delete \$\$ File

Specifies whether or not the \$\$ version of the file should be deleted during the batch process. The request only appears if the \$\$ version exists. If NO is specified (which is the default), then the submit to batch will not take place and a message will be issued to say that the \$\$ version must be deleted. It is still possible to delete it manually using the DLTF command.

Where 9 and 10 character file names are not unique in their first 8 characters, a \$\$ version for a different file may erroneously be found to exist. It still must be deleted.

3.8 Print File Definitions

The steps involved in printing file definition(s) are as follows:



3.8.1 Submit Print Job to Batch

When the option is used to print file definitions this screen is displayed to enable a batch job that will perform the actual printing to be submitted:

DC@P200401 Print File Definition
Submit this job <u>YES</u> YES, NO
Using Job name
Print all files or selected files <u>ALL</u> ALL, SEL Print access routes and batch control logic <u>NO</u> YES, NO Print processes using files cross-reference <u>NO</u> YES, NO
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Input Options

Submit This Job Job Name Job Description Job Queue Output Queue

Print All Files or Selected Files

Specifies whether all files defined to LANSA are to be printed or only fields selected using the standard file selection method described elsewhere.

Mandatory field. Prefilled to SEL. Allowable values are:

- SEL Only selected files should be printed.
- ALL All files defined to LANSA should be printed.

Print Access Routes and Batch Control Logic

Specifies whether the access routes and any batch control logic associated with the file definition should be printed. Using this option can considerably increase the length of the report.

Mandatory field. Prefilled to NO. Allowable values are:

YES Print access routes and batch control logic

NO Do not print access routes and batch control logic

Print Processes Using Files Cross-Reference

Specifies whether a cross-reference list of which user defined processes use the selected file(s) should be included on the report. Using this option can considerably increase the length of the report.

Mandatory field. Prefilled to NO. Allowable values are:

- YES Print process cross-reference.
- NO Do not print process cross-reference.

3.9 Bulk Load of OTHER files

"OTHER" files are non-LANSA files maintained by another system. Before an OTHER file can be used by LANSA, it must be "made known" to LANSA. This can be done using the Bulk Load of OTHER files process described here, or as a single file process as described in 3.19 Load the Definition of a File NOT Maintained by LANSA.

As well as loading new definitions, you can use this process to update definitions that have already been loaded.

To enable this facility for a partition, you must use the LANSA Import facility and import from save file LS@BULKLOD.

The Bulk Load of OTHER files facility can be accessed from:

- an option on the File Control Menu
- using command key on the Work with Files in the advanced menu.

There are three simple steps in the bulk load (or update) process.

These steps are:

- 1. Specify the Library and a file name, or partial filename, of the OTHER files to be made known to LANSA. A list of files that match this criteria is created.
- From the list of files that match the criteria in step 1, select the files for which the definitions are to be loaded or updated. You can also display the file details before you load them.

Specify the details to be used for the batch job that will load the definitions.

For full details on loading OTHER files, and factors you must consider, refer *to* 3.19 Load the Definition of a File NOT Maintained by LANSA.

When the Bulk load of OTHER files option is selected from either the File Control Menu or the Work with Files screen, you will be requested to specify the library and file(s) so that a list of files to be selected can be created.

LS@0001113 Bulk Load of OTHER files

Enter library which contains files <u>PAYLIB</u> Enter full or partial name of file(s) to be worked with or leave blank to select all <u>PR</u> Fnn=Help Fnn=Exit Fnn=Prompt Fnn=Cancel Fnn=Messages

Enter the search criteria for the OTHER files which you wish to work with. The criteria consists of a library, which must be entered, and a file name.

The file name may be a full name, a partial name or blank. All OTHER physical files which match the search criteria and their associated logical files are displayed.

```
LS@0001158
                 Bulk Load of OTHER files
 7=Load definition
                        27=Display details
  File
            Library
                     LANSA
7 PF PR0001
                            Y
                                Personnel Details
                PAYLIB
  LF PR0001L1 PAYLIB
<u>7</u> LF PR0001L2
                PAYLIB
                             Y
<u>7</u> PF PR0002
                              Employee History
                PAYLIB
  LF PR0002L1
                  PAYLIB
7
  LF PR0002L2
                  PAYLIB
Fnn=Help Fnn=Exit Fnn=Prompt Fnn=Cancel Fnn=Messages
```

The file details displayed are:

- PF indicates that this is a physical file, LF indicates that this is a logical file belonging to the preceding physical file.
- The IBM i file name.
- If the OTHER file definition has already been loaded into LANSA, the LANSA column will contain Y.
- The description of each physical file.

The details of any file can be displayed by entering option 27 (Display Details) to the left of the file. The IBM i DSPFD details will be shown.

To load or re-load the definition of a file, enter 7 to the left of each of the required files. A logical file can not be loaded by itself. It must be loaded with its associated physical file. However, not all logicals need be selected. You may select only those which you wish to use in LANSA.

You will be requested to enter the details to be used by a batch job which will be submitted to load the definitions of the selected files.

LS@0001198	LOAD/CREATE OTH	HER FILES	i i
Submit batch load of Using Job name Job description Job queue Output queue .	of files selected <u>LS</u> QUS <u>YOB</u> <u>VSF</u>	<u>YES</u> G@BULKL(SRSYS/DC D RSYS/PRT(YES, NO OD @JOBD)2
Submit batch create OPTIONS FOR BA Produce file and I/C Ignore decimal erro User program to cal	of files after load TCH CREATE) module source listing r / Strip debug data opt ll at completion	<u>YES</u> s <u>NO</u> ion <u>NO</u>	YES, NO YES, NO <u>YES</u> YES, NO
Fnn=Help Fnn=Pro	ompt Fnn=Messages		

If you enter YES for "Submit batch create of files after load", a batch job to make the file operational will be submitted for each physical file successfully loaded.

These batch jobs will use the same Job description, Job queue and Output queue as specified for the bulk load job. The Job name will be the file name.

3.10 The File Definition Menu

The file definition menu is used when creating a new file definition or reviewing/changing an existing file definition. The method by which the file definition menu is used can be visualized as follows:





The file definition menu is the menu that is always used to create, review or change any of the components of the LANSA file definition and looks like this:

pppFILCRT File Definition Menu (aaaaaa)

Enter number of function or place cursor on same line and press Enter.

- _ 1. Review or change fields in file definition
- 2. Review or change logical views of file
- 3. Review or change file validation checks
- 4. Review or change access routes to other files
- 5. Review or change virtual fields
- 6. Review or change batch control logic
- 7. Review or change data base file attributes
- 8. Review or change file multilingual attributes
- 9. Load definition of file not maintained by LANSA
- 10. Exit from LANSA

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note that the "aaaaaa" value shown in the screen title is actually displayed as Create or Review to indicate what action is being taken on the file definition. By working from the file definition menu you can define, review or change the fields in a file, logical views, validation rules, access routes, batch control logic, database file attributes and load the definition of file not maintained by LANSA. Each of these components of the file definition is described in detail in the following sections.

3.10.1 Select Fields When Working from File Definition Menu

Many of the processes that work from the file definition menu will request that field names be specified. For instance which fields are to be included in the file definition?, which fields form the key of the logical file?, etc.

When requested to specify a field name it can usually be done in these ways:

- In full. Enter the full name of the field required. If the field is found then it will be accepted with no further action.
- Partially. Enter as much as desired of the partial name of the field. A search is made for all fields which generically match the name specified and a list is displayed for subsequent selection.
- As a "?". In this case a list of all fields that are available is displayed from which the desired field can be selected.

Note: If literals and system variables are valid entries for the context only the "?" can be used to display the selection list.

The fields that are included in the list will vary with the context in which the fields are being used. For instance, if specifying the fields to be included in the file definition then list of fields will come from the LANSA data dictionary. However, if specifying the key fields of a logical view then the list will come from the fields in the file definition since a file key must be one of the real fields in the file. The list will contain both the real and virtual fields in the file when specifying access route key fields.

The selection display may allow a single or multiple selection of fields depending on the context. For instance when adding real or virtual fields to the file definition a multiple choice may be made, but only a single choice when selecting access route or batch control keys.

Single Field Selection

When only one field is to be selected the list of fields will fill the full screen.

DC@P800501 Select Field from Data Dictionary

Either Choose field required from list below,

Or Enter a new full or partial search name : <u>DEM</u>



Search Name

A new name may be entered to build a new generic list. It may be built from the data dictionary, the real fields in a file definition, or the real and virtual fields in a file definition, depending on which attribute of the file is being defined.

Sel

To add a field to the definition, type any non blank character in the 'Sel' column and press enter.

R/V

When the selection list is built from the fields in a file definition an 'R' for real or a 'V' for virtual will be displayed against each field in the 'R/V' column.

Multiple Field Selection

When a multiple selection is allowed the screen is split horizontally in two. The number of fields which can be selected at one time depends on the context in which the selection is made.

```
DC@P800501Select Field from Data DictionaryType option, press enter4=Delete from current listAct Seq FieldDescriptionTyp Len Dec10DEMAD3Address Line 3A 25
```

Full/Partial name of fields to be displayed from Data Dict Type option, press enter 1 to 99999=Select to add to current list					
Sel Field D	escription	R/V	Type L	len Dec	
_ DEMAD1	Address line 1		Α	25	
_ DEMAD2	Address line 2		А	25	
_ DEMAD3	Address line 3		А	25	
_ DEMCAT	Product category		Α	10	
Fnn=Help Fnn=Cancel Fnn=Messages					

1

The top half of the screen contains the current work list to which selected fields are added.

Act

1

The action field allows fields to be deleted from the current list by entering option 4 and pressing the enter key.

Seq

The sequence number allows the order of the fields to be changed. The fields will be returned to the calling program ordered by the sequence number. In the case of fields being added to a physical file definition the new and existing fields are sorted by these sequence numbers. If several fields have the same sequence number they are sorted alphabetically.

The bottom half of the screen contains the selection list. It may be a full or generic list, and may be built from the data dictionary, the real fields in a file definition, or the real and virtual fields in a file definition depending on the calling program.

Search Name

A new name may be entered to rebuild the list. If a generic name is entered which is unique, its matching field is added to the current work list automatically. If a generic name is a valid name but has other generic matches, it is automatically added to the current work list and the generic selection list is also displayed.

Seq

The sequence number (1 to 99999) allows the selected fields to be ordered and added to the current work list. If several fields are given the same number they are ordered alphabetically.

R/V

When the selection list is built from fields in a file definition an "R" for real or "V" for virtual is displayed against each field.

Partial/Generic Names

Partial/Generic names may be classified in three different ways.

Туре	Description
1	A partial name which is matched by only one full name,
2	A partial name which is a valid name, and has several matches,
3	A partial name which has several matches.

As an example of a type 2 partial name, 'DATE' could be a valid field name, but this may also be generic because field 'DATETIME' may exist.

All type 1 and 2 generic names are automatically added to the working list in a multiple selection list.

Each generic field type causes different actions by the selection program as shown below. The multiple field selection can be activated by a single entry or by several partial names of different types, and will act according to which type they are.

Туре	Single select	Multiple select Single partial name.	Multiple select. Multiple partial names
1	Selection list not displayed matching name returned	Selection list not displayed matching name returned	Matching name added to working list selection list displayed
2	Selection list displayed. See	Selection list displayed, valid	Selection list displayed, valid name

	note below.	name added to working list.	added to working list.
3	Selection list displayed	Selection list displayed	Selection list displayed.

Note when a valid but generic name (type 2) is used in a single select condition, the select screen containing the generic list is displayed. If no selection is made the name is retained on the calling screen but the 'no selection made message' is displayed even although the name is valid. To exit F12 must be used as enter key will again display selection list. This is because the program cannot interpret the user's intention in entering the name as a generic or valid name as it is both.

Comments/Warnings

For response reasons, if the LANSA data dictionary for the current partition contains a large number of fields, it is preferable to use partial names rather than '?'s.

3.10.2 Select Files When Working from File Definition Menu

Some of the processes that work from the file definition menu will request that other file names be specified, for instance the file to which this access route links or the file which is the batch control file.

When requested to specify a file name it can usually be done in any of these ways:

- In full. Enter the full name of the file required. If the file is found (and exists in only one library) then it will be accepted with no further action.
- Partially. Enter as much as desired of the partial name of the file. A search is made for all files which generically match the name specified and a list is displayed for subsequent selection.
- Leave blank. In this case a list of all files that are available is displayed from which the desired file can be selected.

When a list of files is displayed it will look like this example which resulted from specifying partial name DEM as the name of a file to be accessed via an access route:

DC@P800901 Select Required File

Either choose file required from the list below **Or** enter a new full or partial search name : **DEMH**

Sel File Library Type Description and other details

 DEMHEAD DC@DEMOLIB PHY Demo Order Header File Keyed by DEMORD - Order number
 DEMHEADC DC@DEMOLIB LGL Order header by customer, order Based on physical file DEMHEAD Keyed by DEMCUS - Customer number DEMORD - Order number

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

From this display a file can be selected or a new search made by entering a new full or partial name in the top right of the screen.

3.11 Fields in the File Definition

The steps involved in defining, reviewing or changing fields in a file definition are as follows:



3.11.1 Fields in File Definition Maintenance

This screen is displayed when:

- Reviewing the fields in a file definition for possible change.
- Defining the fields in a new file definition.

DC@P200501 Display Fields in File Definition

	Primary				
Seq	Name	Description	Key order Src		
10	NAMCDE	Name and address code	1		
20	NAME	Name of person/company/	organisation		
30	ADDR1	Address line 1			
40	ADDR2	Address line 2			
50	ADDR3	Address line 3			

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

Working from this screen you can:

- Review the fields already in the file definition.
- Specify details of new fields to be included into the file definition. Use the Add function key to indicate that new fields are to be added to the file definition, then specify details of the new fields.
- Change the sequence or primary key order of fields already in the file definition. To do this use the Change function key and make all desired changes.
- Delete fields from the file definition. To do this use the Delete function key. The resulting display will request that the field(s) to be deleted should be selected. A final display will then request that the selected field(s) be deleted. The delete request must be confirmed by entering YES. If YES is not entered no fields will be deleted from the file definition.

Remember that field names can be specified in full, partially or chosen from the data dictionary (by entering a "?").

WARNING: Before deleting fields from a file, print the file definition report. Other files may use fields in this file as the source for Predetermined Join Fields or for batch control. These other files will not compile if the required fields are deleted from this definition.

Input Options

The following input options apply when specifying the fields in a file definition:

Seq

Specifies the sequence/order of the field relative to other fields in the file. Use numbers in the range 1 to 99999 to indicate the order that fields should be included into the file record format.

Fields are ordered from low to high by sequence when they are placed in the file record format. When no sequence number is specified, sequence number 99999 is assumed. When fields are assigned the same sequence number they are sub-ordered by the order in which they are specified on the screen.

Field Name

Specifies the name of a field that is to be included in the file definition. Any field used in a file definition must be defined in the LANSA data dictionary before it can be included in a file definition.

By entering a partial name or "?", or by entering a sequence number or key sequence and blank field name, the multiple field selection screen will be displayed. The list of fields is built from the LANSA data dictionary. It allows up to 30 fields at a time to be added to the file definition. The sequence number used in the working list will be used to insert the selected names into the definition. Refer to prior section, 3.10.1 Select Fields When Working from File Definition Menu for more details.

The complete list of all fields in the file definition form the file record. There is no effective limit to the number of fields that can be included in a file definition. However the net length of a file record (calculated by adding together the byte lengths of all fields in the file definition) should not exceed 9996 bytes. This is one of the LANSA restrictions.

LANSA automatically includes one field into every file it creates. The field is called @@UPID and is used by the LANSA database I/O modules to check for "crossed updates". Although this field is defined in the LANSA data dictionary

it should NEVER be manually included into a file definition. See The @@UPID Field in LANSA Created Files for more details regarding the use of this field by LANSA.

Primary Key Order

Specifies the field(s) that form the primary key of the file. Fields are indicated as being part of the primary key by the entry of a number beside them in the "Primary key order" column. In addition the number specifies the order / hierarchy of the primary key.

Fields that form the primary key of the file should be numbered from 1 (first key in hierarchy / most significant key) to "n" (last key in the hierarchy / least significant key).

Note that the fields do NOT have to be defined into the file in primary key order. Nor do primary key fields have to be defined one after another. Primary keys are identified by a number and ordered by the relative sequence of the number.

The primary key of a file (which is composed of all the fields in the key concatenated in the order specified) must be **UNIQUE.** No more than one record in a file can have any given primary key. This rule is enforced by features in the operating system (DDS keyword UNIQUE) and can never be violated. Attempting to add a record to a file with the same primary key as a record already in the file will result in a "duplicate key error". LANSA automatically handles duplicate key errors and there is no need for user logic to handle or check for them.

The existence of a unique primary key is important to LANSA because it uniquely identifies one and only one record in a file.

There is no requirement to specify a primary key when setting up a file definition, but is strongly recommended that each and every file defined under LANSA has a unique/primary key.

Src

Indicates the "source" of the field definition when reviewing the definition of a field within a file. Normally this field is blank indicating that details of the field were input manually or loaded from an external (OTHER) file definition. If a value appears in this field it indicates that the details of the field were input via the LANSA Built-In Function called FILE_FIELD. Refer to the *Technical Reference Guide* for more details of this (and other) Built-In Functions that

allow file definitions to be created and edited directly from RDML programs. The value shown in this field is identical to the "source" argument given to the START_FILE_EDIT Built-In Function when automatic editing of this file definition was performed.

Examples - Fields in a File Definition

These examples apply to specifying the fields in a file definition.

Define th	he fields	in a name	and address	file
-----------	-----------	-----------	-------------	------

Seq	Field	Description	Primary Key Order
1	NADCD	Name and address code	1
2	NAME	Name of person / organization	
3	ADDR1	Address line 1	
4	ADDR2	Address line 2	
5	ADDR3	Address line 3	
6	POSTCD	Post code	

Define the fields in an order header file

Seq	Field	Description	Primary Key Order
10	ORDNUM	Order number	1
20	NADCOD	Customer name & address code	
30	DATDUE	Date order is due	
40	COMPFL	Order completion flag	
50	USERID	User who entered order details	

Define the fields in an order lines file

	Seq	Field	Description	Primary Key Order
	10	ORDNUM	Order number	1
	20	PRODNO	Product number	
1				
30	QUANTY	Quantity ordered		
----	--------	---------------------	---	
15	ORDLIN	Order line number	2	
40	DISCNT	Discount percentage		

3.12 Logical Views/Files Associated with File Definition

The steps involved in defining, reviewing or changing the logical views associated with a file definition are:



3.12.1 Display Existing Logical Views

This screen is displayed when:

- Reviewing the logical views in a file definition for possible change.
- Defining the logical views in a new file definition.

DC@P200601 Secondary Logical Views

Select logical view to be reviewed, changed or deleted **Or** use the ADD command key to add a new logical view

Select Logical View Description

	0	1
_	XXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
_	XXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
	XXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
_	XXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
—	XXXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
—		

Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Cancel Fnn=Messages

Working from this screen you can:

- Select an existing logical view for detailed review and/or change. To do this enter any non-blank character in the column labeled "Sel" against the logical view required.
- Create a new logical view. Use the Add/Create function key to indicate that a new logical view is to be added to the file definition.

3.12.2 Detailed Logical View Maintenance

This screen is displayed when:

- Reviewing an existing logical view in detail
- Defining a new logical view in the file definition

DC@P2006	502	XXXXX	X Logica	l View	
File : XXX	XXXXXXX	x xxxxx	XXXXX	xxxxxxxxxxxx	XXXXXXX
Logical vie Description Access path Key field d	w name of logical maintenan etails :	: view : ice option:	IMMED	Unique? <u>NO</u> Dynamic	c select? <u>NC</u>
Field	Descripti	on		A/D S/U/A	
	xxx	XXXXXXX	ххххх	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXX
	XXX	XXXXXXX	XXXXXX	XXXXXXXXXXXXXXX	XXXXXX
		XXXXXXX	XXXXXX	****	XXXXXX
	//////	())))))))))))))))))))))))))))))))))))))			
Select/Omi	t critoria ·				
	SFLFC	Γ/ΟΜΙΤ	Field	Operation(s)	
	JLLLU		riciu	Operation(3)	
	·				
					+
Fnn=Help	Fnn=Exit l	Fnn=Cance	l Fnn=M	essages	

Working from this screen you can:

- Review the full details of an existing logical view.
- Change an existing logical view. Use the Change function key to indicate that the logical view should be altered. The screen will be re-displayed and

all fields available for change, except for logical view name. Validation of the logical view will follow the same rules as the Add/Create option.

- Delete an existing logical view. Use the Delete function key to indicate that the logical view should be deleted. On the resulting display enter YES to confirm that the logical view should be deleted. If YES is not entered the logical view will not be deleted.
- Specify details of a new logical view that is to be created.

Input Options

These input options apply when creating a new logical view in a file definition:

Logical View Name

Specifies the name that is to be assigned to the logical view/ file. Logical file names must be unique. No other physical or logical file can exist in the same library with the same name.

This name was once restricted to a maximum length of 8 characters but can now be 9 or 10 characters long. It is strongly recommended that 9 and 10 character file names be unique in their first 8 characters. This retains uniqueness in the formation of the \$\$ file name used in making the file operational and importing.

Note that no library name is required. LANSA will always create the logical file in the same library as the associated physical file. This is the library that was specified when the file definition was first created.

A naming standard that is frequently adopted requires the logical file name to contain or relate to the associated physical file name. This is to prevent confusion amongst users when they are accessing the physical or logical files.

For instance, the physical file (and LANSA file definition) of a customer master file might be named CUSMST. The associated logical views might be called:

CUSMSTV1 Customers ordered by customer name

CUSMSTV2 Customers ordered by post code

CUSMSTV3 Customers ordered by state

Description of Logical View

Specifies a description that is to be associated with the logical view/file. This field is an identification aid to any user of the file. Wherever possible include information in the description that specifies what the logical view/file should be used for. e.g. "Customer master ordered by post code" or "Orders by customer

number, order number"

Access Path Maintenance Option

Specifies what method is to be used to maintain the "access path" associated with this logical view/file. When a logical view is created to arrange the information in a file into a specific order the system creates an "access path".

The access path is essentially the logical file "key fields" arranged in a special structure that allows:

- Extremely fast access to any individual record in the file using just the key(s) of the logical file.
- The processing of the records in order of the logical file key(s) without the need to first sort the file.

An access path exists for every logical view/file created. Every time a record in the file is added, deleted or updated the operating system must update all the access paths to this file. To do this it must use up part of the computer's time and available processing cycles. This of course degrades the performance of the system. The more access paths that are being maintained, the slower the system will run.

This field allows the type of access path maintenance to be used to be manually specified. Allowable values are:

- IMMED The access path should be maintained immediately (i.e.whenever a record is added, updated or deleted from the file). This option is the most common. If the logical file is to be used in an interactive environment then IMMED should always be used.
- DELAY The access path should only be maintained when the logical file is used. This type of access path lies dormant and is not maintained by the operating system until such time as someone needs to use the logical file. Thus it places no burden on the operating system until it is actually required. Typically logical views that use this option are only used once a week (say) to "sort" a file into a specific order.

Unique?

Specifies whether or not the key fields nominated are to form a unique key to the file.

The default value is NO, which indicates that the key fields nominated do not form a unique key to the file. This allows multiple records with the same key to exist in the file. The only other possible value is YES, which indicates that the key fields are to form a unique key to the file. This means that one (and only one) record can exist in the file for any given key value.

When defining a new logical file over a physical file that already contains records and using the YES option make sure that there are no duplicate records (i.e.key values) already in the file. If duplicate records do exist, then the "make operational" job will fail as this logical view cannot be loaded because of the duplicate records. The job log will indicate the cause of this problem. To correct, remove or Change the duplicate records.

Dynamic Select?

Specifies whether or not any select/omit tests specified on the lower portion of the screen are to be done at execution time.

The default value is NO, which indicates that the dynamic select feature should not be used. In this case the access path associated with the logical file will contain only records that match the select/omit criteria specified.

The only other possible value is YES, which indicates that the dynamic select feature should be used. In this case the access path associated with the logical file will contain all records in the file. The select/omit testing should be done when the program reads the records from the file.

If you specify YES, then do not specify any select/omit tests the value will be automatically changed back to NO.

The dynamic select facility is a feature of the operating system. Using it can have significant overall performance benefits in some situations. For more information about this facility refer to the appropriate IBM supplied manual.

Key Field Details

Specifies the fields that comprise the logical file key and the order that the key information should be stored. Key fields are specified from top to bottom in order of their importance in the key hierarchy (i.e.major to minor). Major keys come first. The net length of the logical file key (which is the sum of the lengths of all fields that comprise the key) cannot exceed 255 bytes (characters).

Although only the first 3 key fields are shown (or can be input) on the display the roll key can be used to review or input additional key field details.

Field

Specifies the name of one of the field(s) in the logical file key. Any fields named as a key must first be defined as a field in the file definition. Fields can be selected by entry as a full or partial name or as a "?", which causes the

multiple selection screen to be displayed. If the field name is left blank and the A/D or S/U/A flags are changed this will also cause the multiple selection screen to be displayed. The list will contain all the real fields in the current file definition. A maximum of 20 keys can be specified. Refer to 3.10.1 Select Fields When Working from File Definition Menu for details.

Description

Displays the description of the key field. This is obtained from the LANSA data dictionary.

A/D

Specifies whether the associated key is to be stored in ascending or descending sequence (i.e.from lowest to highest or highest to lowest order). Allowable values are:

- A (Ascending) store in lowest to highest order
- D (Descending) store in highest to lowest order.

This is an optional field. If not specified, A is assumed.

S/U/A

Specifies, for numeric key fields only, additional details about how the key is to be ordered. Allowable values are:

- S (Signed) indicates the numeric fields should be stored taking into account their signs (+ or -).
- U (Unsigned) indicates the numeric field should be stored without taking into account their signs. The numeric field is to be treated just like a character field.
- A (Absolute value) indicates that the numeric field should be stored by its absolute value. Note that this is not the same (and does not always produce the same order) as the U option.

Character fields are always U (unsigned) fields. If you specify something other than U for a character field it will be automatically changed to U.

This field is optional. If you do not specify it will default to U for character fields and S for numeric fields.

These additional options are supported by the operating system . For more details about them refer to the IBM manual *Data Description Specifications*. Refer to logical file keywords SIGNED, UNSIGNED and ABSVAL.

Select/Omit Criteria

Specifies any record selection or omission criteria that is to apply to the logical view/file. If a record in the physical file is "omitted" from a logical file then it is effectively "invisible" to any user accessing the file via the logical view.

A record can be omitted for 2 reasons:

- It fails to match the selection criteria
- It matches the omission criteria

Select/omit criteria are a powerful facility but should be used with some caution because the effect of making some records in the physical file "invisible" when accessed via certain logical views can cause confusion amongst some users of the system.

Select/omit criteria are entered as a series of select/omit statements. When both SELECT and OMIT are used the order of the statements is important. The statements are processed in the order specified. If a record matches the criteria of a statement it is selected or omitted as specified and following statements are not tested.

Extensive examples of select / omit criteria follow this section.

AND/OR

This is an output field only. It indicates whether the select/omit statement is ANDed or ORed with this statement.

SELECT/OMIT

Indicates whether the condition specified in the operation(s) field is to be used to select or omit records when found to be true. Allowable values are:

SELECT Use operation to select records

OMIT Use operation to omit records

Blank Form an "and" with previously SELECT/OMIT operation

When using select/omit statements specify SELECT,OMIT or blanks in the SELECT/OMIT column. Using SELECT or OMIT implies an OR relationship with the preceding select/omit statement. Leaving the entry in the SELECT/OMIT column blank implies an AND relationship with the preceding select/omit statement.

Field

Specifies the name of the field that is to be used in conjunction with the

operations(s) field to evaluate the select/omit expression. Field named must be defined in the file definition. In addition it can be selected by entry as a full or partial name or as a "?", which will cause the single field select screen to be displayed. The list will contain all the real fields in the current file's definition. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.

Operation(s)

Specifies the operation that is to be performed against the field nominated in the "field" field. Allowable formats are

- COMP(<operator> <value>) which indicates that the nominated field should be compared using <operator> to the <value>. Allowable values for <operator> are:
- EQ Equal to
- NE Not equal to
- LT Less than
- NL Not less than
- GT Greater than
- NG Not greater than
- LE Less than or equal to
- GE Greater than or equal to
- The value specified for <value> can be a character literal in quotes (e.g.
 - 'BALMAIN'), a hexadecimal literal in quotes (e.g. X'F1E6'), a numeric literal (e.g. 1.54) or the name of another field in the file definition.
- **RANGE(<low value> <high value>)** which indicates that the nominated field should be tested against the range of values <low value> to <high value>. The <low value> and <high value> specified can be a character literal in quotes (e.g. 'BALMAIN'), a hexadecimal literal in quotes (e.g. X'F1E6'), a numeric literal (e.g. 1.54) or the name of another field in the file definition.
- VALUES(<value1> <value2> <value100>) which indicates that the nominated field should be compared with the list of values specified. Up to 100 values can be specified in the list of values. The value fields specified

can be a character literal in quotes (e.g. 'BALMAIN'), a hexadecimal literal in quotes (e.g. X'F1E6'), a numeric literal (e.g. 1.54) or the name of another field in the file definition.

• **ALL** which is only ever used as the last select/omit statement associated with a logical view/file. It indicates what is to happen after all other select/omit statements have been processed. If used with SELECT then all records not meeting the select/omit statements will be selected. If used with OMIT then all records not meeting the select /omit statements will be omitted.

If the ALL condition is NOT specified as the last statement in a set of select/omit statements then a default value is assumed. The default value is the reverse of the last select/ omit statement specified. If the last select/omit statement is SELECT then a default of OMIT ALL is assumed as the last statement. Likewise if the last select/omit statement is OMIT then a default value of SELECT ALL is assumed as the last select/omit statement.

Where a condition will not fit on one line it can be continued on the next line by entering a "+" sign as the last character on the line. The "+" indicates that the line continues with the first non-blank character on the next line.

A normal example of using continuation lines would be the following which is considered to be one select/omit statement even though it takes 3 lines to enter:

AND/OR SELF	ECT/OMIT Fie	eld Operation(s)	
SELECT_	NAME	VALUES('ACM	IE & CO' +
	· '	SMITH ENGINEE	RING' +
	'	ACME ENGINEEF	RING')

Examples - Create Logical Vew

These examples apply to creating a logical view within a file definition. The file definition used in these examples is called CUSMST (Customer master file) and contains these fields:

Field	Description	Туре	Len	Dec
CUSTNO	Customer number (primary key)	S	7	0
NAME	Name of person / organization	А	30	
ADDR1	Address line 1	А	35	
ADDR2	Address line 2	А	35	
ADDR3	Address line 3	А	35	
POSTCD	Post code	А	4	
STATE	State mnemonic	А	3	
ACTIVE	Active / inactive flag	А	1	
CREDIT	Credit limit	Р	11	2
AMTDUE	Current amount due all accounts	Р	11	2
ACCTYP	Customer account type	Р	1	0

Example 1

Create a logical view/file called CUSMSTV1 that will order CUSMST by NAME and allow generic searching by NAME:

Example 2

Create a logical view/file called CUSMSTV2 that will order CUSMST by STATE then POSTCD (i.e.POSTCD within STATE):

Logical view name : CUSMSTV2

Desc of logical view : Cust master by state and postcode Access path maint opt : IMMED Unique? NO Dynamic select? NO Key field details Field Description A/D S/U/A STATE

STATE_____ POSTCD______

Note that no ascend/descend value has been specified. In this case the value will default to ascend for both keys.

Example 3

Create a logical view/file called CUSMSTS1 that will order CUSMST by CUSTNO. Only records with ACTIVE = 'Y' are to be "visible" when using this logical view:

Logical view name : CUSMSTS1 Desc of logical view : Active customers by customer no Access path maint opt : IMMED Unique? NO Dynamic select? NO Key field details: Field Description A/D S/U/A CUSTNO________

_____ _ _ _ + ____ _ _ +

Select/Omit criteria:

AND/OR SELECT/OMIT Field Operation(s) SELECT_____ACTIVE___COMP(EQ 'Y')_____

Example 4

Create a logical view/file called CUSMSTS2 that will order CUSMST by AMTDUE. Only records with CREDIT greater than 100000 and AMTDUE greater than or equal to 20000 are to be visible via this view. Records are to be

ordered from highest amount due to lowest amount due. Use the dynamic select feature as well.

Logical view name : CUSMSTS2

Desc of logical view : High credit, large debt customers Access path maint opt: IMMED Unique? NO Dynamic select? YES Key field details:

Field	Description	Α/	D	S/U	/A
AMTD	UE	Γ)	S	
		_	_	-	ł
		_	_	-	ł
			+		

Select/Omit criteria :

AND/OR	SELECT/	OMIT	Field	Operation(s)
SEL	ECT	CREE	DIT	COMP(GT 100000)
AND _		AM	TDUE_	COMP(GE 20000)

Note that since no entry was made in the SELECT/OMIT column for the second select/omit statement the 2 statements are ANDed together. In this case only customers with CREDIT greater than 100000 AND AMTDUE greater than or equal to 20000 are selected.

Example 5

Create a logical view/file called CUSMSTS3 that will order CUSMST by CREDIT. Only records with CREDIT greater than 100000 or AMTDUE greater than or equal to 20000 are to be visible via this view.

Logical view name : CUSMSTS3 Desc of logical view : High credit or large debt customers Access path maint opt : IMMED Unique? NO Dynamic select? NO Key field details:

	Field	Description	A/L) 5/	U/A		
	CREDIT		_	_			
			_		+		
					+		
		—	—	+			
c	alact/Omi	t critoria.					
. 7	ејест/Слпп	і Спиена.					

AND/OR SELECT/OMIT Field Operation(s) SELECT____ CREDIT____ COMP(GT 100000)_____ OR SELECT____ AMTDUE___ COMP(GE 20000)_____

Note that since an entry was made in the SELECT/OMIT column for the second select/omit statement the 2 statements are read together. In this case only customers with CREDIT greater than 100000 OR . AMTDUE greater than or equal to 20000 are selected.

Example 6

Create a logical view/file called CUSMSTS4 that will order CUSMST by STATE and CUSTNO. Only records with a STATE of NSW, VIC or QLD are to be visible via this view.

Logical view name : CUSMSTS4 Desc of logical view : East coast customers by state, cust Access path maint opt : IMMED Unique? NO Dynamic select? NO Key field details Field Description A/D S/U/A

 STATE_____

 CUSTNO_____

 ______+

Select/Omit criteria:

AND	/OR	SELECT/O	OMIT	Field	Operation(s)	
	SELI	ECT	STAT	Ъ	COMP(EQ 'NSW')_	
OR	SE	LECT	_ STA	ATE	COMP(EQ 'VIC')_	
OR	SE	LECT	_ STA	ATE	COMP(EQ 'QLD')	

An identical condition can be made using the VALUES keyword: Select/Omit criteria

AND/OR SELECT/OMIT Field Operation(s) SELECT_____ STATE_ VALUES('NSW' 'VIC' 'QLD')

Example 7

Create a logical view/file called CUSMSTS5 that will order CUSMST by CUSTNO. Only records with an ACCTYP value of 2, 3 or 4 are to be visible via this view.

Logical view name : CUSMSTS5 Desc of logical view : Customers with type 2, 3 or 4 accounts Access path maint opt : IMMED Unique? NO Dynamic select? NO Key field details:

Field	Description	А	/D	S/U/A
CUST	NO		_	_
		_	_	+
		_		+
		—	—	

Select/Omit criteria:

AND/C	OR SELECT/O	OMIT Field	Operation(s)	
S	ELECT	ACCTYP_	COMP(EQ 2)	
OR	SELECT	_ ACCTYP	COMP(EQ 3)	_
OR	SELECT	_ ACCTYP	COMP(EQ 4)	

The select /omit condition could also be expressed as:

 SELECT_____ACCTYP___COMP(GE 2)_____

 AND
 ACCTYP___COMP(LE 4)_____

or:

SELECT____ ACCTYP___ VALUES(2 3 4)____

or:

SELECT____ ACCTYP___ RANGE(2 4)____

3.12.3 Select/Omit Statements and LANSA/DDS Compatibility

Note that the type of record select/omit logic used by LANSA is identical and totally compatible with that used in IBM i DDS (data description specifications). In fact the select/omit criteria specified under LANSA are directly translated into DDS select/omit statements.

The only difference is that under LANSA the full words "SELECT" or "OMIT" are used. In DDS these are indicated by entering an "S" or "O" into column 17 of the input form. Other than this difference they are absolutely identical in specification logic.

The type of checking done by LANSA to the select/omit statements entered is relatively unsophisticated. As such the possibility arises of invalid select/omit statements being accepted into the system.

If this happens the problem will manifest the first time that an attempt is made to create the logical file. The LANSA job will fail and messages on the job log will indicate that the logical file failed to create successfully. If this happens, examine all source listings named QDDSSRC produced by the job. Locate those containing errors. If the errors indicate problems with the DDS select/omit statements generated by LANSA then delete and redefine the associated logical file so that the LANSA select/omit statements are acceptable to the DDS processor.

Since the LANSA select/omit statements are totally compatible with those used in DDS it would be beneficial to read the sections in the IBM supplied manual *Data Description Specifications* that relate to select/omit processing and the select/omit keywords COMP, RANGE, VALUES and ALL.

3.13 File Level Rules/Triggers

The steps involved in defining, reviewing or changing the rules and triggers associated with the fields in a file definition are as follows:



3.13.1 Choose Fields for Which Rules And Triggers Are to Be Displayed

A screen similar to the example following will be displayed when the option to "Review or change file rules and triggers" is chosen from the file definition menu:

DC@P200801 Select Field(s) from File Definition

Select from the list below field(s) in this file for which file level field rules and triggers are to be displayed, input, changed or deleted.

Sel	Field De	scription	Type Len Dec
_	DEMNAC	Name code	A 6
_	DEMNAM	Name	A 25
_	DEMAD1	Address line 1	A 25
_	DEMAD2	Address line 2	A 25
_	DEMAD3	Address line 3	A 25
_	DEMPCD	Post code	A 4

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen displays the names of all fields in the file definition currently being worked with. Working from this display you can:

- Select one or more fields from the file definition for which the associated rules/triggers are to be displayed for review, change or addition. The resulting displays are described in the next section.
- Press ENTER/REC ADV without choosing any field. In this case the file definition menu will be re-displayed.

3.13.2 Display All Existing Rules and Triggers

A screen similar to this example will be displayed for each of the fields chosen from the preceding display. It shows all rules and triggers that currently exist for the chosen field.

Remember that this work is being performed at the FILE level. Thus these conditions apply when working with the rules and triggers:

- All existing FILE and DICTIONARY level rules and triggers for the field will be displayed.
- Any existing FILE level rules and triggers can be reviewed, changed or deleted.
- New FILE level rules and triggers can be added.
- Any existing DICTIONARY level rules and triggers can be reviewed. However they cannot be modified or deleted. To modify dictionary level rules and triggers it is necessary to work from the FIELD DEFINITION MENU which is described in Fields Rules/Triggers.

DC@P700101 Select Rule / Trigger

Usage

Sel Seq Description Type Source A/C/D **** DICTIONARY RULES ****

- _ 10 Check not blank. LIST OF VALUES DATA DICTIONARY Y Y Y
- 20 Check valid number range RANGE OF VALUES DATA DICTIONARY *** DICTIONARY TRIGGERS ***
- _ 10 Print field value TRIGGER DATA DICTIONARY

Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Cancel Fnn=Messages

Note: File name is only displayed when working at file level. Working from the Select Rule / Trigger screen you can:

- Add a new rule or trigger by using the Add/Create function key. The resulting display is described in the next section.
- Select an existing rule or trigger for review by entering any non-blank character beside the desired item in the column labeled "Sel". On the subsequent displays the rule or trigger can be reviewed in full, changed or deleted.

The method by which existing rules and triggers can be reviewed, changed or deleted, is described in detail for Fields. While the Field description applies to dictionary level rules and triggers, the displays used and input requirements are absolutely identical for File level rules and triggers.

3.13.3 Add a New Rule or Trigger

This display results if the Add/Create function key was used from the display of all existing rules and triggers.

Select Rule / Trigger DC@P700101 File : XXX (XXXX) Field :XXX : Add Rule / Trigger : (XXXX) : Use cursor to make a selection. : Usage Sel Seq Description : Range of Values Rule : A/C/D**** DICTION : List of Values Rule 10 Check not bl : File Entry Lookup Rule : TIONARY Y Y Y 20 Check valida : Evaluate Expression for Rule : TIONARY Y Y Y : Call User Program for Rule : : Date Range/Format Rule : : Add a Trigger : Cancel Add Request : Fnn=Help Fnn=Messages Fnn=Cancel: •

Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Cancel Fnn=Messages

Position the screen cursor anywhere on the same screen line as the type of item you wish to add/create. This method of selection is identical to that used to select entries from any of the LANSA menus.

Once a type of rule or a trigger has been selected the next display will ask for a detailed specification of the rule or trigger.

A detailed description of new rules and triggers is described in detail for Fields. While the Field description applies to dictionary level rules and triggers, the displays used and input requirements are absolutely identical for File level rules and triggers.

3.14 Access Routes to Other Files

An access route defines a route or path from one file to another file.

An access path answers the question "Given that I have a record from file A, how do I access the associated record(s) in file B"?

File A is always the file definition that is currently being worked upon. File B can be any other physical or logical file defined using the LANSA system.

File A can have many access routes. They could all be from file A to file B, or some could be to any other file that is related to file A.

In addition, access routes can be followed in a chain. Thus if there is an access route from file A to file B, and an access route from file B to file C, then it is possible to start with a record from file A and locate the associated record(s) in file C.

Access routes are very important to the automatic process definition components of LANSA because:

- Users do not need to understand the database structure. The user only needs to nominate a "base" or "starting" file and LANSA can then follow the access route chains to present the other accessible files as simple alternatives.
- The relationship between records in the 2 files in any access route is predefined as "1: 1" or "1: many". This enables the required screen formats to be automatically designed for the type of information expected.

3.14.1 What Is a Predetermined Join Field?

A Predetermined Join Field is used to hold the result of various operations performed on information retrieved from the file nominated by the access route. It must be defined in the Data Dictionary.

The type of operations available to be performed depend on the relationship between the files as indicated by the number of records expected on the access route definition.

The operations are TOTAL, MAXIMUM, MINIMUM, AVERAGE, COUNT, and LOOKUP.

When the relationship is 1 to 1, (number of records 1) using the key defined on the access route a single value will be retrieved into the Predetermined Join Field. For example a product description can be retrieved using the product code if an access route has been defined between an order line file and a product file. A "keep last" value can be nominated to reduce I/Os.

When the relationship is 1 to many (number of records greater than 1) the Predetermined Join Field will hold the result of the selected operation on the selected field in the file nominated by the access route, retrieved using the key in the access route definition. For example the total of the line values from an order line file could be retrieved for display with the order header file.

The access route on which Predetermined Join Fields are defined includes in its definition two attributes which affect Predetermined Join Fields. They are the keep last value and the retrieval timing attribute. Both of these attributes apply to all Predetermined Join Fields defined on the access route.

The keep last value only affects Predetermined Join Fields when the access route relationship is one to one. It causes the last "nnn" values retrieved from the accessed file to be stored in memory and thereby reducing I/Os.

The retrieval timing attribute nominates when the Predetermined Join Fields are to be retrieved from the accessed file, i.e. either before or after virtual fields which are derived 'After input from file'.

Predetermined join fields will be displayed when prompting for fields in the RDML editor in the same way as virtual fields.

If the file to be accessed is a high speed table, the high speed table will be used thus reducing I/Os.

Predetermined join fields are dependent on the access route they are defined on.

WARNING: Predetermined join fields are very efficient when used to look up descriptions in code files where the file accessed is a high speed table but overuse of this facility could cause the opposite effect, particularly in DBOPTIMISED functions when the accessed file is not a high speed table. **As a guide, do not access more than 10 to 15 files for Predetermined Join Fields.**

3.14.2 Steps to Define, Review or Change Access Routes

The steps involved in defining, reviewing or changing the access routes associated with a file definition are as follows:





3.14.3 Display Existing Access Routes

This screen is displayed when:

- Reviewing the access routes in a file definition for possible amendment or deletion.
- Defining a new access route.

DC@P200901 Access Routes

Type options, press Enter. 2=Review/change Access Route 23=Predetermined Join Fields

Access		File
Opt Route	Description	Accessed PJF's
XXXXX	XXXXX XXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXX	XXXXX XXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXX	XXXXX XXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
_ XXXXX	XXXXX XXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Cancel Fnn=Messages

Working from this screen you can:

- Select an existing access route for detailed review and/or change. To do this enter the number 2 in the column labeled "Opt" against the access route required.
- Select an existing access route for detailed review/change or to create Predetermined Join Fields. To do this, enter the number 23 in the column labeled "Opt" against the access route required.
- Create a new access route. Use the Add/Create function key to indicate that a new access route is to be added to the file definition.

Note: The column labeled "PJF's" will contain a "Y" if Predetermined Join

Fields exist on a particular access route.

3.14.4 Detailed Access Route Maintenance

This screen is displayed when:

Г

- Reviewing an existing access route in detail
- Specifying the details of a new access route

DC@P200902	Create Access Route			
File : XXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXX				
Access route name				
Access route description				
File to be accessed via this route				
Maximum number of records expected Keep Last				
PJF before/after Virtual Fld derivation Action to take if no records found				
Key fields/values use	d for access:			
Fnn=Help Fnn=Exit	Fnn=Cancel Fnn=Messages			

Note: The Change and Delete Function keys are only enabled when reviewing an existing access route.

Working from this screen you can:

• Review the full details of an existing access route.

• Change an existing access route. Use the Change function key to indicate that the access route should be altered. The screen will be re-displayed and all fields available for change. Validation of the access path will follow the same rules as the Add/Create option.

Note: A message will be displayed on the change screen if Predetermined Join Fields are defined on the access route, warning that these fields will be deleted if the file to be accessed or the relationship (number of records) is changed.

• Delete an existing access route. Use the Delete function key to indicate that the access route should be deleted. On the resulting display enter YES to confirm that the access route should be deleted. If YES is not entered the access route will not be deleted.

Note: A message will be displayed if Predetermined Join Fields are defined on the access route, warning that these fields will be deleted along with the access route.

• Specify details of a new access route that is to be created.

Input Options

These input options apply when defining a new access route in a file definition:

Access Route Name

Specifies the name that is to be assigned to the access route. Every access route must have a name. Maximum length is 10 characters. Used for identification purposes only.

Name specified must be unique within the current file definition. It is suggested that a naming convention be developed for access route names. For example first 3 letters may indicate current file definition, next 3 indicate the name of the file being accessed, etc

Access Route Description

Specifies the description of this access route. A description must be provided as it is the easiest way users have to identify what information can be obtained from the access route. For example:

Current File Definition	File Accessed Via Access Route	Access Route Description			
Order header	Order lines	Order lines associated with			

		order		
Order header	Cust master	Full details of order customer		
Cust master	Order header	Orders associated with this customer		
Order lines	Order header	Order associated with this order line		

File to be Accessed via This Route

Specifies the name of the physical or logical file that is to be accessed via this access route. Physical or logical file specified must have been already defined to the LANSA system. File name can be entered in full, partially or left blank. If required a list of available files will be displayed and the required file chosen.

Maximum Number of Records Expected

Specifies the maximum number of records that are expected to be found in the "file to be accessed" that have a key matching the "key fields/value used for access" specified. The value entered must be in the range 1 to 9999.

In the current version of LANSA the actual value entered is only significant in that it is 1 or greater than 1. If the value is 1 then a "1:1" relationship between the files is established. If the value is greater than 1 a "1:many" relationship is established. The relationship between the files affects the method by which screen formats are designed.

It is recommended that a value as "realistic" as possible be entered into this field. For instance, if the access route is from an order header to an order lines file and an order can have a maximum of 99 lines, enter a value of 99.

If the number of records expected exceeds 9999 enter a value of 9999.

Keep Last

The Keep Last value (0 to 999) applies to Predetermined Join Fields defined on the access route when the relationship is one to one. Each value retrieved from the accessed file is stored in memory up to the keep last value. If more than the keep last value are retrieved, current values are overwritten starting from the first value retrieved. This is a very useful feature when using Predetermined Join Fields to retrieve values from small frequently used code fields to reduce I/Os. It is ignored if the access route relationship is one to many.

PJF Before/After Virtual Fld Derivation

This attribute nominates when the Predetermined Join Fields defined on this access route will be retrieved, i.e. before (B) or after (A) virtual fields which are derived After input from file. It allows a Predetermined Join Field to be used in deriving a virtual field or a virtual field to be used as a key to a file accessed for Predetermined Join Fields.

Action to Take If No Records Found

Indicates what action should be taken if no records can be found in the "file to be accessed" with the "key fields/values" specified below. Valid entries in this field are:

- ABORT The function attempting to access the file specified should abort (fail) with an error message indicating the cause of the problem. This option can be used to continually verify database integrity. For instance an access route from an order lines file to an order header file should always find a record. An order line without an associated order header probably indicates database corruption.
- IGNORE The function attempting to access the file specified should ignore the no records situation and continue to process. This option may be valid in the reverse access path of the case above. It is perfectly valid for an order header record to have no associated order lines.
- DUMMY The function attempting to access the file specified should create a "dummy" record when no "real" record(s) can be found. The dummy record created will have blanks in all alphanumeric fields and zero (0) in all numeric fields. Only one dummy record will be created.
- N/AVAIL The function attempting to access the file specified should create a "dummy" record when no "real" record(s) can be found. The dummy record created will have zero (0) in all numeric fields, blanks in all alphanumeric fields less than 3 characters long, and as much of the string 'N/AVAIL' as will fit in all other alphanumeric fields. Only one "dummy" record will be created. This option is useful in situations where the no record(s) situation arises occasionally. For instance an access route from an archived invoices file to a customer master file may use this option. When

the name of a customer associated with an archived invoice cannot be found (presumably because it has been deleted) then it will be displayed as 'N/AVAIL', rather than causing an error.

Key Fields/Values Used for Access

Specifies the fields or values that should be used to form the key that will be used to access the record(s) in the "file to access". If field names are used then they must be defined in this file (i.e. the file definition that the access route is being added to - not the "file to access").

At least one key field or value is required. Up to 20 key fields or values can be specified. Use the ROLL UP key to enter more.

Entries made can be:

- An alphanumeric literal (in quotes) such as 'NSW', 'BALMAIN'
- A numeric literal such as 1, 14.23, -1.141217.
- Another field name such as CUSTNO, INVNUM, etc. As mentioned the field must be defined in the current file definition. (A "?" will display the single field selection screen. The list of fields will contain all real and virtual fields defined in the current file. Refer to section 3.10.1 Select Fields When Working from File Definition Menu for more details.)
- A system variable name such as *BLANKS, *ZERO, *DATE or any other system variable defined at your installation.

Key values are checked for type and length compatibility. The entire key list supplied is checked for compatibility with the actual key(s) of the "file to access". The key list specified can be a full or partial key to the file.

In cases where a "1:many" relationship is being defined the key list specified would almost always be partial key to the file.

A warning is issued if a partial key list is specified. In 1:many cases this is normal and should be ignored. In 1:1 relationship it may be valid to only supply a partial key. However, if the warning message is issued check what has just been defined.

Note: The access route facility is provided as an aid to user traversal of database structures. It does not restrict database access to predefined access routes. The RDML component of LANSA makes use of Predetermined Join Fields which are defined on access routes, in the same manner as it does virtual fields.

3.14.5 Display Fields in Accessed File

The Select Fields from Source File screen is displayed as:

- The first step to reviewing Predetermined Join Fields
- The first step to changing Predetermined Join Fields

DC@P200903 Select Fields from Source File

Select Source field to define Predetermined Join Fields

Sel	Field	Descript	tion		Type Le	en Dec			
_	XXXXX	XXXXX	XXXXX	XXXXX	XXXXXX	XXXXX	XXXX	XXXX	XXXX
_	XXXXX	XXXXX	XXXXX	XXXXX	XXXXXXX	XXXXX	XXXX	XXXX	XXXX
_	XXXXX	XXXXX	XXXXX	XXXXX	XXXXXXX	XXXXX	XXXX	XXXX	XXXX
_	XXXXX	XXXXX	XXXXX	XXXXX	XXXXXXX	XXXXX	XXXX	(XX)	XXXX

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Working from this screen you can:

Select a field from the accessed file for review/change/create or deletion of Predetermined Join Fields. To do this enter any non-blank character in the column labeled "Sel" against the required field. The resulting display will depend on the relationship defined on the access route and the type of field selected.

- If the relationship is 1 to 1, only the LOOK UP operation is performed.
- If the relationship is 1 to many and the selected field is alpha only, the COUNT operation can be performed.
- If the relationship is 1 to many and the field selected is numeric, TOTAL, MAXIMUM, MINIMUM, AVERAGE and COUNT can be performed on it.

3.14.6 Detailed Predetermined Join Field Review

This window is displayed when:

- The relationship is 1 to 1
- Reviewing Predetermined Join Fields
- Changing Predetermined Join Fields
- Creating Predetermined Join Fields
- Deleting Predetermined Join Fields

DC@P200904 Select Fields from Source File Select Source field to define Predetermined Join Fields Sel Field **D**: **Nominate Join Fields** :en Dec XXXXXXXXXX X: :99 9 XXXXXXXXXX X: Look Up... XXXXXXXXXX :999Keep Last. 999 XXXXXXXXXX X: :99 9 XXXXXXXXXX X: :99 9 Х : Fnn=Help Fnn=Exit Fnn=Cancel : Fnn=Messages Fnn=Change : Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This window is displayed when:

• The relationship is 1 to many
- Reviewing Predetermined Join Fields
- Changing Predetermined Join Fields
- Creating Predetermined Join Fields
- Deleting Predetermined Join Fields

DC@P200905	Select Fields from Sou	rce File	
Access Route: XXX Source File: XXXX	XXXXXXXX XXXXX XXXXXXX XXXXXX	xxxxxxxxxxxx xxxxxxxxxxxx	XXXXXXXX XXXXXXXX
Select Source field	d to define Predetermin	ed Join Fields	
Sel Field D:	Nominate Join Fie	lds :en Dec	
XXXXXXXXX	XX X	•99 9	
	XX X: Total XX	XXXXXXXX	:99 9
XXXXXXXX	XX X: Maximum	XXXXXXXXXXX	:99 9
X XXXXXXXX	XXX X: Minimum	XXXXXXXXXXX	:99 9
<u> </u>	erage XXXXXXXXX	X :	
: Cou	Int XXXXXXXXXX	X :	
:	:		
: Fnn=	=Help Fnn=Exit Fnn=(Cancel :	
: Fnn	=Messages Fnn=Chang	ge :	
Fnn=Help Fnn=Ex	tit Fnn=Cancel Fnn=M	essages	

Working from Nominate Join Fields window you can:

- Review the Predetermined Join Fields defined to hold the result of the operations displayed when they have been performed on the selected field in the accessed file.
- Change the Predetermined Join Fields defined to hold the result of the operations displayed when they have been performed on the selected field in the accessed file. Use the Change function key to indicate that the

Predetermined Join Fields are to be changed. The window will be redisplayed in change mode.

- Create Predetermined Join Fields. Use the Change function key to indicate new fields are to be added. The window will be displayed in change mode and the new fields can be entered.
- Delete Predetermined Join Fields. Use the Change function key to indicate new fields are to be deleted. The window will be displayed in change mode and the fields to be deleted can be cleared (deleted) from the screen fields.

Input Options

The input options depend on the access route relationship.

If any field is left blank it is assumed that the operation is not required. If a partial name or "?" is entered a single field selection screen is displayed. The selection list will be built from the fields in the LANSA data dictionary. Refer to section 3.10.1 Select Fields When Working from File Definition Menu for more details.

The Predetermined Join Field may have the same name as the source field as long as it complies with following rules:

- It must exist in the data dictionary.
- It must not already exist as a Predetermined join, Virtual or Real field on the file.
- It must be of the type specified by the operation and type of the source field as listed below.

No check is made on the Predetermined Join Field's suitability to hold the required information. For example the field in an accessed file which holds an order line value could be used to hold the total of that field but may not be large enough. The developer must decide this type of suitability.

If the relationship is 1 to 1:

- Look up: Specifies the Predetermined Join Field that is to hold the value retrieved from the selected field in the accessed file. It will be validated as follows: It must be the same type as the field selected from the accessed file.
- Keep Last: Specifies how many occurrences of the field being retrieved are to be stored in memory to reduce I/Os. This value is an attribute of the access route and can only be changed on the access route maintenance screens.

If the relationship is 1 to many and the selected field in the accessed file is a

numeric field:

- Total: Specifies the Predetermined Join Field to hold the total of the selected field retrieved using the access route key. It must be numeric.
- Maximum: Specifies the Predetermined Join Field to hold the highest value of the selected field in records retrieved using the access route key. It must be numeric.

See end of this section for Prompt details.

• Minimum: Specifies the Predetermined Join Field to hold the lowest value of the selected field in records retrieved using the access route key. It must be numeric.

See end of this section for Prompt details.

• Average: Specifies the Predetermined Join Field to hold the average value of the selected field in records retrieved using the access route key. It must be numeric.

See end of this section for Prompt details.

If the relationship is 1 to many and the selected field in the accessed file is a numeric or an alpha field:

• Count: Specifies the Predetermined Join Field to hold the count of the number of records retrieved using the access route key. It will be validated as follows; It must be numeric.

If a partial name or '?' is entered a single field selection screen is displayed. The selection list will be built from the fields in the LANSA data dictionary. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.

Access Route Examples

These examples apply to defining an access route within a file definition. Four related file definitions are used in these examples. These are defined as follows:

File:		ORDHDR	ORDLIN	CUSMST	PROMST
Desc:		Order header	Order lines	Customer master	Product master
Fields:		ORDNUM CUSTNO DATE	ORDNUM ORDLIN QUANTY PRODNO	CUSTNO NAME ADDR1 ADDR2 ADDR3	PRODNO DESC PRICE TAXRTE
Primary	1	ORDNUM	ORDNUM	CUSTNO	PRODNO
Key	2		ORDLIN		
Logical Views:		ORDHDRV10	RDLINV1		
by key	1	CUSTNO	PRODNO		

To allow complete traversal of this database by users the following access routes would be defined:

File Definition Name	Access Route Name	File to be Accessed via this Route	Max records expected	Action to be taken if no records found	Key fields/valu used for access
ORDHDR	ORDHRT1	ORDLIN	99	IGNORE	ORDNUM
	ORDHRT2	CUSMST	1	ABORT	CUSTNO
ORDLIN	ORDLRT1	ORDHDR	1	ABORT	ORDNUM

	ORDLRT2	PROMST	1	ABORT	PRODNO
CUSMST	CUSMRT1	ORDHDRV1	9999	IGNORE	CUSTNO
PROMST	PROMRT1	ORDLINV1	9999	IGNORE	PRODNO

From this table it is easy to see how LANSA can follow the access route chains. For example, say the user nominated PROMST (product master) as a "starting" or "base" file. This would occur:

- Using access route PROMRT1 a route to ORDLIN (order lines) is available (via logical file ORDLINV1) which will access all invoice lines that use the product.
- From ORDLIN the associated order header (ORDHDR) details can be picked up using access route ORDLRT1. The access route ORDLRT2 back to PROMST would be ignored as it forms a "loop".
- From ORDHDR the associated customer details (CUSMST) can be found via access route ORDHRT2. Again ORDHRT1 would be ignored as it forms a "loop".

Predetermined Join Field Examples

Example 1

To retrieve a customer description to be displayed when processing an order:

- First an access route is defined from the order to the customer file. The maximum records retrieved is defined as 1 and the customer code field from the order file as the key.
- Next option 23 is taken against the access route and the fields in the customer file are displayed. The customer description field is selected and in the window displayed after using F21 the Predetermined Join Field to receive the description can be nominated against the LOOKUP operation. The source field name can be used and if required a KEEP LAST value entered.

When the I/O module for the order file is made operational the Predetermined Join Field customer description will be available in the same way as a virtual field when coding RDML without having to include any type of fetch from the customer file.

Where a code file is repeatedly accessed for the same information an appropriate KEEP LAST value will reduce the number of I/Os performed.

Example 2

To retrieve the descriptions of the 'from' and 'to' warehouses in a stock transfer file:

- 2 access routes must be defined, as in example 1, from the transfer file to the warehouse file. The first will use the 'from' warehouse code field as the key and the second will use the 'to' warehouse code field as key.
- When defining the Predetermined Join Field on the first access route the source field name can be used but when defining the Predetermined Join Field on the second access route a different field name must be used so that both the 'from' and 'to' descriptions are available at the same time.

Example 3

To retrieve the total value and number of lines in an order when displaying the order header details:

• Define an access route from the order header file to the order line file with the appropriate key. The number of records expected will be the maximum number of lines in an order.

- Select the access route for Predetermined Join Field definition.
- From the displayed fields in the order line file select the line value field. In the displayed window enter the Predetermined Join Field name against the TOTAL operation. The count of order lines can also be performed on this field or on any other field in the file.

When the order file is made operational the total value and line count fields are available in the RDML editor along with the order fields without coding any reference to the order line file.

Example 4

To calculate a price for each line of an order using a Predetermined Join Field and a virtual field:

- Define an access route from the order line file to the product file with a one to one relationship (maximum expected records 1), a keep last value if appropriate, set "PJF before/after virtual fields derived" to before (B) and use the product code field from the order line file as the key.
- Define a virtual field of extended definition type "mathematical calculation" to be derived "after input from file". Define the calculation as follows: product quantity (from order line file) multiplied by product price (Predetermined Join Field from product file) equals virtual field.

By including the product price (Predetermined Join Field) and product value (virtual field) when defining a select command in RDML, the arithmetic will be performed without any additional RDML code.

Comments/Warnings

Although the Predetermined Join Field processing is coded within the I/O module the source files are only accessed when one of the Predetermined Join Fields on a particular access route is used in the function calling the I/O module.

This means that, unless the Predetermined Join Field is explicitly specified in the I/O operation (for example FETCH or SELECT), it would not be retrieved and calculated from the source file (not even when it is derived from a Virtual Field).

If the function has been DBOPTIMIZED the Predetermined Join Field processing is included in the I/O module logic which is embedded in the function.

If the file to be accessed by the access route is defined as a high speed table the source information for the Predetermined Join Fields will be retrieved from the

high speed table thereby reducing I/Os.

Warning: Read the appropriate sections on high speed tables before attempting to use them.

When defining Predetermined Join Fields careful thought should be given to their use. They are very efficient for looking up code files for descriptions when used with a KEEP LAST value or accessing a high speed table, but if they are used excessively each file accessed will affect the number of files used in the I/O module and DBOPTIMIZSED functions (see *Using *DBOPTIMIZE / *DBOPTIMIZE_Batch* for a further explanation). Also they could greatly increase the number of I/Os performed by the I/O module where there are 1 to many relationships.

As a guide, do not access more than 10 to 15 files.

3.15 Virtual Fields

LANSA file definitions can contain 3 types of fields:

• Traditional or "real" fields

These are the normal fields found in any IBM i file definition and are assembled together to form the record format of the associated physical file. Such fields actually exist in the database file and their content (or value) can be extracted from any record in the file at any time.

• "Virtual" fields

These are fields that do not actually exist in the IBM i database file, but are dynamically derived from "real" fields in the file.

• "Predetermined join" fields

For information, refer to 3.14.1 What Is a Predetermined Join Field? in Access *Routes to Other Files*.

The distinction between "real" and "virtual" fields can be best illustrated by example.

Consider an accounting file called ACCMST that contains an 8 digit field called CHTACC (chart of accounts) that looks like this:

0 8 6 3 4 3 1 2

This example is for chart of accounts number 08634312. In actual fact the chart of accounts has 3 components. These are: a 2 digit company number (08), a 2 digit division number (63), and a 4 digit cost center (4312).

Since the company number, division number and cost centre can be "derived" directly from the chart of accounts number it would be possible to define in file ACCMST three "virtual" fields called COMPNO, DIVNUM and COSTCT. If this was done the definition of file ACCMST would contain:

Field

Alphanumeric Length

A "real" field called CHTACC 8

A "virtual" field called COMPNO 2

A "virtual" field called DIVNUM 2

A "virtual" field called COSTCT 4

Of course there is a missing link. That is the definition of how virtual fields COMPNO, DIVNUM and COSTCT are "derived" from the real field CHTACC.

There are two methods available that allow you to specify how a virtual field is derived.

The first method is the "Extended definition of virtual fields" function, which allows you to perform the most commonly used actions to derive virtual fields from real fields and vice versa, these actions are as follows:

- Substringing
- Concatenation
- Mathematical calculation
- Date conversion

The "Extended definition" function of virtual fields is described elsewhere in this guide.

The second method in specifying how virtual fields are derived from real fields and vice versa is achieved by entering RPG/400 program code. This method is the simplest and by far the most flexible method of specification for more complex actions such as:

- Type and length conversions
- Date to age conversion
- Dynamic data conversion

In this example the virtual fields COMPNO, DIVNUM and COSTCT could have been derived by using the "Extended definition of a virtual field" function by Substringing the real field into CHTACC OR by entering a few lines of RPG/400 code that mapped field CHTACC into fields COMPNO, DIVNUM and COSTCT.

Once this has been done COMPNO, DIVNUM and COSTCT would appear to be in file ACCMST. However, if the file was examined it would be found that the fields do not actually exist. Hence the name "virtual" fields.

Note that fields COMPNO, DIVNUM and COSTCT are only used when reading (i.e. inputting) from the ACCMST file. When writing to or updating the ACCMST file they have no particular meaning. This need not be so, in fact virtual fields COMPNO, DIVNUM and COSTCT could be used to "re-assemble" field CHTACC when outputting to the file. This feature of virtual fields is extremely useful in some situations. Refer to Virtual Fields for more details and examples.

3.15.1 Examples of Virtual Field Applications

The introduction to this section described a very simple example of 3 "virtual" fields that were directly derived from a "real" field by simple mapping.

However, because the virtual field derivation logic is specified to LANSA by RPG program code it is possible to use much more complex derivation logic.

These examples demonstrate some of the capabilities of the virtual field facility and common examples of its use. Hopefully it can also be seen that the capabilities are only really limited by the imagination of the user:

Substringing and concatenation

The example in the preceding section that involved breaking a field down into 3 virtual fields is an example of substringing. Virtual fields can be used in all forms of substringing. This includes substringing 1 real field into multiple virtual fields and substringing multiple real fields into one virtual field (also called concatenation).

Access to files with no external description

Many older S/38 applications, and S/36 migrated IBM i applications, use files that have no "external" field descriptions. These are also referred to as "internally described" files. These files appear as if they only contain one very large field which is in fact the entire file record, rather than a series of fields.

In such cases, all the fields in the file can be defined as virtual fields and then mapped from the record. This is in fact another example of substringing one real field into multiple virtual fields.

This case would also involve the mapping of virtual fields to the real field before output to the file. Refer to examples 1 and 2 in Examples of Virtual Fields & Derivation Code for more details.

Simplification and standardisation of common data manipulations

In some applications the manipulation of data from a file in a particular way is very common and is repeated many times in differing applications.

For instance, consider an inventory file. The rule to determine whether a product requires re-ordering may be something like ("quantity on hand" + "quantity awaiting return" - "quantity on order") < "re-order level". This rule may be used in many different applications, particularly in the inquiry and reporting areas.

The logic in this rule can be simplified and standardized by using a virtual field.

If a virtual field called REORDER was set up in the file definition, then derivation code could be written to evaluate the rule and set REORDER to "YES" or "NO".

This approach has several advantages:

- Simplification. To users of the file there appears to be a field called REORDER which can be very simply tested as "YES" or "NO".
- Standardisation. The rule to determine the re-order flag is standardized. There is no chance of slightly different rules being used in different applications.
- Centralization. The rule is held in one place only, not repeated in many different applications. If the rule changes it need only be changed in one place.

Type and length conversions

When the type (i.e. alpha, packed or signed) or length of a field is inappropriate or inconvenient for common use requirements a virtual field can be established.

For instance a 15 digit numeric field that never contains more than 999 can annoy users in reporting environments because it wastes 12 spaces on the report. A virtual field containing only 3 digits can be set up for use on reports.

Likewise an alphanumeric date may annoy users because it cannot be printed with an edit code/word. A numeric virtual field could be set up to solve this problem.

Aggregation and accumulation

Many database files contain "arrays". This most commonly takes the form of a series of fields like SAL01, SAL02, SAL03 SAL12 representing company sales for each of the months of the year.

Virtual fields can be used to aid users when working with these type of structures, particularly in the reporting and inquiry environments.

For instance virtual field SALYR could be defined to contain the total of fields SAL01 -> SAL12.

Similarly virtual fields SALQ1 (containing total of SAL01 -> SAL03), SALQ2 (containing total of SAL04 -> SAL06), etc, could be defined to contain quarterly sales totals.

Date conversion

When a date is held in a real field in format YYMMDD it can be easily mapped into a virtual field in format DDMMYY. The real field is the most commonly

used format for ordering the file, but the virtual field format is the most commonly used for printing.

Date to age conversion

A date (possibly of date of birth) field in a file can be converted into two virtual fields containing "age in years" and "age in months". Note that derivation logic here would actually involve the date field in the file and the current date (which is why the "age" cannot be stored in the database file - it will be wrong tomorrow).

Note that the logic involved here is more complex. In fact many sites would have a subroutine to do the job. Since derivation logic is specified to LANSA as RPG code it is easy to call existing subroutines.

Dynamic data conversion

Just about any form of dynamic data conversion can be achieved by using virtual fields.

To take an example to the extreme, imagine that a new company requirement for printing customer names is that:

- They are always in uppercase
- All quotes, commas, full stops and hyphens are removed
- The name must be reversed (i.e. printed backwards)

If the customer name field was called CUSNAM then it would be a simple task to define a virtual field in the file called PRTNAM that matched these requirements.

3.15.2 Steps to Define, Review or Change Virtual Fields

The steps involved in defining, reviewing or changing virtual fields in a file definition are as follows:



4. Derive Derive virtual Derive virtual Derive virtual virtual field field by a field by a field by a by substringing concatenation mathematical date conversion a field/record of fields calculation of a field

3.15.3 Virtual Fields Maintenance

This screen is displayed when:

- Reviewing the virtual fields in a file definition for possible change.
- Defining the virtual fields in a new file definition.

DC@P201601 MMMMMMM		
File : XXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXX		
Edit virtual field derivation code YES, NO		
Review/Change		
Extended		
Sequence Name Description Definition PJF		
PJF's before virtuals derived on input		
99999 XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXX		
Virtual Fields derived after input		
99999 XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXX		
PJF's after virtuals derived after input		
99999 XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXX		
Virtual Fields derived before output		
99999 XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXX		
Undefined Virtual Fields		
99999 XXXXXXXX XXXXXXXXXXXXXXXXXXXXXXX		
Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Cancel Fnn=Change Fnn=Delete		
Fnn=Messages		

Note: MMMMMMM on line 2 indicates mode as Display, Add, Change or Delete.

Working from the Virtual Fields in File Definition screen you can:

• Review the virtual fields already in the file definition.

- Specify details of new virtual fields to be included into the file definition. Use the Add/Create function key to indicate that new fields are to be added to the file definition, then specify details of the new fields.
- Change the sequence of virtual fields already in the file definition. To do this use the Change function key and make all desired changes.
- Delete virtual fields from the file definition. To do this use the Delete function key. The resulting display will request that the virtual field(s) to be deleted should be selected. A final display will then request confirmation that the selected virtual field(s) be deleted. The delete request must be confirmed by entering YES. If YES is not entered no virtual fields will be deleted from the file definition.
- Choose to review/change the extended definition of a virtual field. To do this use the Change function key to enable this option and then select the virtual fields you want to work with.
- Choose to review or edit the RPG/400 source code that is used to derive the virtual fields. To do this enter YES in the request field. The resulting display is described in more detail in the following sections.

Remember that virtual field names can be specified in full, partially or chosen from the data dictionary (by entering a "?").

Note: On this screen virtual fields and Predetermined Join Fields are grouped according to when they are retrieved or derived with appropriate headings. It is important to understand this order as you may wish to use Predetermined Join Fields to derive virtual fields or to use virtual fields as keys on access routes to retrieve Predetermined Join Fields. If a virtual field is defined as derived 'after input from file' and 'before output to file', it will only appear once in the list in the 'after input from file' group. The Predetermined Join Fields cannot be added, changed, or deleted from this screen. They can only be maintained via the access route maintenance screens described in the section in this manual on access routes and Predetermined Join Fields.

Input Options

These input options apply when specifying the virtual fields in a file definition:

Edit Virtual Field Derivation Code?

This field is used to indicate that the RPG/400 source code used to derive the virtual fields is to be reviewed or edited.

Enter YES to edit the RPG/400 source code. The resulting display is described in detail in the following sections.

Leave blank or enter NO if you do not wish to review or edit the RPG/400 source code.

Seq

Specifies the sequence/order of the virtual field relative to other virtual fields in the file. Use numbers in the range 1 to 99999 to indicate the order that fields should be included into the file definition.

Virtual fields and Predetermined Join Fields are grouped according to when they are retrieved or derived. This is flagged on the virtual field definition and on the access route on which Predetermined Join Fields are defined. The sequence number for virtual fields only applies within the group they are listed. To move virtual fields or Predetermined Join Fields from one group to another, the flags on the virtual field definition or access route definition must be changed.

Note: If a virtual field is flagged as derived both 'after input' and 'before output', it only appears once in the 'after input' group.

When no sequence number is specified, sequence 99999 is assumed. When fields are assigned the same sequence number they are sub- ordered by the order in which they are specified on the screen.

Field Name

Specifies the name of a field that is to be included in the file definition as a virtual field. Any field nominated here must have been already been defined in the LANSA data dictionary. By entering a partial name, '?' or sequence number with a blank field name the multiple field selection screen is displayed. The selection list will be built from the fields in the LANSA data dictionary. A maximum of 30 fields can be selected at a time. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.

Review/Change Extended Definition

Provides the option to elect whether the extended definition of a virtual field can be reviewed or changed. This option is only available when "virtual" fields have been selected for change or creation.

If a value has been entered in the extended definition field against a virtual" field and an extended definition type has already been specified for the virtual field then the relevant screen will be displayed, these screens are described in more detail later on in this section.

If no extended definition types exist for the field, the user will be prompted with

an "Action Box" from which one of these options can be selected to generate "virtual" code automatically:

- Substringing
- Concatenation
- Mathematical calculations
- Date conversion

The Action Box that will provide these options is as follows:



If an option has been selected then the relevant screen will be displayed, these screens are described in more detail later in this section.

Note: The mathematical calculation option will only be available if the selected virtual field is numeric.

In addition, any field nominated as a virtual field must not have been defined as a "real" field in this file definition. Refer to elsewhere for details of how "real" fields are defined in a file definition.

PJF

This display-only column indicates that the field is a Predetermined Join Field retrieved before (B) virtual fields which are derived after input from file, or after

(A) virtual fields which are derived after input from file, or is a virtual field which is undefined (U).

Virtual Field Extended Definitions

The Virtual field extended definition function allows you to derive virtual fields from real fields and vice versa by mapping data using the Substringing, Concatenation and Date conversion options. It also allows you to perform mathematical calculations to derive a resulting virtual field.

This function dramatically reduces the requirement and dependence of RPG/400 programming skills for specifying RPG/400 code to derive virtual fields, the four extended definition options available - substringing, concatenation, mathematical calculation and date conversion are the most common cases used to derive virtual fields and should be used wherever possible to derive the resulting virtual field. It is recommended that you only use RPG/400 code to derive virtual fields where the already defined options cannot produce the desired result.

Each one of the virtual field extended definition options is described in more detail in:

- 3.15.4 Substringing of a Field/Record
- 3.15.5 Concatenation of Field(s)
- 3.15.6 Mathematical Calculation of a Field
- 3.15.7 Date Conversion of a Field.

3.15.4 Substringing of a Field/Record

This screen is displayed when the option to perform a Substring for a virtual field has been chosen from the Virtual field extended definition action box.

DC@P201605	Substringing of a Field/Record
File XXXXXXXX	<pre><x td="" xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx<=""></x></pre>
Virtual field XXXX	xxxxxx xxxxxxxxxxxxxxxxxxxxxx
Derive "virtual" field Setup "real" field be	l on input from file : YES, NO fore output to file : YES, NO
Field to substring fro)m
Start position	·····

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

Working from the Substringing of a Field/Record screen you can:

- **Review** the substring extended definition of the virtual field currently being worked with.
- **Specify** the substring extended definition of the virtual field currently being worked with.
- **Change** the substring extended definition of the virtual field currently being worked with.
- **Delete** the substring extended definition of the virtual field currently being worked with. There is no confirm option when delete has been specified for a virtual field extended definition.

Input Options

These input options apply when specifying the substringing extended definition

of the virtual field currently being worked with:

File

Is the name of the physical file currently being worked with.

Virtual Field

Is the name of the "virtual" field within the physical file currently being worked with. The value of the "virtual" field will be the result of the substringing of the field/*RECORD. The "virtual" field must be large enough to contain the result of the substring otherwise unpredictable results could occur.

Special notes for substringing an alpha field into a numeric field:

Note 1: The field should only contain the digits 0-9 and optionally the decimal point. Any other character, including a sign character ('+' or '-'), will give unpredictable results.

Note 2: Substringing is from left to right, therefore if a field containing '123.45' is substringed into position 1 of a signed (6,2) field (which is initially set to *ZERO), the value will be set to 1234.50.

Derive "Virtual" Field on Input from File

Specifies whether the "virtual" field should be derived from the "real" field ("real" field can be either a "real" or "virtual" field name within this file) on input. Valid entries are "YES" or "NO".

Setup "Real" Field Before Output to File

Specifies whether the "real" field ("real" field can be either a "real" or "virtual" field name within this file) should be setup from the "virtual" field before output. Valid entries are "YES" or "NO".

Field to Substring from

Specifies whether the "virtual" field will be generated from a field ("real" or "virtual") in the physical file or the record contents of the physical file (this is indicated by (*RECORD)).

Warning: If "*RECORD" has been specified for this field the utmost care should be taken in the substringing of the file record. There will be no validation checks to ensure data type/length compatibility. Virtual fields formed as a result of a substring operation specifying "*RECORD" are totally your responsibility.

PLEASE be careful when performing this option when using "*RECORD" as field definition errors could cause unpredictable results.

By entering a partial name or "?", the single field selection screen will be displayed. The selection list will be built from the real and virtual fields in the current file's definition. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.

Start Position

Specifies the start position within the field ("real" or "virtual") or the physical file record that the "virtual" field is to be generated from. There is no requirement for an "End position" as this is automatically calculated by adding the length of the virtual field to the start position.

Note: There is one restriction when specifying the substringing extended definition for a virtual field and that is you cannot perform a substringing operation from a packed numeric field to an alpha field.

Substring Example

This screen demonstrates the use of the "Substring of a field/record" option and how it may be used in breaking down a field/record after input from a file.

Using the example of the accounting file ACCMST and the chart of accounts field CHTACC described in the last section, the "virtual" field COMPNO can be derived from the "real" field CHTACC as follows:

DC@P201605 Substringing of a Field/Record

File: ACCMST Accounts Master file

Virtual field: COMPNO Company number (first 2 digits of CHTACC)

Derive "virtual" field on input from file . . : YES YES, NO Setup "real" field before output to file . . : NO YES, NO

Field to substring from CHTACC

Start position1

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

A similar substringing operation could be performed on CHTACC to derive virtual fields DIVNUM and COSTCT simply by specifying a start position of 3 for DIVNUM and 5 for COSTCT.

3.15.5 Concatenation of Field(s)

This screen is displayed when the option to perform a Concatenation for a virtual field has been chosen from the Virtual field extended definition action box.

DC@P201606 Concatenation of field(s)

Derive "virtual" field on input from file . . . _ YES, NO Setup "real" field before output to file . . . _ YES, NO

Using fields

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

Working from the Concatenation of field(s) screen you can:

- **Review** the concatenation extended definition of the virtual field currently being worked with.
- **Specify** the concatenation extended definition of the virtual field currently being worked with.
- **Change** the concatenation extended definition of the virtual field currently being worked with.
- **Delete** the concatenation extended definition of the virtual field currently being worked with. There is no confirm option when delete has been

specified for a virtual field extended definition.

Input Options

File Is the name of the physical file currently being worked with.

Virtual Field

Is the name of the "virtual" field within the physical file currently being worked with. The value of the "virtual" field will be the result of the concatenation of the fields specified in the "Using fields" list. The "virtual" field must be large enough to contain the result of the concatenation otherwise unpredictable results could occur.

Derive "Virtual" Field on Input From File

Specifies whether the "virtual" field should be derived from the file on input. Valid entries are "YES" or "NO".

Setup "Real" Field Before Output to File

Specifies whether the "real" field ("real" field can be either a "real" or "virtual" field name within this file) should be setup from the "virtual" field before output. Valid entries are "YES" or "NO".

Using Fields

Each field ("real" or "virtual" name within this file) specified will be joined together to form the "virtual" field. The accumulated length of all the specified fields must not exceed the total length of the "virtual" field. When the fields have been concatenated they will form the "virtual" field starting at position 1.

By entering a partial name or '?' the multiple field selection screen will be displayed. The selection list will be built from the real and virtual fields in the current file's definition. A maximum of 50 fields can be selected. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.

Concatenation Field Examples

This screen demonstrates the use of the "Concatenation of field(s)" option and how it may be used in joining together a number of fields (real or virtual) to form a virtual field after input from a file.

Using the idea of the chart of accounts example we will switch the real and the virtual fields around so we now have 3 "real" fields COMPNO (alpha length 2), DIVNUM (alpha length 2) and COSTCT (alpha length 4) and join them together to form the "virtual" field CHTACC (alpha length 8):

DC@P201606 Concatenation of field(s)

File: ACCMST Accounts Master file

Virtual field: CHTACC Chart of accounts number

Derive "virtual" field on input from file . . . YES YES, NO Setup "real" field before output to file . . . <u>NO</u> YES, NO

Using fields
COMPNO
DIVNUM
COSTCT
+

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

3.15.6 Mathematical Calculation of a Field

This screen is displayed when the option to perform a Mathematical calculation for a virtual field has been chosen from the Virtual field extended definition action box.

DC@P201607 Ma	athematical calculation of a	field
File : XXXXXXXXXX	x xxxxxxxxxxxxxx	xxxxxxxxxxxx
Virtual field : XXXXX	xxxxxx xxxxxxxxx	****
Derive "virtual" field o	on input from file Y	ES, NO
CALCULATIC Value 1 Operation	DN SPECIFICATI n Value 2 Result	O N
		-
Fnn=Help Fnn=Exit F	Fnn=Cancel Fnn=Messages	Fnn=Change Fnn=Delet

Working from the Mathematical calculation of a field screen you can:

- **Review** the mathematical calculation extended definition of the virtual field currently being worked with.
- **Specify** the mathematical calculation extended definition of the virtual field currently being worked with.
- **Change** the mathematical calculation extended definition of the virtual field currently being worked with.
- **Delete** the mathematical calculation extended definition of the virtual field

currently being worked with. There is no confirm option when delete has been specified for a virtual field extended definition.

Input Options

File

Is the name of the physical file currently being worked with.

Virtual Field

Is the name of the "virtual" field within the physical file currently being worked with. The value of the "virtual" field will be the result of the mathematical calculation of the fields specified in the "calculation specification" list. The "virtual" field must be a numeric data type, either packed or signed. The "virtual" field must be large enough to contain the result of the mathematical calculation otherwise unpredictable results could occur.

Derive "Virtual" Field on Input From File

Specifies whether the "virtual" field should be derived from the file on input. Valid entries are "YES" or "NO".

Value 1

Specifies the name of a field ("real" or "virtual" field name within this file), a valid numeric literal or an *WORKnnnnn field. This is an optional entry for all operation codes, except for when the operation code of "S" is specified, in this case the "Value 1" field must be blank. If a field name has been specified its data type must be either packed or signed.

By entering a '?' the single field selection screen is displayed. The selection list will be built from all the real and virtual fields in the current file's definition. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.

Operation

Specifies the operation to be performed for the calculation line. Valid entries are:

- "+" Add
- "-" Subtract
- "/" Divide
- "*" Multiply

"S" Set

Note: If the operation for a calculation line is '/' (Divide) a check will be automatically performed to ensure "Divide by zero" errors are prevented. If the value within "Value 2" is 0 the "Result" value will be automatically set to 0, if the value within "Value 2" is not 0 then the '/' (Divide) operation will be performed.

Value 2

Specifies the name of a field ("real" or "virtual" field name within this file), a valid numeric literal or an *WORKnnnnn field. This is a mandatory entry for all operation codes, except for when the operation code of "S" is specified, in this case the "Value 2" is optional. If the operation code is "S" and "Value 2" is blank the system will automatically set "Value 2" to 0. If a field name has been specified its data type must be either packed or signed.

By entering a '?' the single field selection screen is displayed. The selection list will be built from all the real and virtual fields in the current file's definition. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.

Result

Specifies the name of the "virtual" field or a *WORKnnnnn field. This is a mandatory entry.

Note:

- The *WORKnnnnn fields are exclusively reserved for use in "virtual" code extended definition "mathematical calculations" only.
- The *WORKnnnnn fields do not have to be defined within LANSA, the *WORKnnnnn field length will be automatically assumed to be 30,9.
- All *WORKnnnnn fields that are used in a "mathematical calculation" are automatically initialised before the first time they are used.
- Multiple *WORKnnnnn fields can be used in "mathematical calculations" simply by replacing the "nnnnn" portion of the work field name with a unique replacement value, for example: *WORK00001, *WORK00002, *WORKTOT01 etc.
- If *WORKnnnnn fields with the same name are used in other virtual field "mathematical calculations" their values will not be carried forward to the next virtual field's mathematical calculation.

Mathematical Examples

This screen demonstrates the use of the "Mathematical calculation of a field" option and how it may be used to produce a "virtual" field's value by performing a mathematical calculation.

Calculate the Gross sales price of a transaction by multiplying the Net sales price by the Current sales tax rate and adding the result to the Gross sales price:

DC@P201607 Mathematical calculation of a field

File: SALTRN Sales Transaction file

Virtual field: GRSPRC Gross sales price

Derive "virtual" field on input from file . . : YES YES, NO

CALCULATION SPECIFICATION			
Value 1 Opera	tion Value 2	Result	
S	0	GRSPRC	
NETPRC	/ 100	*WORKNET01	
*WORKNET01	* SALTAX_	*WORKTAX01	
NETPRC	+ *WORKTA	X01 GRSPRC	
		+	

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

3.15.7 Date Conversion of a Field

This screen is displayed when the option to perform a Date conversion for a virtual field has been chosen from the Virtual field extended definition action box.

DC@P201608	Date conversion of a field
File : XXXXXXXX	** ****
Virtual field : XXX	xxxxxxx xxxxxxxxxxxxxxxxxxx
Derive "virtual" field Setup "real" field bef	on input from file YES, NO ore output to file YES, NO
"Real" field to conve	rt from
"Virtual" field date fo	ormat is
"Real" field date form	nat is
Fnn=Help Fnn=Exit	Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delet

Working from the Date conversion of a field screen you can:

- **Review** the date conversion extended definition of the virtual field currently being worked with.
- **Specify** the date conversion extended definition of the virtual field currently being worked with.
- **Change** the date conversion extended definition of the virtual field currently being worked with.
- **Delete** the date conversion extended definition of the virtual field currently being worked with. There is no confirm option when delete has been

specified for a virtual field extended definition.

Input Options

File

Is the name of the physical file currently being worked with.

Virtual Field

Is the name of the "virtual" field within the physical file currently being worked with. The value of the "virtual" field will be the result of the date conversion from the "real" field. The "virtual" field must be large enough to contain the result of/value for the date conversion otherwise unpredictable results could occur.

Derive "Virtual" Field on Input from File

Specifies whether the "virtual" field should be derived from the "real" field ("real" field can be either a "real" or "virtual" field name within this file) on input from file. Valid entries are "YES" or "NO".

Setup "Real" Field Before Output to File

Specifies whether the "real" field ("real" field can be either a "real" or "virtual" field name within this file) should be setup before output to file. Valid entries are "YES" or "NO".

"Real" Field to Convert from

Specifies the name of the field ("real" or "virtual" field name within this file) to be used for the date conversion to the resulting "virtual" field. The "real" field must be large enough to contain the result of/value for the date conversion otherwise unpredictable results could occur.

By entering a partial name or a '?' the single field selection screen will be displayed. The selection list will be built from the real and virtual fields in the current file's definition. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.

"Virtual" Field Date Format Is

Specifies the date format to be used to convert the specified "real" field ("real" or "virtual" field name within this file) to the value of the "virtual" field. Valid date format values for this field are as follows:

SYSFMT	(QDATFMT)
SYSFMT8	(8 digit date in QDATFMT format)

DDMMYY	(Day, Month and Year)
MMDDYY	(Month, Day and Year)
YYMMDD	(Year, Month and Day)
DDMMYYYY	(Day, Month and Full Year)
MMDDYYYY	(Month, Day and Full Year)
YYYYMMDD	(Full Year, Month and Day)
YYMM	(Year and Month)
MMYY	(Month and Year)
YYYYMM	(Full Year and Month)
MMYYYY	(Month and Full Year)

"Real" field date format is

Specifies the date format of the "real" field ("real" or "virtual" name within this file). Valid date format values for this field are as follows:

SYSFMT	(QDATFMT)
SYSFMT8	(8 digit date in QDATFMT format)
DDMMYY	(Day, Month and Year)
MMDDYY	(Month, Day and Year)
YYMMDD	(Year, Month and Day)
DDMMYYYY	(Day, Month and Full Year)
MMDDYYYY	(Month, Day and Full Year)
YYYYMMDD	(Full Year, Month and Day)
YYMM	(Year and Month)
MMYY	(Month and Year)
YYYYMM	(Full Year and Month)
MMYYYY	(Month and Full Year)

CYYMMDD (Century indicator, Year, Month and Day)

Notes

- The CYYMMDD format is only valid as a format for a real field on LANSA virtual field derivation. It can be defined in Visual LANSA, but not made operational.
- If an alpha date field (real or virtual) is being converted to a resulting numeric date field (real or virtual), a check will be automatically performed to ensure the resulting numeric date field contains all numeric characters ("0" to "9"), if the resulting numeric date field does not contain all numeric characters (i.e. alpha characters have been found) the resulting numeric date field will be set to 0.
- If a date containing just Month and Year is to be converted to a date containing Day, Month and Year, a day of 01 will be used for the resulting date. eg MMYY of 1296 to be converted to DDMMYYYY will result in 01121996.
- If a date containing zeros is to be converted to, for example, YYYYMMDD format, the resulting date will be 19000000.
Date Conversion Examples

This screen demonstrates the use of the "Date conversion of a field" option and how it may be used to convert a date from one date format to another date format.

Convert the Employee's start date field EMPSDT which is stored in YYMMDD format in the Employee master file EMPMST to DDMMYY format in the EMPDMY field for screen inquiry purposes:

DC@P201608 Date Conversion of a Field

File: EMPMST Employee Master file

Virtual field: EMPDMY Employee start date (screen display format)

Derive "virtual" field on input from file . . . YES YES, NO Setup "real" field before output to file . . . NO YES, NO

"Real" field to convert from EMPSDT

"Virtual" field date format is DDMMYY

"Real" field date format is YYMMDD

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delete

3.15.8 Virtual Field RPG Source Code Maintenance

This screen is displayed when the option to review or edit the virtual field RPG source code has been chosen on the screen described in preceding sections.

DC@P201602 Edit Virtual Field Derivation Code

Choose section(s) of file I/O module in which virtual field derivation code is to be included

Choose	I/O module code section	RPG type
_	File specifications	'F" specs
_	Array specifications	"E" specs
_	External record format or field re	names "I" specs
_	Data structure specifications	"I" specs
_	Calculations after input from file	"C" specs
_	Calculations before output to file	"C" specs
_	Internal subroutines	"C" specs
_	Output specifications	"O" specs
_	Compile time array data	N/Applicable

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Working from the Edit Virtual Field Derivation Code screen you can:

- **Review** any of the various RPG code sections that are used to derive the virtual fields.
- **Edit** any of the various RPG code sections that are used to derive the virtual fields.

To edit or review any RPG code section(s) enter any non-blank character beside the desired section and press the enter key.

If no sections are chosen for review or edit, the list of virtual field definitions

display, described in 3.15.3 Virtual Fields Maintenance, will be re-displayed.

The sections of RPG code displayed describe the parts or portions of the RPG file I/O module in which you can include user code that is to be used to derive virtual fields.

More details about the RPG I/O module that is created for every file defined within LANSA are supplied elsewhere.

When LANSA is automatically generating the RPG code for an I/O module it looks for any virtual field RPG code associated with the file and automatically includes it at the correct point.

3.15.9 Edit Virtual Field RPG Code

When a section of the virtual field RPG code has been chosen for review or edit a screen will be presented. This screen can be presented by the SEU (edit source) utility.

SEU is provided by IBM and is used to create and manipulate source or text.

SEU is widely used by IBM i programmers and other users. If you are unfamiliar with the features and functions of SEU refer to the appropriate IBM manual.

In addition, SEU has online HELP facilities. Once you have started to use SEU press the HELP key and review the extensive HELP text available.

By using SEU, the RPG code section can be reviewed or edited as desired.

Once all reviewing or editing has been completed use function key 3 to exit from the SEU utility.

A screen similar to this example will be presented:

Exit Type choices, press Enter. Change/create member <u>N</u> Y=Yes, N=No Member <u>EDITWORK</u> Name

File <u>EDITWORK</u> Name Library <u>QTEMP</u> Name Text				
Resequence member Y Y=Yes, N=No Start 0001.00 0000.01 - 9999.99 Increment 01.00 00.01 - 99.99				
Print member <u>N</u> Y=Yes, N=No				
Return to editing $\dots \dots N$ Y=Yes, N=No				
Go to member list <u>N</u> Y=Yes, N=No				
F3=Exit F5=Refresh F12=Cancel				

The SEU exit option - change/create member, will be pre-filled to N if you did not change the RPG code section in any way. Otherwise it will be pre-filled to Y.

Normally you would use the option "Y" to change/create member and exit from SEU to return to LANSA. Notice also the option to return to editing. Other options can be used if desired, but you may find that any changes made are lost and the RPG code section remains unchanged within LANSA.

Note that the member, file and library always contain EDITWORK, EDITWORK and QTEMP respectively. This applies no matter what file or section's virtual field RPG code is being edited. This is a special temporary source file that LANSA has created in library QTEMP. These values can be changed if required, but again you may find that any changes are lost and the RPG code section remains unchanged within LANSA.

Edit Virtual Field Examples

These examples apply to specifying virtual fields in a file definition. They do not demonstrate how the associated RPG derivation code would be specified.

Refer to the Examples of Virtual Fields & Derivation Code for more comprehensive examples of the virtual field facility.

• Define three virtual fields called COMPNO, DIVNUM and COSTCT:

Seq	Field	Description
1	COMPNO	Company number
2	DIVNUM	Division number
3	COSTCT	Cost center

• Define two virtual fields called YYMMDD and MONTH:

Seq Field		Seq Field Description		Description
10	YYMMDD	Date in format YYMMDD		
20	MONTH	Month of the year		

3.16 Batch Control Logic

Batch control logic is used to define the logic by which numeric fields in one file are to be accumulated into fields in another file.

3.16.1 How Batch Control Logic is applied

3.16.2 Steps to Define, Review, or Change Batch Control Logic

3.16.3 Display Existing Batch Control Logic

3.16.4 Detailed Batch Control Logic Maintenance

Batch Control Examples

3.16.1 How Batch Control Logic is applied

Batch control logic (even though it is not always referred to by that name) is common. Some examples of its use are:

• A "batched transaction" data entry application may use a control file structure that looks like this:

Daily Totals	Batch Totals	Entry Data	
DATE	DATE	DATE	
DDEBIT	BATCH	BATCH	
DCREDIT	BDEBIT	TRANSACTION	
	BCREDIT	ACCOUNT	
		DESCRIPTION	
		DEBIT	
		CREDIT	

In this structure BDEBIT and BCREDIT are totals of all DEBIT and CREDIT values for a particular DATE and BATCH. Likewise, DDEBIT and DCREDIT are totals of CREDIT and DEBIT values for a particular DATE.

• An order header and an order lines file have the following fields defined in them:

Order Header	Order Lines
ORDNUM	ORDNUM
CUSTNO	ORDLIN
DATE	PRODNO
TOTDUE	QUANTY
	VALUE

In this structure TOTDUE (in order header) is a count of all associated VALUE fields in the order lines file. i.e. TOTDUE is the total value of all associated invoice lines.

In either of these cases the batch control component of LANSA can be used to automatically maintain the "control totals" in the other files.

When setting up the definition of the "Entry data" file in the first example, the batch control logic component would be used to indicate that:

- Fields DEBIT and CREDIT in the "Entry data" file should be accumulated by DATE and BATCH into fields BDEBIT and BCREDIT in the "Batch totals" file.
- Fields DEBIT and CREDIT in the "Entry data" file should be accumulated by DATE into fields DDEBIT and DCREDIT in the "Daily totals" file.

When the I/O module for the "Entry data" file is compiled (see elsewhere for a description of I/O modules) it will contain logic to automatically maintain the batch control fields in the "Batch totals" and "Daily totals" files.

This means that whenever a record is added, updated or deleted from the "Entry data" file the totals will be maintained. No other logic is required to maintain the totals.

The logic generated will handle changes to the batch control "keys" as well. Thus even if a record in the "Entry data" file was changed from BATCH 1234 to 4567 and DATE 010187 to 150187 the control totals will be maintained correctly.

If a user creates a function to add records to the "Entry data" file it will use the I/O module. The I/O module will maintain the batch control totals in the other 2 files without the user even knowing of their existence.

The actual procedure for specifying batch control logic is described in the following sections.

3.16.2 Steps to Define, Review, or Change Batch Control Logic

The steps involved in defining, reviewing or changing batch control logic associated with a file definition are as follows:



3.16.3 Display Existing Batch Control Logic

This screen is displayed when:

- Reviewing the batch control logic in a file definition for possible amendment or via delete and recreate.
- Defining new batch control logic

DC@P201001 Select Batch Control Block

Select an existing Batch Control block to be reviewed, changed or deleted

Or use the ADD command key to add a new Batch Control block

Batch Control

Sel	File Name	Description
_	XXXXXXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Cancel Fnn=Messages

Working from the Select Batch Control Block screen you can:

- Select existing batch control logic for detailed review only. To do this enter any non-blank character in the column labeled "Select" against the batch control logic required.
- Create a new batch control logic. Use the Add/Create function key to indicate that new batch control logic is to be added to the file definition.

3.16.4 Detailed Batch Control Logic Maintenance

- The Display Batch Control Logic screen is displayed when reviewing existing access batch control logic in detail.
- The Create Batch Control Logic screen is displayed when specifying the details of new batch control logic.

DC@P201002	Create Batch Control Logic
File : XXXXXXXXX	x xxxxxxxxx xxxxxxxxxxxxxxxxxxxxx
Description	••
Batch Control File Na	ame
Linked Batch Control -In this file -In control file	fields
Key to Access Contro —	ol file

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note: The Delete function key is only enabled when reviewing existing batch control.

Working from the Create Batch Control Logic screen you can:

- Review the full details of existing batch control logic.
- Delete existing batch control logic. Use the Delete function key to indicate that the batch control logic should be deleted. On the resulting display enter YES to confirm that the batch control logic should be deleted. If YES is not entered the batch control logic will not be deleted.

Note that there is no Change function. To change batch control logic it must first be deleted then (re)created.

Input Options

These input options apply when defining new batch control logic to be included into a file definition:

Description

Specifies a description that will aid other users of this file definition in identifying the purpose of the batch control logic. Entry of some form of description is mandatory.

Batch Control File Name

Specifies the name of the physical or logical file that is to be maintained by this batch control logic. The physical or logical file specified must have already been defined to the LANSA system and made operational.

The file specified cannot contain batch control logic itself. If this causes a problem, investigate moving the batch control logic from the "batch control file" to this file (the file definition currently being worked with). In most cases this technique will satisfy all batch control requirements.

File name can be entered in full, partially or left blank. If required a list of available files will be displayed and the required file chosen.

Linked Batch Control Fields

Specifies the "pairing" of the batch control fields from this file (i.e. the file definition being worked with) with those from the file nominated as the "batch control file name". At least one pair of fields is required.

Note that while LANSA will check that all fields nominated exist in their respective files, and are numeric, it will not check the sizes. Thus it is possible to accumulate a field with 4 decimal positions into a field with no decimal positions. All decimal precision would be lost and the accumulation would probably be "rubbish".

Likewise a field with 15 significant digits could be accumulated into a field with

1 significant digit. Again the accumulation would almost certainly be "rubbish". **Note:** The checking (and testing) of accumulated field precision is a user responsibility and is not performed by LANSA.

In this file

Specifies from 1 to 4 fields in this file (i.e. the file definition currently being worked with) that are to be accumulated into the "batch control file". All fields nominated must be defined in this file and must be numeric.

In control file

Specifies from 1 to 4 fields in the "batch control file" that are to hold the accumulations of the fields from this file (nominated directly above). All fields nominated must be defined in the "batch control file" and must be numeric.

By entering a partial name or a '?' in these fields the single field selection screen will be displayed for each. The selection list will be built from the real fields in the current file or control file as appropriate. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.

Keys to Access Control File

Specifies the fields or values that should be used to form the key that will be used to access the appropriate record in the file named as the "batch control file". If field names are used then they must be defined in this file (i.e. the file definition that is being worked with - not the file named as the "batch control" file).

At least one key field or value is required. Up to 20 key fields or values can be specified. Use the ROLL UP key to enter more.

Entries made can be:

- An alphanumeric literal (in quotes) such as 'NSW', 'BALMAIN'
- A numeric literal such as 1, 14.23, -1.141217.
- Another real field name such as CUSTNO, INVNUM, etc. As mentioned the field must be defined in the current file definition. Please note that you must not use a virtual field. If you need to use a virtual field in the key, then code the batch control update logic into a trigger. Use of virtual fields may lead to unpredictable and/or unexpected results. (A '?' will display the single field selection screen. The selection list will be built from all the real fields in the current file. Refer to 3.10.1 Select Fields When Working from File Definition Menu for more details.)
- A system variable name such as *BLANKS, *ZERO, *DATE or any other

system variable defined at your installation.

Key values are checked for type and length compatibility. The entire key list supplied is checked for compatibility with the actual key(s) of the "batch control file". The key list specified can be a full or partial key to the file.

A warning is issued if a partial key list is specified. The use of a partial key in this particular situation would be rare. If a warning message is issued carefully check/reconsider exactly what batch control logic has just been defined.

Batch Control Examples

These examples apply to defining batch control logic within a file definition.

Example 1

The "batched transaction" data entry application example used at the beginning of this section has 3 control files that were defined as follows:

File name:		DAYTOT	BCHTOT	ENTDTA
Description:		Daily totals	Batch totals	Entry data
Fields:		DATE	DATE	DATE
		DDEBIT	BATCH	BATCH
		DCREDIT	BDEBIT	TRANSACTION
			BCREDIT	ACCOUNT
				DESCRIPTION
				DEBIT
				CREDIT
Primary key	1	DATE	DATE	DATE
	2		BATCH	BATCH
	3			TRANSACTION

In this structure BDEBIT and BCREDIT are totals of all DEBIT and CREDIT values for a particular DATE and BATCH. Likewise, DDEBIT and DCREDIT are totals of CREDIT and DEBIT values for a particular DATE.

To define this particular batch control logic requirement to LANSA the following batch control logic details should be added to the definition of file **ENTDTA (Entry data):**

File Definition Name	Batch Cntl File Name	Linked Fields In This File	Linked Fields In Cntl File	Keys To Access Cntl File
ENTDTA	BCHTOT	DEBIT	BDEBIT	DATE

		CREDIT	BCREDIT	BATCH
ENTDTA	DAYTOT	DEBIT	DDEBIT	DATE
		CREDIT	DCREDIT	

Example 2

The second example at the beginning of this section dealt with an order header and order lines file that were defined as follows:

File name:		ORDHDR	ORDLIN
Description:		Order header	Order lines
Fields:		ORDNUM	ORDNUM
		CUSTNO	ORDLIN
		DATE	PRODNO
		TOTDUE	QUANTY
			VALUE
Primary key	1	ORDNUM	ORDNUM
	2		ORDLIN
	3		

In this structure TOTDUE (in order header) is a count of all associated VALUE fields in the order lines file. i.e. TOTDUE is the total value of all associated invoice lines.

To define this particular batch control logic requirement to LANSA the following batch control logic details should be added to the definition of file **ORDLIN (order lines):**

File Definition Name:	ORDLIN
Batch Control File Name:	ORDHDR

Linked Fields In this File:VALUELinked Fields In Control File:TOTDUEKeys to Access Control File:ORDNUM

3.17 Database File Attributes

When a file definition is set up under LANSA some of the things that it contains include:

- The names of the fields in the file
- Any logical views of the file that are required
- Rules/triggers that are to be performed
- Access routes to other files
- Batch control logic

Using this information LANSA can create the required physical file, all required logical files and the I/O module required to support I/O to the file.

However, whenever a physical or logical file is set up, it is also possible to specify to the IBM i operating system a series of "attributes" that influence how the file is to be set up and used. Some of the attributes available include:

- How big the file is allowed to be
- When the file is to be recovered after a system failure
- Whether space should be allocated for the file when it is created
- Whether or not the file is to be under commitment control

There are several other attributes, each of these is described in detail in the following sections.

3.17.1 Steps to Review or Change Database File Attributes

The steps involved in reviewing or changing the database file attributes are as follows:



3.17.2 Database File Attributes Maintenance

A screen similar to this example is displayed when the option to review or change database file attributes is chosen from the file definition menu. This example is a non-RDMLX view:

DC@P201501 Database File Attributes
File : TESTFILE DC@TOOLLIB Test name and address file(XX)Description : Test name and address fileOS/400 HSTab : NOMaintained by: LANSA Record format name : TESTFILE I/O Mod Lib : ECommitment : NO AUTOCOMMIT parameter : NO Share/Secure : NO / YStrip debug : YES Suppress IOM0034 msg : NO Ignore decimal error: NOI/O Module : YES Create Batch Ctl Rcd : NO ALTSEQ : TABLE1CRTPF&CHGPF parameters: SIZE(5000 1000 10) LVLCHK(*YES)
View name Rcd format Description ALTSEQ TESTV01 TESTV01 Names and addresses by customer TABLE2 Parameters LVLCHK(*YES)
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

For an RDMLX partition, the view will be slightly different:

DC@P201501

Database File Attributes

File : TESTFILE DC@TOOLLIB Test name and address file (XXX	
Description : <u>Test name and address file</u> OS/400 HSTab : <u>NO</u>		
Maintained by: <u>LANSA</u> Record format name: <u>TESTFILE</u> I/O Mod Lib		
Commitment : <u>NO</u> AUTOCOMMIT parameter : <u>NO</u> Share/Secure: <u>N</u>		
Strip debug : <u>YES</u> Suppress IOM0034 msg : <u>NO</u> Ignore decimal error: <u>N</u>		
I/O Module : <u>YES</u> Create Batch Ctl Rcd : <u>NO</u> ALTSEQ: <u>TABLE1</u> : <u>I</u>	RDM	
RDMLX DB trigger used: NO DB trigger PGM : *NONE		
CRTPF&CHGPF parameters: SIZE(5000 1000 10) LVLCHK(*YES)		
View name Rcd format Description ALTSEQ		
@DEVJLF @REC01V1 View of @dej with key		
Parameters		
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages		

Working from the Database File Attributes screen you can:

- **Review** the existing database attributes of the physical file and all logical files associated with the file definition. Use the ROLL keys to scroll backwards and forwards through the list of logical files that is shown at the bottom of the screen.
- **Change** the existing database attributes of the physical file and all logical files associated with the file definition. To do this use the Change function key to make the screen input capable and make the desired changes.

Input Options

These input options apply to reviewing or changing the attributes of database

files associated with the LANSA file definition.

The input options are described in two parts:

Input Options - Physical File Attributes (Upper Half Of Screen)

Input Options - Logical File Attributes (Lower Half of Screen)

Input Options - Physical File Attributes (Upper Half Of Screen)

Description

Specifies the description of the physical file. If no description is specified it will default to the physical file name. This description is used within LANSA and within the operating system to aid system users in identifying the file.

OS/400 HSTab

Specifies whether this file definition and associated logical views should be mirrored into a high speed OS/400 User Index to allow more rapid access in "read only" situations.

YES indicates that the file should be mirrored in a high speed index. NO indicates that the file should not be mirrored in a high speed index. NO is the default value.

For further information refer to More About High Speed Tables.

Maintained By

Specifies who is to create and maintain this file definition. Allowable values are:

- LANSA Indicates that the actual physical and logical files that result from this file definition are to be maintained (i.e. created, changed and deleted) by the LANSA system.
- OTHER Indicates that the actual physical and logical files associated with this file definition are to be maintained (i.e. created, changed and deleted) by some OTHER system or mechanism.

Use this option when you wish to load the definition of an existing file into LANSA, thus making the file accessible through LANSA but do not wish to have LANSA responsible for the setup and maintenance of the associated physical and logical files.

If this option is used it is possible to use the facility "Load definition of file not maintained by LANSA" from the file definition menu to automatically load the definition of associated physical and logical files into the LANSA system. Refer to 3.19 Load the Definition of a File NOT Maintained by LANSA for more information about this facility.

WARNING: If this value is shown as OTHER, use EXTREME CAUTION when changing it to LANSA. You are saying that the future setup and maintenance of this file is to be done by LANSA. This may upset existing applications that already use the file. It is recommended that you do NOT make this type of change unless you are fully aware of ALL the ramifications. Contact your product vendor for further advice.

Record Format Name

Specifies the record format name to be assigned to the physical file record. When the file definition was first set up this value was initialized to be the same as the file name. Change name as desired. The name specified must conform to IBM i record format naming conventions and must be no more than 10 characters long. It is recorded that no more than 8 characters are used.

I/O Mod Lib

Specifies what library the file's I/O module will reside.

F, which is the default value, indicates that the I/O module will reside in the same library as the file.

M, which is the other allowable value, indicates that the I/O module will reside in the partition module library.

Commitment

Specifies whether or not the file is to be placed under commitment control. Refer to the appropriate IBM supplied manuals for more details of commitment control and commitment control processing. Commitment control is a facility provided by the IBM i operating systems.

Allowable values for this field are:

YES The file is to be under commitment control

NO The file is not to be under commitment control

When a file definition is first created the default value for commitment control (YES or NO) is set from the system definition data area. Refer to System Definition Data Areas for more details and information about how to change the system default values.

Note: Using this option indicates that the file is to be placed under

commitment control all the time, in all applications. To selectively use (or not use) commitment control, refer to the *PGMCOMMIT / *NOPGMCOMMIT options of the FUNCTION command in the *Technical Reference Guide*.

AUTOCOMMIT Parameter

This option was made redundant by LANSA version 4.0 at program change level E5.

Share

Specifies whether or not this file (and any of its associated logical views) should be opened with the option to share an open data path.

Despite the fact that IBM recommend data path sharing as one of the fundamental design considerations for efficient applications, the default for LANSA created files is NO (do not share open data paths). The reason for this is that the I/O module concept implicitly emulates an open data path, because one active I/O module is shared by all RDML functions (within a job) that are accessing a file.

Generally use option NO (do not share an open data path), except in the special situation where the file is only being used as a "joined record format" for data that is dynamically created by the IBM OPNQRYF (open query file) command.

Note that this value only affects the way that the I/O module (and hence RDML applications) open the file at run time.

To actually get the physical file and logical files defined to the operating system as SHARE(*YES) or SHARE(*NO) for the benefit of external application programs, refer to the CRTPF/CHGPF and CRTLF/CHGLF options below.

Note: This option relates to the opening of the file's open data path when used in I/O modules and *DBOPTIMISE programs, this option does not relate to the creation or changing of the database file attributes.

Secure

Specifies whether or not this file (and any of its associated logical views) should be opened with the option to be secured from file override commands already issued by higher level program(s).

Refer to the appropriate IBM supplied manual(s) for more information about the SECURE parameter on file override commands and the effects of using it.

The default for this field is YES. Do not change this value unless you understand what it does and how it is used.

Strip Debug

Specifies whether or not the RPG I/O module associated with the file definition should have its associated debug symbolic information removed. This information is only required in two situations:

• When attempting to use the IBM supplied debugging aids with the compiled RPG I/O module.

Since these situations are relatively rare, the default for this field is YES (debugging information should be stripped). By using this option, the size of the compiled I/O module will typically be reduced by 40 to 60%. This size reduction has no bearing on execution speed, just on the size of the compiled object.

To enable RPG I/O modules to be debugged, use option NO (debug information should not be stripped).

Suppress IOM0034 Msg

When an I/O module is used to access a file that has never been directly defined to LANSA, it issues message IOM0034 indicating that no LANSA level security information exists for the file.

To suppress this message specify YES.

To allow this message to be issued, specify NO (message is not to be suppressed).

The default value for this option is NO.

Ignore Decimal Error

Specifies whether or not the RPG I/O module associated with the file definition should be compiled with the IGNDECERR (ignore decimal data error) option. The default is NO. The only other possible value is YES. The use of YES is not recommended.

While the IGNDECERR option is available in LANSA, the IBM warnings about using this option still apply. Should using this option cause problems your product vendor may assist you in correcting them, but obviously cannot accept any responsibility for the problem or its cause. Do not use this option unless absolutely necessary. Refer to the IBM i CRTRPGPGM command for more details of the IBM warnings and disclaimer before attempting to use this option.

I/O Module

Specifies whether or not an I/O module is required for this file and its associated logical views.

YES, which is the default value, indicates that an I/O module is required for this file and its associated logical views.

NO, which is the other allowable value, indicates that an I/O module is not required for this file or any of its associated logical views.

To effectively use NO, you must use the FUNCTION RDML command with the OPTIONS(*DBOPTIMISE) keyword in every RDML function that attempts to access this file or any of its associated logical views. If this command is omitted from the RDML function it will compile correctly, but at execution time fail because calls it places to the non-existent I/O module will be unresolved.

It is strongly recommended that you do not use NO until you have had some experience with the LANSA product and are familiar with the concept and workings of I/O modules.

Refer to Using *DBOPTIMIZE / *DBOPTIMIZE_Batch and the FUNCTION command in the *Technical Reference Guide* for more details of how to set up and use a system without using I/O modules.

Note that if this value is changed from YES to NO, and the resultant change made operational, any existing I/O module will be automatically deleted. After this has been done, all existing RDML functions that access this file without using *DBOPTIMISE will fail when used. This is because they are attempting to resolve to a now non-existent I/O module. In such cases, add the *DBOPTIMISE option to the RDML functions involved and recompile them.

Create Batch Ctl Rcd

Specifies whether or not other database files that perform batch control totaling against this file should be allowed to automatically create missing "batch" records in this file.

NO, which is the default, specifies that when a batch control record cannot be found in this file, the other file operation should fail with error message IOM0032.

YES, which is the other allowable value, indicates that when a batch control record cannot be found in this file, the other file's I/O operation is allowed to automatically create one.

The other file's I/O operation is modified to create the missing batch control record in this file like this:

• All fields in this file's record format are set to blanks (alphanumeric fields)

or zero (numeric fields)

- The field(s) or value(s) from the other file that were used as key(s) to access the batch control record in this file are mapped into their corresponding field(s) in this file. This mapping is by key matching, not by name.
- The new record thus assembled is inserted into this file.

Before attempting to use this option ensure that you understand the ramifications of using it and then thoroughly test all resulting applications.

Also note that when changing this option, it is necessary to make the amendment to this file operational. Additionally, all I/O modules for other files that perform batch control totaling against this file must also be (re)compiled.

ALTSEQ

Specifies the name of an Alternate Collating Table to be used when sequencing the records for retrieval on a Keyed file. The Alternate Collating Table name if specified is not validated that it actually exists. When attempting to use any file that has been created with an Alternate Collating Table the library list is used to locate the Table file.

Note: At File Creation time you must have operational rights to the Alternate Collating Table. Alternate Collating sequences are not valid for key fields with a data type of packed decimal. Causes zoned key fields to default to unsigned sequence.

DB triggers

If DB triggers used is YES, then this file will generate operating system level DB triggers.

DB trigger PGM is the name of the program that will be generated to action the DB trigger(s).

CRTPF & CHGPF Parameters

Specifies any additional command parameters that are to be used by LANSA when creating (CRTPF command) or changing (CHGPF command) the physical file.

When the file definition is set up this value is initialized to include SIZE and LVLCHK parameters which are set from the default values specified in the system definition data area. Refer to System Definition Data Areas for more details.

When specifying parameters input them exactly as would be done when

entering them through the IBM i command entry facility. Use "keyword" (rather than positional) specification of any parameters required.

Parameters specified are checked for validity. If invalid, the screen will be redisplayed with an error message.

Parameters that can be specified include EXPDATE, MAXMBRS, MAINT, RECOVER, FRCACCPTH, SIZE, ALLOCATE, UNIT, FRCRATIO, WAITFILE, WAITRCD, SHARE, DLTPCT and LVLCHK.

Note: The SHARE parameter relates to the CRTPF & CHGPF command common parameters only, it does not relate to the share an open data option.

Refer to the IBM supplied manual *Control Language Reference Manual* for more details of the CRTPF and CHGPF commands and the associated common parameters.

Input Options - Logical File Attributes (Lower Half of Screen)

Rcd format

Specifies the record format name to be assigned to the logical file record. When the file definition was first set up this value was initialised to be the same as the file name. Change name as desired. Name specified must conform to IBM i record format naming conventions and must be no more than 10 characters long.

Description

Specifies the description of the logical file. If no description is specified it will default to the logical file name. This description is used within LANSA and within the operating system to aid system users in identifying the file.

ALTSEQ

Specifies the name of an Alternate Collating Table to be used when sequencing the records for retrieval on a Keyed file. The Alternate Collating Table name if specified is not validated that it actually exists. When attempting to use any file that has been created with an Alternate Collating Table the library list is used to locate the Table file.

Note: At File Creation time you must have operational rights to the Alternate Collating Table. Alternate Collating sequences are not valid for key fields with a data type of packed decimal. Causes zoned key fields to default to unsigned sequence.

Parameters

Specifies any additional command parameters that are to be used by LANSA when creating (CRTLF command) or changing (CHGLF command) the logical file.

When the file definition is set up this value is initialised to include a LVLCHK parameter which was set from the default specified in the system definition data area. Refer to System Definition Data Areas for more details.

When specifying parameters input them exactly as would be done when entering them through the IBM i command entry facility. Using "keyword" (rather than positional) specification of any parameters required.

Parameters specified are checked for validity. If invalid the screen will be redisplayed with an error message.

Parameters that can be specified include MAXMBRS, RECOVER, FRCACCPTH, UNIT, FMTSLR, FRCRATIO, WAITFILE, WAITRCD, SHARE and LVLCHK.

Note: The SHARE parameter relates to the CRTLF & CHGLF command common parameters only, it does not relate to the share an open data option.

Refer to the IBM supplied manual *Control Language Reference Manual* for more details of the CR073mvxTLF and CHGLF commands and the associated common parameters.

More About High Speed Tables

These points provide basic information about the IBM i high speed table facility. They should all be read and understood before this facility is used in any way.

- A high speed table is not a "thing" in itself. A high speed table is a normal LANSA file definition that has its "high speed" flag set to YES.
- A LANSA file definition flagged as high speed table is actually implemented as a normal database file. All functions which insert, update or delete data in the table actually access the normal database file.

This normal database file actually contains the data, so there is no difference in the risk of data loss between a normal file definition and a high speed table.

This also means that you can, at any time, set the high speed table flag back to NO and revert to a normal database file again without any loss of definition or data.

The difference between a normal file and a high speed table is that a high speed table uses an extra object (over and above the normal database file). This object is called an IBM i "User Index" and it contains a duplication or "mirror" of the data in the associated database file and its logical views.

- Functions that only read the file actually access the "mirror" data in the user index. Such a method of access has some strong advantages:
 - It is very fast.
 - There is no open or close overhead.
 - There is very little of the normal space (PAG) overhead associated with having a normal database file open.

In a traditional commercial application that had, say, 40 database files open, there is a significant overhead in space and time to open and keep open the 40 files.

However, if 20 of these files were implemented as high speed tables, then the space and time overheads would be reduced by a factor of approximately 50%. This would probably significantly enhance the performance of this application.

- The high speed table index is an IBM i "platform specific" option. On other platforms the "high speed table" flag may be ignored and the file would be implemented just like any other file.
 This may be a design consideration. Do not use the very high speed of IBM i User Indexes to "over-engineer" an application to the point that it will not be able to function on other platforms using normal database management facilities.
- For details of the IBM i User Index facility and its use, refer to IBM documentation.
- It is possible for existing 3GL based application programs to also access the high speed tables. Contact your product vendor for more information about this feature.

To be valid as a high speed table a file definition must conform to the following rules. Most of these things are checked during the "make operational" phase of creating/changing a file. If a rule is violated the make operational will fail with appropriate error message(s). These rules apply to the basic physical file definition and all logical views defined over it:

- No form of alternate collating sequence is supported. The IBM i User Index facility only supports simple binary collation. From the manual that elaborates upon access to an IBM i User (or Independent) Index:
 "Each entry is inserted into the index at the appropriate location based on the binary value of the argument. No other collating sequence is supported."
- All key fields must be ascending, unsigned values.
- When a file with date, time or timestamp fields in its key list is mirrored in a high speed table, a LANSA function with read-only access to the file will not use the I/O module. The date, time or timestamp field is treated as an alphanumeric field in the high speed table. Therefore values must be entered in full (for example, as 1999-01-02 not as 1999-1-2) when fetching a record. Also, if an invalid value is entered, the LANSA function will not check if it is a valid date, time or timestamp, but just return a not-found status.
- The table can have no more than 799 fields.
- The maximum table entry record length depends on the system data area DC@OSVEROP. If option *HSTABEXTEND has been inserted, the maximum entry record length is 1988 bytes (this is an IBM i limit) and a maximum key length of 108 bytes (this is a LANSA limit for storage and performance reasons). The key is included in the 1988 record length. If option *HSTABEXTEND is not in the system data area, the table entry

record length cannot exceed 108 bytes.

Warning: Entry record lengths greater than 108 bytes cannot be saved to or restored from an IBM i release prior to V2R2M0. Note that, for packed fields, their decimal length is counted, rather than their byte length. Refer to the Compile and Edit Settings in *Review System Settings* for information about setting these option.

- The base physical file must have one or more primary key fields.
- The concepts of file members, run time library list changes and any form of file override or rename are NOT supported in any way in the high speed table execution environment. There is one high speed table index per LANSA partition. When an application is invoked that needs to access the index, it uses the single index associated with the current partition.
- No select omit logic can be specified.
- No batch control logic can be specified.
- No form of open, read or close trigger can be specified for any field in the file, either at the dictionary or file level.
- No virtual fields or logic (code) can be defined.
- No read security will be actioned for the table. This means that a function cannot be stopped from reading the content of the high speed table. However, they can be stopped from modifying it in the normal manner (because they are actually modifying the normal database file, not the high speed index).

This restriction exists to ensure maximum performance in read only applications. Applying read security as above would severely impact the performance of tables where only a few accesses are made.

In fact the security checking time would be far longer than the actual time taken to access many table entries.

• Functions that modify (INSERT, UPDATE or Delete) files that are tagged as high speed tables cannot use *DBOPTIMISE, *DBOPTIMISE_BATCH or any other option that infers these options.

This restriction exists because the special logic required to "mirror" the real file data into the high speed index only exists in the associated I/O module. Thus all "table modifiers" must be forced to use the appropriate I/O module.

• Functions that only read from a high speed table may use *DBOPTIMISE or

*DBOPTIMISE_BATCH in the normal manner.

- When the definition (i.e. layout) of a high speed table is changed all functions and I/O module validation rules that read from high speed table rather than the real file need to be recompiled.
 Again, this restriction exists to provide maximum performance.
 By definition tables are largely static in design and content, so this should not be a problem. If it proves to be, remove the high speed table option from the definition of the file.
- No form of locking is supported in applications that only read from high speed tables. If you need record locking in a "read only" function, then your file is not a good candidate for the high speed table facility.
- The use of any of these facilities with high speed tables is not checked, but they are not supported in any form within functions that require "read only" access to high speed tables:
 - The use of the OPEN command with the *OPNQRYF option.
 - The use of the *BLOCKnnn option in any form.
 - The use of SELECT_SQL in any form.
 - The use of WITH_RRN, RETURN_RRN or any form of relative record addressing.
 - The ISS_MSG parameter in any form.
- In summary, the high speed table facility is designed for use with simple lookup and decode style files only. Files that are to be used in any other "fancy" way at all should not be implemented as high speed tables.
 Warning: It is strongly recommended that if option *HSTABEXTEND is added to system data area DC@OSVEROP to make the extended entry record length available, or is removed to limit entry length, that all files tagged as high speed tables, all read only functions that use these files, and all other I/O modules and DBoptimized functions that use high speed tables for lookup validation rules be recompiled AFTER deleting the current user index which is DC@TBLIDX if adding *HSTABEXTEND, DC@TBLIDY if removing *HSTABEXTEND.

If this is not done all functions that use a particular file and the I/O module must be recompiled at the same time or they will not be pointing to the same index. The situation will be further complicated by I/O modules and DBoptimized functions which use high speed table files for lookup validation rules also pointing to the wrong index. It may not be obvious to

the user that there is a problem as the database file and one index will be unsynchronized but this will not cause program errors.

Some common questions asked about high speed tables

Q: What type of files are candidates to be high speed tables?

Broadly speaking, database files that have these characteristics are good candidates for high speed table implementations:

- The data content is widely used for decode (e.g.: state code 'CA' is printed as description "CALIFORNIA") and validation (e.g.: is state code 'CA' valid?).
- The data content is relatively stable. (e.g.: How often is a new state acquired). Generally this means files that are not subject to continual and random change on a daily basis. A "product" file would be a good candidate if it only contained descriptive details because products are not created/changed often. However if it contained stock levels it would not be a good candidate because stock levels are continually changed.
- There are usually a small number of records in the file (say, for example, 5000 or less).
- There is usually only one application that "maintains" the file, and it is not used often (say, once per day or less often).
- The vast bulk of applications only "read" from the file for decode and validation purposes.

Q: Where is the high speed table data kept?

A LANSA file definition flagged as a high speed table is set up just like any other file. The actual file data is stored and maintained in this normal file. However, the data is also mirrored into a "read only" high speed index to allow very fast access from "read only" applications.

The high speed index is actually an IBM i User Index (object type *USRIDX). It is automatically created in, and must always remain in, the module (or program) library of the current partition. You do not have to create this index, but you may choose to periodically delete and rebuild it. See the following points for an example of this. It is named DC@TBLIDX or if option *HSTABEXTEND is in the system data area DC@OSVEROP it is named DC@TBLIDY.

Q: Do I need to backup the high speed index?

Not really. Since each individual table has an associated data real file containing the "real" data, then you can actually re-create the high speed index for all tables

in just a few minutes by using the Built-In Function REBUILD_TABLE_INDEX.

However, a synchronized backup of the index and all the associated database files containing the "real" data may simplify and speed up your restore procedures, should they need to be invoked.

Q: When is the high speed index accessed?

At various points the LANSA code translators may generate code to access database files. When this is done and the file involved is a high speed table, then the high speed index will actually be accessed instead of the real file in these situations:

- In RDML functions that only "read" from the file via CHECK_FOR, FILECHECK, FETCH or SELECT commands. When an RDML function is compiled it is checked for direct access to a high speed table. If all accesses to all the high speed tables used in the function are "read only" then the I/O will be directed to the high speed table rather than to the real database file.
- In interfile validation checks. I/O modules or *DBOPTIMISE generated code that needs to lookup a file entry as the result of a validation check will always look in the high speed index rather than the real file.

Q: Can I use *DBOPTIMISE/*DBOPTIMIZE with high speed tables?

Yes you can in all situations except where the function updates a high speed table. Functions that update a high speed table must do all their I/O to the table via the associated I/O module. This ensures that real data file and the mirrored high speed index are updated together.

Q: When is the high speed index updated?

When the I/O module for a file that is flagged as a high speed table is created extra code is added to it to count the number of inserts, updates and deletes performed to the file.

When the file is being closed this count is examined, and if greater than 0, all existing entries for the file are erased from the high speed index, then the real file (and its views) are read from end to end to insert new entries into the high speed index.

This architecture has some impacts on the use of high speed tables:

• The file and the mirror index are not actually maintained simultaneously. When the file is being closed the existing mirrored index entries are erased and then recreated from the updated version of the file.
- The file and all its views are maintained as separate high speed index data. This means that a table with 4 views actually uses 5 times the index space of the source table. One for the table and one for each of the views.
- Contention may occur if multiple users attempt to update a file that has a high speed index mirror simultaneously. This problem is easily overcome by ensuring that applications that update high speed tables are restricted to single user access.

There are a variety of simple methods that may be used to restrict a function to single user access. Contact your product vendor if require assistance in designing such an application.

Q: Can the "real" file and the index get out of synchronization?

From the preceding points it can be seen that it is possible for a file and its mirrored high speed index to get out of synchronization. For example, a function may insert 3 new entries to a table and then fail. At this point the new entries are in the real file but they are not reflected in the high speed index.

Q: How can the lack of synchronization be corrected?

If a file and its high speed index get out of synchronization then they may be desynchronized by:

- Doing a "dummy" update to the file. The associated I/O module will then rebuild the index to reflect the updated file thus synchronizing the file and index again.
- Use the Built-In Function REBUILD_TABLE_INDEX to manually trigger the I/O module to rebuild the index of one or more files.

The sequence of commands:

```
EXEC_OS400 CMD('DLTUSRIDX DC@TBLIDX')
```

or with option:

```
*HSTABEXTEND EXEC_OS400 CMD('DLTUSRIDX DC@TBLIDY')
USE BUILTIN(REBUILD_TABLE_INDEX) WITH_ARGS('''*ALL''')
```

will physically delete the entire IBM i user index area and then rebuilt the indices of all high speed table within the current partition. The first file rebuild will recreate the IBM i user index if it does not currently exist.

Q: What happens when I change the layout of a file?

If you change the layout of a file and then "make the change operational" a resynchronization of the table and index will be automatically performed. This automatic synchronization is not performed if you then export the changed

definition to another system.

Q: What happens if I import a high speed table to another system?

A high speed table is imported to another system just like a normal file. However, if the file data is imported, or the file layout is changed, the associated index is not automatically updated/reformatted. To do this you should trigger a "resynchronization" of the file and its index using any of the techniques described in this guide.

Note: A user index greater than 1 gigabyte or with an entry record length greater than 108 bytes cannot be saved to or restored from an OS400 release prior to V2R2M0.

3.18 File Multilingual Attributes

In a multilingual partition, file multilingual attributes may be specified for each language that is defined within the partition. File multilingual attributes include the physical file description and all associated logical file descriptions.

3.18.1 Steps to Review or Change File Multilingual Attributes

3.18.2 File Multilingual Attributes Maintenance

3.18.1 Steps to Review or Change File Multilingual Attributes

The steps involved in reviewing or changing the file multilingual attributes are as follows:



3.18.2 File Multilingual Attributes Maintenance

When the option to review or change a file's multilingual attributes on the file definition menu is chosen, a screen similar to this example will result:

DC@P20	01701	Multilingual File - LRTB		
File : PSI	lmst do	C@DEMOLIB		
Type	File Name	Physical or Logical File Descrip	otion	Language
Physical	PSLMST	Personeels hoofd	(DU	JT)
Physical	PSLMST	Personnel Master	(EN	NG)
Physical	PSLMST	Anagrafico Impiegato	(]	TL)
Logical	PSLMST1	PSLMST - dept,sect,empno		(DUT)
Logical	PSLMST1	PSLMST keyed on dept, sect, e	empno	(ENG)
Logical	PSLMST1	PSLMST chiave = dept,sect,e	mpno	(ITL)

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

From this screen you can:

- **Review** the multilingual information associated with a file.
- **Change** the multilingual information associated with a file. Use the Change key to place the screen in change mode and make the desired changes.

This panel will be repeated for each language group: i.e. all LRTB (Latin languages), DBCS (Ideographic or Double Byte Character Set languages) and RLTB (Bi-directional languages).

Input Options

These input options apply to changing a file's multilingual attributes:

Physical File Description

Specifies the description of the physical file. If no description is specified it will default to the physical file name. This description is used within LANSA and

within the operating system to aid system users in identifying the file.

Logical File Description

Specifies the description of the logical file. If no description is specified it will default to the logical file name. This description is used within LANSA and within the operating system to aid system users in identifying the file.

3.19 Load the Definition of a File NOT Maintained by LANSA

Generally, a file is not maintained by LANSA if it was created by, and belongs to, an application created outside of LANSA.

If such a file (frequently referred to as an OTHER file) needs to be accessible within the LANSA system, the definition must be loaded into LANSA. At the same time, any alterations to the definitions must be prohibited from within LANSA, as this could affect the system which is maintaining it.

The facility to "Load/re-load external file definitions" is designed specifically to handle these files.

You need to be sure of:

3.19.1 What Happens When a File Definition Is Loaded (or Re-Loaded)?

3.19.2 When Should a File Definition Be Loaded (or Re-Loaded)?

3.19.3 The Steps to Loading Definition of File Not Maintained by LANSA

3.19.4 Elect whether or not to Make Logical File Accessible

3.19.5 Review Messages that Result from Attempted Load

When you create a new file definition via the "Create new file definition" prompt (accessed from the file control menu) you must specify whether the file being defined is to be NOT maintained by LANSA by specifying "OTHER". (The alternative is that the file is to be maintained by LANSA by specifying "LANSA".)

Note: You must specify *OTHER_DATETIME and *OTHER_VARCHAR options if the OTHER files that you want to made known to LANSA contain:

- date (L), time (T) or timestamp (Z) fields
- variable length fields.

For more information about these options, refer to OTHER file I/O modules in Compile and Edit Settings in the Review System Settings.

This facility is NOT to be used to load a LANSA File. Instead, export/import the definition you want to use into the new partition. Do not compile the file. Set the user's library list to access the File and IO Module.

3.19.1 What Happens When a File Definition Is Loaded (or Re-Loaded)?

These are the steps taken in the loading / re-loading of a file definition:

- The user's authority to alter the LANSA file definition is checked.
- The LANSA file definition is checked to ensure that the file is defined as being "OTHER". (Maintained by some method OTHER than LANSA.)
- The fields within the external file are checked to see if they currently exist within LANSA data dictionary. If they do then their type, length, number of decimal positions and keyboard shift must match the dictionary definition. If they do not exist then the field definition is automatically added to the LANSA data dictionary.

Note: At this stage, if any FATAL errors have occurred then the process will end prematurely and the LANSA file definition will remain unchanged.

- The LANSA physical file definition is then deleted and re-loaded from the external file definition. Any rules or triggers that directly reference fields no longer in the file definition are automatically deleted.
- The LANSA definitions of any logical files based on the physical file are then deleted and re-loaded from the external file definitions. As each logical file is located, the user is asked whether or not they wish to make the file accessible via LANSA.

Note: A logical file may be "dropped" for any of these reasons during this stage of the load process:

- You elect to not make the logical file accessible
- It resides in a different library than the physical file
- It is not recognized as being a logical file
- It contains more than one record format
- Its key is made up of concatenated fields
- Its name exceeds 10 characters in length
- It has already been defined to LANSA
- An identical access path already exists

• Other errors prevent the extraction of its definition.

If a logical view is "dropped" by LANSA it just means that the logical view cannot be used / accessed via LANSA. This has no impact on any system other than LANSA.

3.19.2 When Should a File Definition Be Loaded (or Re-Loaded)?

A file definition should be loaded or re-loaded when any of these situations apply to the "external" file:-

- The physical file has been created or modified.
- A logical file which is based on the physical file has been created, modified or deleted and access to it via LANSA is required or already exists.

3.19.3 The Steps to Loading Definition of File Not Maintained by LANSA

The steps involved in loading (or re-loading) the definition of a file not maintained by LANSA can be visualized as:



3.19.4 Elect whether or not to Make Logical File Accessible

If the physical file whose definition is being loaded has logical files based on it, then a pop up window similar to this example will be presented for each and every logical file encountered:

PppFILCRT File Definition Menu (Create)

Enter number of function or place cursor on same line and press enter:

____1. Review or change fields in file definition Load Logical File Definition : DC@P201402 : : Do you wish to make this file accessible via LANSA? YES,NO : : * You already made 99 logical views accessible during this load. : : * Try to avoid making more than 10 logical files accessible. : * Investigate the command OPEN USE OPTION(*OPNORYF) to provide : access to rarely used access paths rather than loading them here. : : The keys of this logical file are as follows . . . : Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages : •

This panel allows you to specify whether or not access to the logical file from LANSA RDML programs is required.

Enter YES to specify that access to the logical file from RDML programs is required.

Enter NO to indicate that access to the logical file is not required at this time. Please read the notes on this panel and follow the guidelines whenever possible. By following these guidelines you will ensure optimal system performance. Also note that by electing to not make a logical file accessible, you are in no way committing yourself. Access to a logical file can be set up at any time later by simply (re)loading the file definition again and electing to make the logical file accessible.

3.19.5 Review Messages that Result from Attempted Load

A screen similar to this example is displayed after an attempt to load (or re-load) the definition of a file not maintained by LANSA has completed:

DC@P201401 File Load / Reload Messages

The following message(s) resulted from attempt to load or re-load this File. Please ensure that you review ALL the messages before attempting to make this file definition operational. For more information about a message place the screen cursor on the message and press HELP. Do NOT attempt to use this file if one or more fatal messages were issued.

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

When this screen is displayed you should:

- 1. Review all the messages displayed. If necessary position the screen cursor on the message and press the HELP key to read any second level text associated with the message.
- 2. Look at the last 2 messages that indicates the final status of the load (or reload) operation. Either the load will have completed successfully with "n" warning messages, or failed altogether with "n" fatal messages. If the load completed successfully review any warning messages and be sure that you understand why they were issued and the consequence of them. If the load failed altogether review the fatal messages issued and take any corrective action required or suggested. If you do not fully understand the cause or

consequence of any message do not hesitate to contact your product vendor for assistance.

- 3. After the definition of the external file has been successfully loaded (or reloaded) you may choose to add or review associated LANSA rules or triggers, access route information or batch control logic. These details are maintained purely within the LANSA system and have no impact on any "external" user applications that use the file.
- 4. If the File not maintained by LANSA uses the ALTSEQ Keyword on the Physical or any of the associated Logical files then it will be necessary to manually enter the Table name associated with the ALTSEQ keyword as the Table name is not imbedded in the file definition, only the 512 bytes of Table source information. The procedure to enter the ALTSEQ Table name is defined in 3.17.2 Database File Attributes Maintenance. This step must be exercised if the file is to be created successfully using the ALTSEQ keyword.
- 5. After the preceding steps have been completed you should submit the job to make the new or amended file definition "operational".

An optional facility is available to load several OTHER files in a single batch job. Refer to 3.9 Bulk Load of OTHER files for details.

OTHER files may also be loaded by use of the LOAD_OTHER_FILE Built-In Function. Refer to LOAD_OTHER_FILE in the *Technical Reference Guide* for details.

3.20 Tips and Techniques for Defining Files Using LANSA

3.20.1 The Chicken and the Egg Situation

3.20.2 Where Did the Changes to the File Definition Go?

3.20.3 The Make New or Amended Definition Operational Job Failed

3.20.1 The Chicken and the Egg Situation

This situation may arise when setting up a series of file definitions that refer to one another. Consider the following file definitions that are to be set up using LANSA:

FILEA: has a validation rule that references FILEB. FILEB has an access route to FILEC.

FILEB: has batch control logic that references FILEA, which has an access route to FILEA. FILEA has an access route to FILEC.

FILEC: has a validation rule that references FILEB, which has an access route to FILEA. FILEA has an access route to FILEB.

If we begin to set up **FILEA** using LANSA it will not be possible to specify the validation rule that references FILEB because FILEB does not exist yet. Likewise the access route to FILEC cannot be specified because FILEC does not exist yet.

The chicken and egg situation is this: FILEA cannot be set up because it references FILEB ... and FILEB cannot be set up because it references FILEA. The solution:

- Define the fields and logical views of FILEA, FILEB and FILEC as required.
- Use the appropriate menu option to make the file definitions operational.
- Amend the definitions of FILEA, FILEB and FILEC. It should now be possible to specify the validation rules, access routes and batch control logic because all the files are operational.
- Use the appropriate menu option to make the amended file definitions operational. In all cases only the I/O modules will be recompiled because no actual changes to the fields in the file or the logical views have been made.

3.20.2 Where Did the Changes to the File Definition Go?

You may find when amending a file definition that the amendment appears to disappear or has been ignored.

This example is typical of this problem:

- A file named FILEB is to be defined. It is to contain a validation rule that references FILEA. However, this particular validation rule requires a special logical view of FILEA which currently does not exist.
- To set up the special logical view of FILEA which is required the definition of FILEA is amended. The new logical view, which is called FILEAV01, is successfully added to the definition of FILEA.
- The definition of new FILEB is now started. When attempting to add the validation rule that references FILEAV01 the system insists that no file with this name exists. Even using the option to choose the file from a list fails to show FILEAV01 in the list.
- What has happened to the amendment made to FILEA that defined a new logical file called FILEAV01?

The answer is that although the definition of FILEA was amended the definition was not made operational.

Although logical file FILEAV01 exists in the definition of FILEA it does not exist as an "accessible" file until the amended definition of FILEA has been made operational. This is an option on the file control menu.

Just remember: When any amendment is made to a file definition it is effectively "invisible" until the amended definition is made operational.

3.20.3 The Make New or Amended Definition Operational Job Failed

The most common causes of a "make new or amended file definition operational" job failure are described in detail below:

The Renamed "\$\$" File Has Not Been Deleted

When LANSA re-creates a file it renames the existing version of the file by appending the prefix "\$\$" to the name. Thus existing file CUSMST would be renamed \$\$CUSMST before the new version of CUSMST is created.

This will always work the first time that a file is re-created because no "\$\$" version exists. However, if the file definition is changed a second time, and nobody bothered to delete the "\$\$" version created during the first change, it will fail. This is because it will attempt to rename the existing file to a name that already exists.

When re-creating a file after amending its definition this is recommended:

- 1. Check that the new file has been re-loaded correctly (i.e. it actually contains the correct data).
- 2. Delete the "\$\$" version. Note that "\$\$" versions are only created when it is necessary to delete and re-create the physical file. Thus not all file re-creation jobs will actually produce a "\$\$" version of the file.

The File Cannot Be Allocated for Exclusive Use

The file re-creation job attempts to place an exclusive lock on the file being recreated. If any other user or job in the system has the file open then the recreation job will not be able to gain an exclusive lock and will fail. This situation is indicated clearly by messages on the job log.

In such cases request that all users of the file stop using it. This may mean waiting until batch jobs complete or asking interactive users to cease using certain processes. In cases where LANSA processes were created with the HEAVY usage option (see process definition for more details) the user may have to sign off and on again to actually close the file.

To find which users / jobs have a particular file open use the WRKOBJLCK (display object locks) command that is part of the IBM i operating system control language. Refer to the IBM supplied *Control Language Reference Manual* for more details.

Note: When a "make file definition operational" job fails, examine the associated job log carefully for problems such as these before contacting your product vendor. When examining a job log read it **backwards**. The message(s) associated with the problem that caused the job to fail will be one of the last messages issued. Messages at the beginning of the job log may appear to be the problem, but may in fact indicate normal situations that are catered for by LANSA.

4. Processes and Functions

You will find these topics described in this chapter:

- 4.1 What Is a Process and What Is a Function?
- 4.2 Process Control Menu
- 4.3 Select Processes to Be Worked With
- 4.4 Create a New Process Definition
- 4.5 Existing Process Definition Maintenance
- 4.6 Delete a Process Definition
- 4.7 Compile a Process from New or Amended Definition
- 4.8 Process Help Text
- 4.9 Print Process Definitions
- 4.11 Submit a Process to Batch
- 4.12 LANSA Debug Mode
- 4.13 Process Definition
- 4.14 Create a New Function
- 4.15 Edit a Function's RDML Commands
- 4.16 Rename a Function
- 4.17 Delete a Function
- 4.18 Function Control Table
- 4.19 Special Entries on the Process Menu
- 4.20 Multilingual Special Entries
- 4.21 Other Processes on the Process Menu
- 4.22 Process Parameters
- 4.23 Miscellaneous Process Details
- 4.24 Process Multilingual Attributes
- 4.25 The Action Bar Control Table

4.1 What Is a Process and What Is a Function?

A process is a group of related functions.

When used in an interactive environment a process appears to be a menu of associated functions.

When used in a batch environment a process usually consists of only one function or a group of closely related functions (such as related reports).

A function is an executable IBM i "program" that **was defined** by a user and created by LANSA.

A function usually manipulates the IBM i database in some way.

For instance an interactive process named ORDERS which has 4 functions associated with it can be visualized like this:



Some of the characteristics of processes and functions are:

- A function must "belong to" or be "owned by" a process
- Both processes and functions must have names. A process name must be unique within the entire LANSA partition. A function name must be unique within the process to which it belongs.
- Functions can be created automatically using an "Application Template". An application template asks the user a series of questions and then generates the function automatically.
- Functions can also be created manually. In this case the user must manually specify the commands that are to executed within the function.
- All functions contain a series of commands that define the "program" that is to be created. If the function was created automatically then the commands were created automatically, if the function was created manually then the commands must be entered manually.
- The ability to manually define and change the commands associated with a function is what gives LANSA its power, flexibility and speed in creating IBM i application programs.
- The commands within a function are called RDML commands where RDML is an acronym for "Rapid Development & Maintenance Language". RDML commands work at very high level. One LANSA RDML command may perform the same work as 100 or more commands in programming languages such as RPG, COBOL or PL/1.

The topics in this chapter describe how processes and functions are created and changed using the LANSA system.

4.2 Process Control Menu

Access to the process control components of LANSA is from the process control menu, which is accessed from the main system menu.



PppPROMENU

Process Control Menu

Enter number of function or place cursor on same line and press Enter.

- _ 1. Create a new process definition
- 2. Review or change a process definition
- 3. Delete a process definition
- 4. Compile a process from new or amended definition
- 5. Review, change or create process HELP text
- 6. Print process definitions
- 7. Use a process
- 8. Submit a process to batch
- 9. Turn LANSA debugging mode on

10. Turn LANSA debugging mode off 11. Exit from LANSA

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

4.3 Select Processes to Be Worked With

Many of the options that can be invoked from the process control menu will request the name(s) of the processes that you wish to work with (i.e. process to execute, review/change, compile, etc). In all cases the process selection is done in exactly the same way.

Initially a screen is presented that requests the name(s) of the processes to be worked with.

If you have been working with processes previously then the name will be prefilled with the name of the **last** process that was worked with.

SELPROCES	Select Process to Work with			
Enter full or partial name of the process(s) to be worked with or leave blank to select from all processes				
Fnn=Help Fnn=Exit	Fnn=Cancel Fnn=Messages			

Using this display it is possible to nominate the process(s) required in 3 different ways:

- **In full.** Enter the full name of the process required. If the process is found then the option chosen from the process control menu will be directly invoked for the process specified.
- **Partially.** Enter as much as desired of the partial name of the process. A search is made for all processes which generically match the name specified and the resulting list is displayed for selection.
- **Leave blank.** In this case a list of all processes defined to LANSA is displayed from which the desired process(s) can be selected.

When you press Enter, a list of processes is displayed.

SELPROCES S	elect Process to Work with				
Enter full or partial name of the process(s) to be worked with or leave blank to select from all processes					
Sel ProcessDescription. BOOKINGSSeminar Bookings System. MJDMJD Demonstration Process. ORDERSInquire and/or Maintain Company Orders. PKPROCName & Address Update					
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages					

Using this display it is possible to:

- Alter the process name at the top right of the screen. The change can be to specify a full process name, a partial process name or a change to blanks. In all cases another search is made for the process(s) required.
- **Select** one or more process(s) from the list by entering any non-blank character beside the process in the column labeled "Sel". If this is done then the option selected from the process control menu will be invoked for the process(s) selected.

4.4 Create a New Process Definition

The steps involved creating a new process definition are:



For a detailed description of the process definition menu and the input of each of the process and function components refer 4.13.1 The Process Definition Menu.

4.4.1 Specify the New Function's Details

When the option to create a new process definition has been chosen from the process control menu the *Create New Process Definition* screen is displayed. It requests that the name, description and other basic details of the process to be created are specified:

CRTPROCES	Create New Process Definition					
New process name Description						
Copy from process						
Initial public access <u>NORMAL</u> NORMAL, ALL, NONE Anticipated usage <u>LIGHT</u> HEAVY, LIGHT Process/menu style SAA/CUA R SAA/CUA, ACT/BAR, CURSOR, NUMBER, FUNCTION						
Exit/System funct ke Menu " " Messages " " Add " " Change " " Delete " "	y 1 < 2 : . 7 : These details do not appear 9 : in full SAA/CUA systems. . 10 : 11 <					
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages						

After details requested by this screen have been supplied the process definition menu will be displayed. This allows the specification of the various components of the process definition (i.e. functions required in the process, special entries on the menu, parameters, etc). The specification of these components is described in the following sections.

Input Options

New Process Name

Specifies the name by which the new process is to be known. Must be a valid object name. Refer to Valid Names on the IBM i in *Common Input Options*. Maximum length for a process name is 10 characters. Name specified must be unique within entire LANSA system. It is recommended that a naming standard be developed for process names.

Description

Specifies the description to be associated with the process. A brief description must be specified for every process created. The description aids other users of this process in identifying what it can be used for. See selecting required processes earlier in this chapter.

Copy from Process

Optionally specifies an existing process that is to be copied when creating this new process. This facility is useful when a process is to be changed. It can be copied to a new name, changed and tested without any disruption to users. A new process that is copied from an existing process is identical to the copied process except for details that are specified on this screen (i.e. name, description, etc).

Initial Public Access

Specifies what access other system users are to be given to the process initially. The access to the process can be changed at any time later using the option available on the housekeeping menu Mandatory. Pre-filled to NORMAL. Allowable values are:

NORMAL	Other users can use (execute) this process but they cannot modify its definition in any way.
ALL	Other users can use this process and can modify or even delete its definition.
NONE	Other users cannot use this process, nor can they modify its definition in any way.

Anticipated Usage

Specifies what amount of usage of the process is anticipated. This value can be changed later (see miscellaneous process details). Mandatory field. Pre-filled to LIGHT. Allowable values are:

- LIGHT Anticipated usage is LIGHT. The process will not be used repeatedly and continuously. Most processes are considered to be LIGHT usage.
- HEAVY Anticipated usage is heavy. The process will be used repeatedly and continuously. This option is normally only used in repetitive data entry applications.

In technical terms this value indicates whether or not the RPG programs created for the functions in this process should set on the LR (last record) indicator and close all files when terminating.

This value can be changed dynamically (without having to recompile any programs) so it may be worthwhile experimenting with it to modify system performance/throughput.

Process/Menu Style

Specifies the "style" of the process that is to be created. Pre-filled to SAA/CUA in an SAA/CUA partition, otherwise it is pre-filled to CURSOR. Allowable values are:

- SAA/CUA All menus and screen formats used by this process and any of its associated functions are to conform to the SAA (Systems Application Architecture) and/or CUA (Common User Access) standards defined for the partition in which the process is being defined. Refer to *Partition Definitions* in *SAA/CUA Implementation* in the *LANSA Application Design Guide* for more details of what the SAA/CUA standards are for a partition and how they apply.
- ACT/BAR The process is to act as an "Action bar" as defined by the CUA (Common User Access) standards defined by IBM and for this partition. To use this option these prerequisites must be satisfied:

- The current partition must be SAA/CUA enabled.

- You must be totally committed to the CUA 1989 standard for the "look" and "feel" of your application software.

Please read all relevant information in the LANSA Application Design Guide and in the IBM supplied CUA 1989 Basic Interface Design Guide.

Note: The following list of menu styles are now considered to be obsolete and

their use is not recommended for new systems. They are supplied purely to allow compatibility with existing systems.

- CURSOR Entries are to be selected from the process menu by positioning the screen cursor on the same line as the entry.
- NUMBER Each entry on the menu is to be assigned a number. An entry is selected by entering the number associated with the entry into a field at the bottom of the screen. This is the "traditional" style of menu and is the most commonly used in other IBM i application systems.
- FUNCTION Entries on the menu are to be displayed with their associated function name. An entry is selected by entering the name of the associated function into a field at the bottom of the screen. This style of menu processing is called "next function" processing and allows the users to go from function to function without having to return to the process menu. Refer to the 4.18 Function Control Table for more details.

Exit Function Key Delete Function Key

Specifies the assignment of function keys to functions within this process. The pre-filled values are the installation standards and correspond to the function key assignments within LANSA itself.

This information only appears on the screen when creating a new process in a non-SAA/CUA partition, or when specifically creating a non-SAA/CUA process. In an SAA/CUA style process all function key assignments exactly follow those defined for the associated partition.

If desired change the function key number assigned to the function. Function key number specified for any of the functions must be in the range 1 to 24. The same function key cannot be assigned to more than one function.

Function key assignments can be changed dynamically (without having to recompile any programs). Refer to 4.23 Miscellaneous Process Details for a description of how function key assignments can be changed.

4.5 Existing Process Definition Maintenance

The steps involved in the review or change of process definition are:



For a detailed description of the process definition menu and the review/change of each of the process and function components refer to later sections in this chapter.

4.6 Delete a Process Definition

The steps involved in the deletion of a process definition are:



4.6.1 Submit the Job to Delete a Process Definition

When the option to delete a process definition has been selected from the process control menu a screen similar to the following example will result:

Submit this job	<u>YES</u> YES, NO
Using Job name	<u>XXXXXXXXXX</u>
Job description	*LIBL/QBATCH
Job queue	*JOBD
Output queue	*LIBL/QPRINT

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

If the delete request is confirmed a batch job will be submitted to delete the process definition.

Note that the batch deletion job requires exclusive use of the process. If any other user or job is using the process the deletion job will fail and the process will not be deleted.

Messages on the job log clearly indicate that this situation has arisen. If required submit the deletion request again when no other user or job is using the process.

Input Options

Submit This Job?

Refer to Submit This Job? in Common Input Options.

Job Name Refer to Job Name. Job Description Refer to Job Description.

Job Queue Refer to Job Queue.

Output Queue

Refer to Output Queue.
4.7 Compile a Process from New or Amended Definition

In 4.1 What Is a Process and What Is a Function? the concept of a process and the functions within it were briefly described. Also mentioned was the fact that the functions within a process consist of a series of RDML commands. These commands are the "program" that defines to the IBM i exactly what is to be done by the function.

You may also remember that the RDML commands could come from 2 sources. They could be automatically generated by an Application Template or could be manually input by the user.

These commands cannot be executed by the IBM i until they are "compiled". This means that they are converted into a form that can be executed by the IBM i.

Most programming languages that run on the IBM i require that programs be compiled. If you have written RPG, COBOL or PL/I programs you will be familiar with the concept of compiling a program.

LANSA is no different. The RDML commands that make up each function in a process must be compiled into an IBM i executable form.

However, within LANSA the process is a little different. The following steps are performed:

- 1. The RDML commands associated with each function in the process are read and checked for errors.
- 2. If RDML commands are used that display information on a workstation, an IBM i "display file" will be designed and created. A "display file" is a special type of IBM i file that is used to read from and write to a workstation.
- 3. The RDML commands are then converted into an RPG program which is compiled just like any other RPG program.

Note: Any process (or more correctly, function within a process) must be "compiled" before it can be used. A function must be compiled when it is first created and must be re-compiled after it has been changed.

4.7.1 Steps to Compile a Process Definition

The steps involved in the compilation of a process from its new or amended definition are as follows:



4.7.2 Submit the Job to Compile a Process Definition

When the option to compile a process definition has been selected from the process control menu a screen similar to the following example will result:

DC@I	2300302	Compile	e / Re-Compil	e a Process	
Proces	s : XXXXXX	XXXX X	XXXXXXXX	XXXXXXXXX	xxxxxxxxxx
Submi Using Jol Ou Or Produc Optim Allow Dump Produc Genera Genera Sel – –	t this job Job name o description o queue itput queue ile the process ce RDML / RP ise/Ignore deci debug / Progra code generator ce Documentor ate HTML page ate HTML/Vali ate XML Function Us XXXXXXX XXXXXXX	as well as G and DI mal data mal data mobserv work are details . es / Valida date num able D XXX XXX	YES XXX XXX *LIBL *JOBD *LIBL functions below functions below Source listing errors rability erics erics XXXXXXX XXXXXXX XXXXXXX XXXXXXXX	YES, NO XXXXXXX /QBATCH /QPRINT ow . NO ngs NO / ngs NO / . NO / NO . YES / YES . NO YI YES YES YES / Y YES/NO YES, NO XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	YES, NO / NO YES, NO YES, NO S YES, NO ES, NO YES YES, NO YES, NO O XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

If the compile request is confirmed, a batch job will be submitted to compile the process definition.

Note that the batch compile job does not require exclusive use of the process for the entire time it is executing.

As each individual RDML function is compiled it is locked for exclusive use then released again when it has finished.

Similarly, the entire process is only locked if and when the process itself is to be compiled. It is unusual to compile processes in a development environment, so this should not be a problem.

When recompiling processes that use the usage HEAVY option (refer to 4.4.1 Specify the New Function's Details for more details), it may be necessary for any user of the process to sign off and sign on again to actually release all the locks held on the process.

This complication most commonly presents a problem in the following situation:

- User A is using process P (which has the usage HEAVY option) and decides to change it. The amendment is made and the job to recompile the process is submitted.
- The recompile job fails because it cannot allocate all components of process P for exclusive use.

Since user A was using process P he/she should have signed off and on again before submitting the recompile job.

If user A was accessing LANSA from command entry mode then an alternative to signing off and on again would be to exit from LANSA back to command entry mode, enter the command RCLRSC *CALLER, then invoke LANSA again to submit the recompile job.

Note: this complication only arises for processes that use the usage "HEAVY" option.

Input Options

Submit This Job?

Refer to Submit This Job? in Common Input Options.

Job Name

Refer to Job Name.

Job Description

Refer to Job Description.

Job Queue Refer to Job Queue.

Output Queue

Refer to Output Queue.

Compile the Process as well as Functions below?

Specifies whether or not the process should be turned into compiled form (in addition to any of the RDML functions selected from the list on the lower portion of the screen).

Any RDML function must be associated with one and only one process. The process is responsible for invoking and controlling each of its associated RDML functions.

It is not mandatory that a process be compiled. If it is not compiled an "interpretive" version of the process will be used at execution time. However, for better throughput it is possible to compile the process as well as its associated RDML functions.

The recommended procedure is to use the "interpretive" version of the process during application development. Just prior to placing the process into a production environment it should be compiled to gain the maximum throughput possible.

Mandatory field. Default value is determined from the system definition data area. Refer to The System Definition Data Area Overview. Allowable values are:

YES Compile the process as well as any selected RDML functions.

NO Do not compile the process. An "interpretive" version of the process will be used to control the associated RDML functions.

Produce RDML/RPG and DDS Source Listings

Specifies whether source listings should be produced by the batch compile job. Source listings produced include:

RDML

- A full listing of the RDML commands associated with each function.
- A representation of each screen format that was designed for each "RDML command" that displays information on a workstation.

Mandatory field. Default value is determined from the system definition data area. Refer to The System Definition Data Area Overview. Allowable values are:

YES Produce RDML source listings.

NO Do not produce RDML source listings.

RPG and DDS

- The DDS (data description specifications) used to create the display file associated with a function (if any).
- The compiler listing of the RPG program produced from the RDML commands associated with the function.

Mandatory field. Default value is NO. Allowable values are:

YES Produce RPG and DDS source listings.

NO Do not produce RPG and DDS source listings.

Note that if errors are found in RDML commands associated with a function, a source listing of them will be produced regardless of whether or not it was actually requested .

Optimize Compiled Programs

Specifies whether or not optimization should be used when compiling the RPG program that results from the "function control commands".

How this feature is implemented depends on the version of RPG code being compiled. For more information on the different versions of RPG code that can be compiled refer to ILE Implementation.

This facility is provided for compatibility with the IBM i. No claims are made regarding the effect of using (or not using) the optimization option. Refer to the IBM supplied CL reference manual for details. Using this option may increase the time taken to compile a process.

Mandatory field. Default value is determined from the system definition data area. Refer to The System Definition Data Area Overview . Allowable values are:

• If RPG/400 Code Is Being Compiled

YES Use the optimize option.

- NO Do not use the optimize option.
- If RPG/IV Code Is Being Compiled

```
YES Use the OPTIMIZE(*FULL) parameter.
```

NO Use the OPTIMIZE(*NONE) parameter.

Ignore Decimal Data Errors in Programs?

Specifies how decimal data errors should be dealt with in the compiled RPG program that results from the "function control commands".

How this feature is implemented depends on the version of RPG code being compiled. For more information on the different versions of RPG code that can be compiled refer to ILE Implementation.

This facility is provided for compatibility with the operating system and because it is required by some installations. It is strongly recommended that this option is **not** used. Refer to the IBM supplied CL reference manual for details.

Mandatory field. Default value is determined from the system definition data area. Refer to The System Definition Data Area Overview. Allowable values are:

• If RPG/400 Code Is Being Compiled

YES Use the IGNDECERR(*YES) parameter.

NO Use the IGNDECERR(*NO) parameter.

• If RPG/IV Code Is Being Compiled

YES Use the FIXNBR(*ZONED) parameter.

NO Use the FIXNBR(*NONE) parameter.

Allow Debug/Program Observability

Specifies whether the functions compiled from this process should be able to be used in DEBUG mode.

DEBUG mode operates at 2 levels within LANSA.

The first level is LANSA debug mode which allows debugging facilities to be used at the "RDML command level". Refer to a section later in this chapter for more details.

The second level is the debug facility provided by the operating system. This facility allows the RPG program generated by LANSA for a function to be debugged. Refer to the appropriate IBM supplied manuals for more details on how this level of debug is used.

If NO / NO is used all functions within this process cannot be used in debug mode at either level. The reason that debug mode is disabled is because the RPG

program will not contain symbol table and debug information. If debug is used on the program a message indicating that the program is not "observable" will result. By not including this information the size of the compiled RPG program is reduced by 40% to 50%, thus saving disk space and improving performance.

Mandatory fields. Default value is determined from the system definition data area. Refer to The System Definition Data Area Overview. Allowable values are:

YES /YES	Allow functions to be used in DEBUG mode/Do not remove program observability.
NO/NO	Do not allow functions to be used in DEBUG mode/ Remove program observability.
NO/YES	Do not allow functions to be used in DEBUG mode/Do not remove program observability.
YES /NO	Allow functions to be used in DEBUG mode/Remove program observability.

Dump Code Generator Work Areas?

Specifies whether the internal work areas used by the LANSA code generator should be dumped at the end of the compile job. This option would normally only be used when requested by your product vendor to aid in problem diagnosis or correction.

Mandatory field. Default value is NO. Allowable values are:

NO Do not dump code generator work areas.

YES Dump code generator work areas.

Produce Documentor Details?

Specifies if details for use in LANSA/DOCUMENTOR should be produced when this process and/or function(s) is compiled. This option will only appear if Documentor is enabled for the current partition.

If NO is specified, any previously produced Documentor details will be deleted for the process and/or function(s) being compiled.

Mandatory field. Default value is determined from the partition definition. Allowable values are:

NO Do not produce Documentor details.

YES Produce Documentor details.

Generate HTML Pages?

Specifies if HTML pages should be generated for DISPLAY/REQUEST/POP_UP commands when this function(s) is compiled. This option will only appear if the process is Web enabled.

Mandatory field. Default value is YES. Allowable values are :

- YES Generate HTML pages for each DISPLAY/REQUEST/POP_UP command in each function that is compiled.
- NO Do not generate HTML pages for each DISPLAY/REQUEST/POP_UP command in each function that is compiled.
 This value should be used with caution. It is generally only used when changes to the function have not included modification of any DISPLAY/REQUEST/POP_UP command. This allows HTML pages that have been modified using an HTML editor to be preserved.

Validate numerics?

Specifies if input numeric fields should be validated via JavaScript according to the allowable number of digits before and after the decimal point. This option is only applicable if the 'Generate HTML pages' option is 'YES'. This option will only appear if the process is Web enabled.

Mandatory field. Default value is NO unless the DC@OSVEROP data area contains *WEBNUMVAL. Allowable values are :

- NO Do not generate JavaScript function calls for input numeric fields for each function being compiled.
- YES Generate JavaScript function calls for input numeric fields for each function being compiled.

Generate XML?

Specifies if XML should be generated for DISPLAY/REQUEST/POP_UP commands when this function(s) is compiled. This option will only appear if the process is XML enabled.

Mandatory field. Default value is YES. Allowable values are:

YES Generate XML for each DISPLAY/REQUEST/POP_UP command in each function that is compiled.

NO Do not generate XML for each DISPLAY/REQUEST/POP_UP command in each function that is compiled.
 This value should be used with caution. It is generally only used when changes to the function have not included modification of any DISPLAY/REQUEST/POP_UP command. This allows XML that has been modified using the XML editor to be preserved.

Sel/Function/Usable/Description

This list of information that appears at the bottom of the screen details all of the RDML functions that are part of the process.

Note: When the advanced menus and an Impact List are being used, only functions which exist in the Impact List will be shown in the list of functions.

The function name is shown under the column "Function". The "Usable" column indicates whether or not the function is currently in a state in which it can be used (i.e. executed). The final column headed "Description" displays the user description associated with the function.

The most important column is that headed "Sel" (select). A non- blank entry against an RDML function in this column indicates that it should be selected for compilation (or re-compilation) by the batch job that is to be submitted.

Where a function is not currently usable the "Sel" value will be pre-filled with a "/" indicating that it will have to be compiled before it can be used.

If required, use the roll up and roll down keys to scroll through the list of all functions. Alter the "Sel" (select) values as desired to add to or remove from the list of functions that will be compiled (or re-compiled) by the batch job.

Note that each selected function is an independent RDML program and will be converted into an RPG program. If a large number of functions are selected for compilation (or re-compilation) then the batch job will accordingly take a long time to complete.

If the process is Web enabled, graphical HTML pages will be generated for screens in the function when the function is compiled.

4.8 Process Help Text

HELP text is information that is displayed to the user when the HELP key is used.

Generally HELP text has these characteristics:

- It is free format. No restrictions usually exist on the content or format of HELP text.
- It relates directly to the action the user was taking at the time the HELP key was pressed. Usually the process or function that the user is using is explained in some detail.
- Quite often the fields on the screen that was on display when the HELP key was pressed are each described in detail. Part of the description usually involves the name and description of the field, any validation rules that apply to its input and comments on the use and interpretation of the field's value.

LANSA automatically controls the handling of the HELP key. In addition LANSA will automatically determine the type of HELP that is required (field, process or function) and automatically display the associated HELP text (if any exists).

LANSA can dynamically, and in the correct language, create the HELP text associated with a field from the dictionary and the rules that it contains. You can turn off this automatic field level help text feature: globally, by field, or precede it with your own HELP text.

What LANSA cannot do is create the free format HELP text that is associated with the processes or functions. This is generally the task of the person who creates the process or function.

Once the HELP text has been entered, LANSA will automatically display it at the appropriate time.

The actual method used by LANSA to display HELP text is described in some detail in the *Messages and the Help Key*.

Refer to *Field Help Text* for information about how field HELP text is entered.

The following topics describe how HELP text is entered for processes or functions:

4.8.1 Steps to Create or Change Process HELP Text

4.8.2 Choose the Process or Function HELP Text

4.8.3 HELP Text Copy Options

- 4.8.4 Using the IBM Source Entry Utility SEU (Source Entry Utility)
- 4.8.5 Use Special Characters to Enhance HELP Text
- 4.8.6 Use Substitution/Control Values in HELP Text
- 4.8.7 Substitution/Control Values for Windows-specific HELP Text
- 4.8.8 Exit from the SEU Utility

4.8.1 Steps to Create or Change Process HELP Text

The steps involved in the review, change or creation of the HELP text associated with a process or function are as follows:



5. Review, change or create | HELP text for the process | or function chosen. Actual | text editing is done using | the IBM utility SEU. |____

SI	EU		
U7	TLIT	Y	
	-		

4.8.2 Choose the Process or Function HELP Text

If the option to "Create, review or change process HELP text" has been chosen from the process control menu and one or more processes are chosen using the method described earlier in this chapter then the Maintain Process HELP text screen will be presented.

DC@P100202	Maintain Process HELP text
Process : ORDERS	- INQUIRE AND/OR MAINTAIN COMPANY ORDE

Either enter YES to maintain HELP text associated with this process? **Or** select the function for which the HELP text is to be maintained

Function	Function description
ADDCUST	- Add new customer details to system
ADDORDS	 Add new order(s) to system
BRWCUST	- Browse / maintain customer master file
CUSTOMR	- Inquire about orders by customer number
INQUIRE	- Display full details of a specific order
PRINT -	Print details of an invoice
	Function ADDCUST ADDORDS BRWCUST CUSTOMR INQUIRE PRINT -

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The screen requests that either the HELP text associated with the process be chosen or the HELP text associated with one of the functions within the process be chosen for creation, review or change.

To choose the HELP text associated with the process enter YES at the top right of the screen.

To choose the HELP text associated with a function within the process select the function by entering any non-blank character beside it in the column labeled "Select".

Note: When the advanced menus and an Impact List are being used, only functions which exist in the Impact List will be shown in the list of functions.

4.8.3 HELP Text Copy Options

If no HELP text currently exists for the chosen process or function, then the Create HELP text by Copy screen will be presented.

DC@P100205 Create HELP text by Copy

Process ...: ORDERS- INQUIRE AND/OR MAINTAIN COMPANY ORFunction ...: INQUIRE- Display full details of a specific order

No HELP text currently exists for this FUNCTION **Optionally** use any one of the options below to initialise the HELP text

Either copy HELP text from field . . . _____Or copy HELP text from process . . _____Or copy HELP text from process . . _____function . . _____Or set up a standard HELP text form by entering YES ____

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Using this format it is possible to "initialize" the chosen process or function's HELP text by copying the HELP text from another field, process or function (within a process).

In addition it is possible to initialize the chosen process or function's HELP text to a "standard form".

The "standard form" that comes with the shipped version of LANSA can be changed. Refer to Formats, Values and Codes in the *Technical Reference Guide* for details.

If you use a "standard form" for the help, you can alter it, using this procedure:

- 1. Sign on as the LANSA security officer and invoke LANSA.
- 2. Review or change the HELP text associated with field STDFORM.

4.8.4 Using the IBM Source Entry Utility SEU (Source Entry Utility)

If HELP text exists for the process or function, or after completion of input on the HELP text copy options screen, a screen similar to the example following will be displayed:

Columns . . . : 1 71 Edit QTEMP/HELPWORK Find ... **HELPWORK** FMT ** ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 . 0001.00 \$\$LANGUAGE=ENG 0002.00 0003.00 This field represents the {employee number} assigned to the 0004.00 employee by this company. Each and every employee must have 0005.00 an employee number. On any screen where you are asked to 0006.00 specify an employee number you can press the prompt key on 0007.00 employee number to cause a pop-up window to be displayed. 0008.00 You can then search by employee name and select the required 0009.00 employee number.

F3=Exit	F4=Prompt	F5=Refresh
F10=Top	F11=Bottom	F24=More keys

The HELP text input screen is presented by the SEU (Source Entry Utility). SEU is provided by IBM and is used to create and manipulate source or text. SEU is widely used by IBM i programmers and other users. . If you are unfamiliar with the features and functions of SEU refer to the appropriate IBM manual.

In addition, SEU has online HELP facilities of its own. Once you have started to use SEU press the HELP key and review the extensive HELP text available.

Using SEU the HELP text associated with the process or function can be reviewed, created or changed. All the features of SEU are available and can be used.

If SEU is not available on your IBM i machine, the help text will be edited by EDTF command. Refer to help on this command for features. On exit from the EDTF you will be asked if you wish any changes made to the LANSA help text to be copied to the LANSA files.

4.8.5 Use Special Characters to Enhance HELP Text

Some special features are provided by LANSA to enhance the display of HELP text. These consist of special characters that can be used as follows:

Special Character Effect

%(percentage)	Display in high intensity
{(left parenthesis)	Underline
@(at)	Display in reverse image
~(accent)	Blink
<pre>}(right parenthesis)</pre>	Revert to normal display mode
\(backslash)	Revert to normal display mode

Some examples of using these special characters are:

HELP	Text Input	Using SEU	Displayed By	LANSA As
	1	0	1 5 5	

the%item number\must	the item number#\$ must
the{item number}must	the \$@item number@\$ must
%{VALIDATION RULES\	\$%VALIDATION RULES%\$
%{VALIDATION RULES}	\$%VALIDATION RULES%\$
{%VALIDATION RULES}	\$%VALIDATION RULES%\$

Where multiple enhancement characters are used together with no spaces between them they are "consolidated" into one enhancement character. For instance "%{@" (high intensity, underline, blink) would be consolidated into one enhancement character that is used in the same position as the "@". The "%" and "{" are replaced by blanks. This explains why in the last example above the underline ("{") attribute starts under the "V" in "VALIDATION" and not one character before it.

Note that a display enhancement cannot be continued over multiple lines. Display enhancements are considered to end at the end of the text line, even if a "revert to normal display" character ("\" or "}") is not used:

```
contact the{IT department or
```

your data entry supervisor for more information. would be displayed as: contact the \$@IT department or_____your data entry supervisor for more information. @\$

4.8.6 Use Substitution/Control Values in HELP Text

LANSA provides a set of "substitution" and "control" values that be included into HELP text to control and enhance its display to the user.

HELP text substitution/control values are substituted/actioned at execution time (i.e. when the HELP text is being actively displayed to a user).

Substitution Value	Description, Effects And Comments
\$\$PAGE	Causes the string "Page: 999" to be inserted into the text. Use to indicate the current HELP text page number to the user.
\$\$NEWPAGE	Causes a new HELP text page to be started. A HELP text page is 18 lines long by default. The line containing the \$\$NEWPAGE value is included into the displayed HELP text. The \$\$NEWPAGE value is replaced by blanks.
\$\$TITLE XXXXXXXX	Indicates to LANSA the title that should be associated with the HELP text. Up to 40 characters of title can be specified. LANSA will automatically centre the title and convert it to uppercase. The line containing the \$\$TITLE value is NOT included into the displayed help text.
\$\$FLDNAM	Causes the name of the current field to be inserted into the HELP text. Use this value in field level HELP text only.
\$\$FLDDES	Causes the description of the current field to be inserted into the HELP text. Use this value in field level HELP text only.
\$\$PRONAM	Causes the name of the current process to be inserted into the HELP text. Use this value in process or function level HELP text only.
\$\$PRODES	Causes the description of the current process to be inserted into the HELP text. Use this value in process or function level HELP text only.

\$\$FUNNAM	Causes the name of the current function to be inserted into the HELP text. Use this value in function level HELP text only.
\$\$FUNDES	Causes the description of the current function to be inserted into the HELP text. Use this value in function level HELP text only.
\$\$RIGHT	Causes all manually defined help text to be right aligned when displayed. This support is provided for bi-directional languages.
\$\$NOAUTO	Indicates that automatically generated help text should not be created for this field. Use this option in field level help text only.
\$\$LANGUAGE=XXXX	Delimits the boundary between the help text associated with different languages when working in a multilingual partition. These values are automatically created when editing help text and should not be altered in any way or the help text associated with a language may be corrupted.
\$\$TECH	Indicates that the following text is technical help text. This help will only be displayed to a developer within LANSA.
\$\$USER	Indicates that the following text is user help text. This help will always be displayed within LANSA. Help text will default to user help text if no \$\$TECH value is used.
\$\$HI=	Causes the special character assigned for high intensity display to be re-assigned to a user defined character.
\$\$RI=	Causes the special character assigned for reverse image display to be re-assigned to a user defined character.
\$\$BL=	Causes the special character assigned for blink display to be reassigned to a user defined character.

\$\$UL=	Causes the special character assigned for underline display to be re assigned to a user defined character
\$\$N1=	Causes the special character assigned for normal display one to be re-assigned to a user defined character.
\$\$N2=	Causes the special character assigned for normal display two to be re-assigned to a user defined character

4.8.7 Substitution/Control Values for Windows-specific HELP Text

These help text substitutions are used to link help for other objects to the automatically generated Index (& Contents) for the current object. They do not produce visible entries in the generated help text.

Note that the Contents are only produced for the new help interface provided with Windows.

Substitution Value	Description, Effects and Comments
\$\$INDEXFLD = <field-name></field-name>	Causes an item to be included in the Index that will display help text for the designated field. The rest of the current line is ignored. The label used in the Index is the field's description.
<pre>\$\$INDEXCOM = <object- name="">, <component- name=""></component-></object-></pre>	Causes an item to be included in the Index that will display help text for the designated component. The component is specified using the owning object (e.g. form) and component names, separated by a comma. The rest of the current line is ignored. The label used in the Index is the component's name. If no component name is specified then the index item will display help for the object (e.g. form) itself.
\$\$INDEXPRO = <process- name></process- 	Causes an item to be included in the Index that will display help text for the designated process. The rest of the current line is ignored. The label used in the Index is the process's description.
\$\$INDEXFUN = <process- name>, <function- name></function- </process- 	Causes an item to be included in the Index that will display help text for the designated function. The function is specified using the owning process and function names, separated by a comma. The rest of the current line is ignored. The label used in the index is the function's description.
\$\$ROOT	Causes any following Index substitutions (\$\$INDEX) to be added to the root of the Contents tree-view. Useful for index items that might be applicable to all or a number of

topics.

\$\$LEAF Causes any following Index substitutions (\$\$INDEX...) to be added below the current topic in the Contents tree-view. This is the default value and only needs to be specified to switch back after \$\$ROOT has been used.

\$\$IMAGE = A bitmap (*.BMP) image file to be included in the help text <filename>, at this point. The image is centered in the screen. The default path is the current partition's Execute directory, but any path may be specified. Note: Universal Naming Convention (UNC) paths are not supported. The alternate text is used instead of the image in an interface where images are not displayed, such as a 5250 terminal. If the alternate text is not specified then the filename is used instead.

4.8.8 Exit from the SEU Utility

Once finished with the process's HELP text use function key 1 to exit from SEU. A screen like this will be presented:

Exit

Type choices, press Enter.

Change/create member <u>N</u> Member HELPW File HELPWOR Library QTEMP Text	Y=Yes, N=No YORK Name RK Name Name
Resequence member Y Start 0001.00 Increment 01.00	Y=Yes, N=No 0000.01 - 9999.99 00.01 - 99.99
Print member <u>N</u>	Y=Yes, N=No
Return to editing $\dots \dots N$	Y=Yes, N=No
Go to member list $\dots \dots N$	Y=Yes, N=No
F3=Exit F5=Refresh F12=Can	cel

The SEU exit option will be pre-filled to 1 (exit without update) if you did not change the field HELP text in any way. Otherwise it will be pre-filled to 2 (exit and update member).

Normally you would only ever use option 1 or option 2 to exit from SEU and return to LANSA. Option 6 may be used to return to SEU and continue editing. Other options can be used if desired, but you may find that any changes made are lost and the process or function's HELP text remains unchanged.

Note that the member, file and library always contain HELPWORK,

HELPWORK and QTEMP respectively. This applies no matter what process or function's HELP text is being edited. This is a special temporary source file that LANSA has created in library QTEMP. These values can be changed if required, but again you may find that any changes are lost and the process or function's HELP text remains unchanged.

If SEU is not available on your IBM i machine, the help text will be edited by EDTF. When finished editing the help text use function key 2 or 3 to save any changes made. Use function key 3 to exit from EDTF.

You will be asked: Do you wish to save the changes made to HELPWORK? Y/N.

Reply Y to save changes to the LANSA Help files.

4.9 Print Process Definitions

The steps involved in printing process definitions are as follows:



4.9.1 Submit the Print Job to Batch

When the option to print process definitions is taken, the *Print Process Definitions* screen (from which a batch job to perform the actual process/function printing can be submitted) is displayed:

LANSA/XXX	Print Process Definitions	
Submit this job	<u>YES</u> YES, NO	
Using Job name	PROCESSPRT	
Job description	*LIBL/QBATCH	
Job queue	*JOBD	
Output queue	*LIBL/QPRINT	
Print HELP text for pro	ocess NO YES, NO	
Print HELP text for fur	nctions NO YES, NO	
Print fields used by functions NO YES, NO		
Print files used by functions NO YES, NO		
Print RDML for functions / Include audit stamps . YES / <u>NO</u> YES, NO		
Sel Process/Function	Description	
PSLSYS	Personnel System Main Menu	
<u>X</u> EMPLIST	Full Employee Listing	
<u>X</u> ENROL	Enrol a New Employee	
<u>X</u> INQUIRE	Browse/Maintain Employee and Skill Files	
<u>X</u> PHONE	Employee Business Phone Number List +	

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Input Options

Submit This Job?

Refer to Submit This Job? in Common Input Options.

Job Name

Refer to Job Name.

Job Description

Refer to Job Description.

Job Queue

Refer to Job Queue.

Output Queue

Refer to Output Queue.

Print HELP Text for Process

Specifies whether the HELP text associated with the processes should be printed. Mandatory. Pre-filled to NO.

Print HELP Text for Functions

Specifies whether the HELP text associated with the functions should be printed. Mandatory. Pre-filled to NO.

Print Fields Used by Functions

Specifies whether all fields used by each function within the processes should be printed. Mandatory. Pre-filled to NO.

Print Files Used by Functions

Specifies whether files used by each function within the processes should be printed. Mandatory. Pre-filled to NO.

Print RDML for Functions / Include audit stamps

Specifies whether the RDML for each function within the processes should be printed. Mandatory. Pre-filled to YES.

If the RDML for each function is to be printed, you can specify whether the RDML audit stamps are to be included. Include audit stamps is ignored if RDML is not to be printed.

Printing of the HELP text, fields, files and RDML can substantially increase both the size of the report and the length of time taken to produce it.

Select Functions within Each Process

A list of all functions for each previously selected process will be displayed for selection. All functions are pre-selected with a "/" to indicate they are to be printed. Blank out the "Sel" column for any functions that are not to be printed.

Note: When the advanced menus and an Impact List are being used, only functions which exist in the Impact List will be shown in the list of functions.

4.10 Use/Execute a Process from a Workstation

- 4.10.1 Steps to Using/Executing a Process
- 4.10.2 Specify Process Parameters
- 4.10.3 The Process Main Menu
- 4.10.4 Using a Process or Function from Your Own Application

4.10.1 Steps to Using/Executing a Process

The steps involved using/executing a process from a workstation are as follows:



|_____| |_____|

4.10.2 Specify Process Parameters

If a process has been selected for use and it has been defined with one or more parameters a display similar to the example following will result:

EXAMPLE Print Ledger Account History

Process : EXAMPLE - Print Ledger Account History

DescriptionValueValue
Company number :
Account number :

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

For more information on how process parameters are specified refer to later sections in this chapter.

The use of process parameters with processes that are designed to be used exclusively from a workstation is not recommended. Process parameters are primarily intended as a means of passing information into processes that are designed to run in batch. See the 4.11 Submit a Process to Batch for more details.

Input Options

The following input options apply to specifying process parameter values when using a process:

Description

This is an output field. It shows the description associated with the process parameter. For more information about defining process parameters refer to later sections in this chapter.

Value

Specifies the value that is to be placed into the associated process parameter. The value specified can only be an alphanumeric literal or a numeric literal. If an alphanumeric literal is specified it should NOT be enclosed in quotes or the quotes will be mapped into the parameter as well.

Values specified are checked for compatibility with the definition of the associated process parameter. The compatibility check validates type, length and number of decimal positions.
4.10.3 The Process Main Menu

Whenever a process is used from a workstation the process main menu will be displayed.

Working from the process main menu any of the associated functions can be used.

The layout of the process main menu varies according to 3 factors:

- The number and description of the functions associated with the process.
- The "menu style" that has been defined for the process. For more specific details about the menu style options refer to creating a new process and miscellaneous process details in this chapter.
- Any "additional menu options" that have been defined for the process. For more specific details about additional menu options refer to later sections in this chapter.

Some example process main menus follow:

Example 1

A process named ORDERS which uses process menu style SAA/CUA and has quite a few associated functions might look like this:

ORDERS Inquire and/or Maintain Company Orders

Enter number of function or place cursor on same line and press enter:

- _ 1. Add new customer details to system
 - 2. Add new order(s) to system
 - 3. Browse / maintain customer master file
 - 4. Inquire about orders by customer number
 - 5. Display full details of a specific order
 - 6. Print details of an invoice
 - 7. Display process or function HELP text
 - 8. Exit from system

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

NOTE: This first example menu is in SAA/CUA format, which is the RECOMMENDED STYLE for all newly developed LANSA applications. Refer to the *LANSA Application Design Guide* for details of *SAA/CUA Implementation* and why it is an important strategic direction for your company.

Example 2

A process named BOOKINGS which uses menu style NUMBER and has several associated functions might look like this:



Example 3

A process named EXAMPLE which uses menu style FUNCTION and has only one associated function might look like this. Note that menu entries C1, C2 and C3 are all "additional menu options" that have been defined for this process.

Select function required by entering function name

ADDRECS - Add records to the name & address file

C1 - Display workstation message queue

C2 - Display job submitted from workstation

C3 - Display jobs submitted by user

HELP - Display process or function HELP text

MENU - Display process main menu

SELECT - Select from list of available functions

EXIT - Exit from system

Next function : MENU - Display process main menu Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

4.10.4 Using a Process or Function from Your Own Application

This section has so far described how to use a process from within LANSA. However, it is possible to use a process or function directly from your own application program or from command entry.

The direct use of a process is achieved by specifying parameters on the LANSA command or when calling the LANSA program.

Accessing LANSA described how LANSA can be invoked by executing the command:

LANSA

or by calling LANSA:

CALL LANSA

To use a process called ORDERS directly from your own application, without going though the normal LANSA menus, the following command could be used:

LANSA RUN ORDERS

or by calling LANSA:

CALL LANSA ('RUN' 'ORDERS')

To use a function called ADDCUST (which is part of process ORDERS) directly from your own application, without going though the normal LANSA menus, or even the process main menu, the following command could be used:

LANSA RUN ORDERS ADDCUST

or by calling LANSA:

CALL LANSA ('RUN' 'ORDERS' 'ADDCUST')

Either the LANSA command or the CALL command can of course be compiled into your own application programs and the name of the process and optional function to be run can be passed as a variable instead of a literal.

The LANSA or CALL commands can be executed from an interactive application or from a batch application.

Additional parameters on the LANSA command and the LANSA program allow the passing of parameters from your application program into the process and/or the nomination of the system partition to be used.

The LANSA Command Parameters

The LANSA command has the following command parameters - all of which

are optional:

Keyword	Description / Comments	
REQUEST	Request made to LANSA. Must be either RUN or SUBMIT	
PROCESS	Name of process to RUN or SUBMIT.	
FUNCTION	Name of function to RUN.	
PARM01	User parameter 1. Must exactly match parameters defined for the process in type and length. Note that numeric process parameters are always "packed" numeric variables.	
PARM02 - PARM10	User parameters 2 through 10. A process can have at most 10 associated parameters defined. The type and length rules as for PARM01 apply.	
PARTITION	Name of the system partition that is to be used. If not specified or passed as blanks, the major application partition "SYS" will be assumed.	
LANGUAGE	Language that the system is to use. If not specified or passed as blanks the partition default language will be assumed.	
TASK ID	Task Id that is to be used. This should normally be passed as blanks.	
PC_TYPE	This parameter is used internally by LANSA and is not usable in a normal LANSA invocation. This is 1 character in length and should be passed as N.	
DEVELOPER	Used to specify whether LANSA presents a command line and work with interface. It is up to 9 characters in length and should be passed as N. For further information, refer to the LANSA Command.	
ALLOW_MSGS	Specifies whether or not access is given to developer services. It is up to 4 characters in length and should be passed as N.	

	For further information, refer to the LANSA Command.
DATE_SRCE	Specifies where the date and date format is retrieved from. It is up to 7 characters in length.
	*SYSTEM , S : Specifies that the date and date format are to be retrieved from the system values. This is the default.
	*JOB , J : Specifies that the date and date format are to be retrieved from the job attributes.
BDEBUG	Specifies that batch debug of LANSA RDML functions is required. This is used internally by LANSA and is not useable in a normal LANSA invocation. It should be passed as blanks.
BDEBUG_DEV	Name of the display device to be used for debugging batch RDML functions. This is used internally by LANSA and is not useable in a normal LANSA invocation. It should be passed as blanks.
BDEBUG_MSG	Name of the message queue when using batch debugging. This is used internally by LANSA and is not useable in a normal LANSA invocation. It should be passed as blanks.
X_RUNADPRM	Additional parameters for an X_RUN request in up to 512 characters. These additional parameters must be specified in the X_RUN format, e.g. PRTR=*PATH. Refer to Standard X_RUN Parameters in the <i>Technical Reference Guide</i> for details of X_RUN parameters.

The LANSA Program Parameters

The LANSA program has the following parameter specifications. It is not necessary to pass all the parameters to the LANSA program, only those up to and including the last one that you wish to use:

Number	Туре	Length	Description / Comments

1	A	10	Request made to LANSA. Must be either MENU or RUN
2	А	10	Name of process to RUN or SUBMIT.
3	А	10	Name of function to RUN.
4	any	256(max)	User parameter 1. Must exactly match parameters defined for the process in type and length. Note that numeric process parameters are always "packed" numeric variables.
5 - 13	any	256(max)	User parameters 2 through 10. A process can have at most 10 associated parameters defined. The same type and length rules as for PARM01 apply.
14	A	3	Name of the system partition that is to be used. If not specified or passed as blanks the major application partition "SYS" will be assumed.
15	A	4	Language that system is to use. If not specified or passed as blanks the partition default language will be assumed.
16	А	8	Task Id that is to be used. This should normally be passed as blanks.
17	A	1	The PC_TYPE parameter is used internally by LANSA and is not usable in a normal LANSA invocation. It should be passed as N.
18	A	1	This parameter is used to specify whether LANSA presents a command line and work with interface. Should be passed as N. For further information, refer to the LANSA Command.
19	A	1	This ALLOW_MSGS parameter is used to specify whether or not access is given to

			developer services. It is not usable in a normal LANSA invocation and should be passed as N.For further information, refer to the LANSA Command.
20	A	10	This PC_NAME parameter is used internally by LANSA and is not usable in a normal LANSA invocation. It should be set to blanks.
21	A	1	 This parameter is used to specify where the date and date format is retrieved from. S specifies that the date and date format are to be retrieved from the system values. This is the default. J specifies that the date and date format are
22	A	1	To be retrieved from the job attributes.This parameter is used to specify that batch debug of LANSA RDML functions is required. It is used internally by LANSA and is not useable in a normal LANSA invocation. It should be passed as blanks.
23	A	10	This parameter is used to specify the name of the display device to be used for debugging batch RDML functions. It is used internally by LANSA and is not useable in a normal LANSA invocation. It should be passed as blanks.
24	A	10	This parameter is used to specify the name of the message queue when using batch debugging. This is used internally by LANSA and is not useable in a normal LANSA invocation.
25	A	512	Specify additional parameters for an X_RUN request. These additional parameters must be specified in the X_RUN

form	mat, e.g. PRTR=*PATH. Refer to
Star	ndard X_RUN Parameters in the
Tec	<i>hnical Reference Guide</i> for details of
X_1	RUN parameters.

Error Messages for LANSA

Refer to Escape messages

4.11 Submit a Process to Batch

The term Batch Processing refers to the process of executing a series of noninteractive jobs all at one time.

Usually, batch jobs are stored up during working hours and then executed during the evening or whenever the computer is idle. Batch processing is particularly useful for operations that require the computer or a peripheral device for an extended period of time. Once a batch job begins, it continues until it is done or until an error occurs. Note that batch processing implies that there is no interaction with the user while the program is being executed.

The opposite of batch processing is transaction processing or interactive processing. In interactive processing, the application responds to commands as soon as you enter them.

4.11.1 Steps to Submitting a Process to Batch

The steps involved in submitting a process to batch for execution are as follows:



4.11.2 Specify Process Parameters

If a process that is to be submitted to batch has been defined with one or more parameters a display similar to the example following will result:

EXAMPLE Print Ledger Account History

Process : EXAMPLE - Print Ledger Account History

DescriptionValueValue	
Company number :	
Account number :	

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

For more information on how process parameters are specified refer to later sections in this chapter.

The use of process parameters with processes that are designed to be used exclusively from a workstation is not recommended. Process parameters are primarily intended as a means of passing information into processes that are designed to run in batch. See the following section for more details.

Input Options

The following input options apply to specifying process parameter values when submitting a process to batch:

Description

This is an output field. It shows the description associated with the process parameter. For more information about defining process parameters refer to later sections in this chapter.

Value

Specifies the value that is to be placed into the associated process parameter. The value specified can only be an alphanumeric literal or a numeric literal. If an alphanumeric literal is specified it should NOT be enclosed in quotes or the quotes will be mapped into the parameter as well.

Values specified are checked for compatibility with the definition of the associated process parameter. The compatibility check validates type, length and number of decimal positions.

4.11.3 Submit the Process to Batch

When the option to submit a process to batch has been used a screen similar to that following is always the last displayed before the job is submitted:

EXAMPLE	Print L	edger Account	History
Process : EXAMPL	E -	Print Ledger	Account History
Submit this job	••••	<u>YES</u>	S YES, NO
Using Job name Job description . Job queue Output queue	· · · · · · · · · · · · · · · · · · ·	EX *LII *JOB *LII	XAMPLE BL/QBATCH D BL/QPRINT

Select functions to be executed :

Sel Function Usable Description

	XXXXXXX	XXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
_	XXXXXXX	XXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
_	XXXXXXX	XXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen requests that the submit request be confirmed and allows the job name, job description, job queue and output queue to be changed from their default values.

In addition the individual functions that make up the process are displayed. All functions that are to be executed in batch should be chosen. Each function chosen results in the submission of one batch job. Thus if six functions are chosen, six batch jobs will be submitted.

There is no harm in submitting a function to batch that attempts to "talk" to a workstation. In such cases the function will abort during initialization and

messages on the job log will indicate the cause of the problem.

Generally processes and/or functions designed to run in batch have a different form to those designed to run interactively (i.e. from a workstation).

Input Options

The following input options apply when specifying details of a process that is to be submitted to batch:

Submit This Process to Batch?

Refer to Submit This Job? in Common Input Options.

Job Name

Refer to Job Name.

Job Description

Refer to Job Description.

Job Queue

Refer to Job Queue.

Output Queue

Refer to Output Queue.

Sel/Function/Usable/Description

This list displays details of all functions that are associated with the process and are directly accessible from the process main menu (refer to the 4.18 Function Control Table for more details of excluding functions from the process main menu).

The function name and description are displayed along with an indication of whether the function is currently usable (i.e. compiled and ready to be executed). Normally a non-usable function would only be submitted if the compile job has been previously submitted but had not completed execution yet.

Where more functions exist than can be displayed, use the roll keys to scroll backwards and forwards through the list of all functions.

A function is selected for batch execution by entering any non-blank character beside it in the column headed "Sel". Each function selected for submission to batch results in one batch job being submitted. Thus if 6 functions are chosen 6 batch jobs will be submitted.

The function chosen becomes the "initial" function that is executed in the batch job that is submitted. Whether or not it is the only function that is executed

depends upon how the function is written (e.g.: uses TRANSFER or CALL commands) and of course on the function control table entries associated with the function.

4.11.4 Submit a Process to Batch Directly from Your Own Application

This section has so far described how to submit a process to batch from within LANSA. It is possible to submit a batch process directly from your own application program or from command entry.

The direct submission of a process is achieved by specifying parameters on the LANSA command or when calling the LANSA program.

Accessing LANSA described how LANSA could be invoked by be executing the command:

LANSA

or by calling LANSA:

CALL LANSA

To submit a process called HISTORY to batch directly from your own application, without going though the normal LANSA menus, the following command could be used:

LANSA SUBMIT HISTORY

or by calling LANSA:

CALL LANSA ('SUBMIT' 'HISTORY')

Either the LANSA command or the CALL command can be compiled into your own application programs and the name of the process to be run can be passed as a variable instead of a literal.

The LANSA or CALL commands can be executed from an interactive application or from a batch application.

Additional parameters on the LANSA command and the LANSA program allow the passing of parameters from your application program into the process as well. This a very useful facility because the parameters are submitted with the process. Refer to the LANSA Command.

4.12 LANSA Debug Mode

The IBM i operating system contains a very powerful facility for locating logic errors in programs. It is generally referred to as "DEBUG" and when it is being used on a program the program is said to be in "DEBUG mode".

The most powerful feature of DEBUG is that it allows an executing program to be stopped at any point and the content of fields used by the program to be displayed or changed.

By using this facility, logic errors in programs can be quickly and easily located. In fact some people feel that it is this facility that enables programmers on the IBM i to achieve such high rates of productivity.

4.12.1 Using IBM i Debug Mode

The IBM i operating system DEBUG facility can be used on any function compiled by LANSA in exactly the same way as it would be used on a normal application program. The only restriction is that the function must have been compiled with debug "enabled". Refer to the section on compiling processes for more information about this option.

To use IBM i DEBUG on a function compiled by LANSA it will also be necessary to have a source listing of the associated RPG program. Refer to the section on compiling processes for details of how this is obtained.

From the RPG listing of the program associated with a function, two pieces of information are required to successfully use IBM i DEBUG:

- The first is the name of the program and the library in which it resides. The program name is internally generated and has nothing to do with the function name. These are required to enable you to place the program in debug mode.
- The second is the RPG source statement numbers. These are required to enable you to specify when the program should stop at a "breakpoint" to display or change the content of fields.

It is beyond the scope of this guide to describe the IBM i DEBUG facilities in any detail. Refer to the appropriate IBM supplied reference manual for specific information on how to use them.

4.12.2 About LANSA Debug Mode

As mentioned already the RPG program that was compiled by LANSA from the RDML commands can be debugged using the CPF or IBM i DEBUG facility.

However, LANSA provides a debugging facility that is an extension of the IBM i DEBUG facilities and is much easier to use. The reasons that it is much easier to use are:

- It operates at the RDML level. The actual RDML commands are debugged and not the resulting RPG statements (of which there may be hundreds and they may appear quite cryptic).
- No knowledge of the IBM i debug facility, program names, source statements numbers or field names is required. Using simple displays you can choose the statements and the fields required.

LANSA Debug allows three types of Debugging facilities:

- **1. DEBUG Interactively.** This is commonly known as a 'Breakpoint' facility. This allows RDML commands to be selected to halt execution of the function prior to executing the selected statement(s) and in addition allows selection of the Function variables to be displayed with their current values in Character, Zone and Decimal format.
- 2. TRACE All/Selective statements. This visually displays via a printed report the sequence of RDML commands executed within the function. It allows RDML commands to be selected to halt execution of the function prior to executing the selected statement(s) and in addition allows selection of the Function variables to be printed with their current values in Character, Zone and Decimal format. When the function completes a message is issued informing that the TRACE report is available to be displayed or printed.
- **3. COUNT statements.** This will count the number of times each RDML command is executed within the function and produce a printed report detailing the RDML commands in the function and the count value for each RDML command. This acts as a useful cross-reference to ensure that RDML commands are being executed within the relevant logic structure of the function. It will certainly help in discarding redundant code and to highlight performance issues. When the function completes a message is issued informing that the COUNT report is available to be displayed or printed.

4.12.3 Turn LANSA Debug Mode ON

The first thing that must be done to use LANSA debugging mode is to turn it on. To do this, on the Work with Processes screen, select F10=Other, to display the LANSA Debug On/Off popup box. Select Turn LANSA debugging mode on.

On the resulting display you may optionally nominate the name of the process and/or function that is to be debugged.

Note: Avoid using IBM i and LANSA debug mode at the same time. Attempting to use them together will not do any harm but they may "interfere" with one another and not work correctly.

The default names for the process and function to be debugged are *FIRST / *FIRST which indicate that the first function in the first process you intend to use should be debugged.

Generally you only ever need to nominate a specific process and/or function when it is going to be called by the process/function you intend to run first (i.e. you are attempting to debug a "subroutine" function).

After selecting the process/function to be debugged, the system should respond with a message indicating that debugging mode has been turned on. If debugging mode is already on the message will indicate this fact instead.

If Function Routing is enabled, the routed "to" function will be executed. It is this "to" function which must be specified for debug.

4.12.4 LANSA Debug in Batch

LANSA functions may also be debugged while executing in batch.

You must first turn on LANSA debugging mode from the Process Control Menu. After this, any subsequent process or function that is submitted to batch via the Submit process to batch menu option or the SUBMIT RDML command can then be debugged interactively on another display device.

Once the batch job begins execution, a message will be sent to the display device from which the LANSA job was originally submitted. The response to this message may be one of these:

<display device name></display 	the name of the display device to be used for interactive debugging of the batch LANSA function. Note that the nominated display device must be signed off before it can be used for interactive batch debugging.
<cancel></cancel>	to terminate the batch job immediately
<nodebug></nodebug>	indicates that batch debugging of the LANSA function is not required. The function will continue to execute as though debugging had not been turned on.
<wait></wait>	to cause the job to wait for 5 minutes before sending the same message again.

The default reply after 120 seconds is to cancel.

Once an available display device has been nominated, this display device will be used for the interactive debugging of the LANSA job with the same interaction as an interactive LANSA job detailed in *Using LANSA Debugging Mode*.

4.12.5 Use LANSA Debug Mode

Once LANSA debug mode has been turned on it can be used to aid in the location and correction of logic errors in the RDML commands that define what processing a function should perform.

To illustrate the use of LANSA debugging mode a process called EXAMPLE that has one function called ADDRECS is used. Function ADDRECS is to be debugged.

The RDML commands that define function ADDRECS follow and are used to add records to a name and address file named DEMNAME:

Seq	Lbl	Command	Parameters
1		GROUP_BY	NAME(#NAMESLIST)
			FIELDS(#DEMNAC #DEMNAM #DEMAD1 #DEMAD2 #DEMAD3 #DEMPCD)
3		SET_MODE	TO(*ADD)
4	L1	CHANGE	FIELD(#NAMESLIST)TO(*DEFAULT)
5		DISPLAY	FIELDS(#NAMESLIST)
6		INSERT	FIELDS(#NAMESLIST) TO_FILE(DEMNAME)
7		GOTO	LABEL(L1)

Debugging the RDML commands in ADDRECS is described in detail in the following steps.

Step 1

Turn LANSA debugging mode on. This has already been described.

Step 2

Select the option to use process EXAMPLE. This is exactly the same as electing to use process EXAMPLE in the normal way. This results in the menu associated with process EXAMPLE being displayed. Function ADDRECS is the first entry on the menu.

EXAMPLE DEBUG EXAMPLE PROCESS

Enter number of function or place cursor on same line and press enter

- _ 1. Add records to the name & address file
- 2. Display process or function HELP text
- 3. Display process main menu
- 4. Select from list of available functions
- 5. Exit from system

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Step 3

Elect to use function ADDRECS from the display in Step 2. This is exactly the same as electing to use function ADDRECS in the normal way. The resulting display is the first "interruption" to the normal flow of events.

This screen states that function ADDRECS is in debug mode and requests for the debug option to be selected. The DEBUG options have been previously described in this chapter.

If Options 1 or 2 were selected then the following displays will be shown requesting the relevant selections to be made. If Option 3 was selected then the function will immediately start to execute and act as if it is not in DEBUG mode i.e. no DEBUG interaction.

Step 4

For Options 1 and 2 only, the resulting display is the second "interruption" to the normal flow of events.

```
F@DEBUG02 Add Records to the Name and Address File
                              Mode: DIS
Function ADDRECS is in debug mode. Choose from list below the command
To be halted just prior to execution for display of debug information.
Position at Seq . . . .
Find . . .
Debg Sq Lbl Command Parameters
     GROUP BY NAME(#NAMESLIST) FIELDS(#DEMNAC #DEMN
1
                     #DEMAD2 #DEMAD3 #DEMPCD)
_ 3
      SET MODE TO(*ADD)
 _ 4 L1 CHANGE FIELD(#NAMESLIST) TO(*DEFAULT)
_ 5
    DISPLAY FIELDS(#NAMESLIST)
X 6 INSERT FIELDS(#NAMESLIST) TO FILE(DEMNAME)
_ 7
      GOTO LABEL(L1)
F16=F/Fwd F17=F/Bkwd F20=Select All
```

This screen states that function ADDRECS is in debug mode and requests that the RDML commands that are to be halted before execution are selected. Note: On this screen it is possible to:

- Find a particular sequence number. To do this, enter the sequence number required in the 'Position at Seq' field and press enter. If the sequence number entered is beyond the range of sequence number, the last sequence number will be displayed.
- Find forwards. To find a particular string of characters enter them into the 'Find' field and select command key 16. If the string is found, the cursor will position on the first character. Please note that words split over lines will not be found.
- Find backwards. To find a particular string of characters in the RDML prior to where the cursor is currently, enter them into the 'Find' field and select command key 17. If the string is found, the cursor will position on the first character. Please note that words split over lines will not be found.
- Select all RDML lines. To select all the available RDML statements for breakpoint, use command key 20. The Select Variables screen will then be displayed.
- When RDML statements are selected, they will automatically be retained, along with any selected variables. When the function is run again under debug mode, the previous RDML statements selected will be indicated as selected. This will occur within the duration of a job. The retained breakpoints will be lost after signing off and also if the function is recompiled since they were selected.

In this case statement 6 has been chosen. DEBUG will stop the function just before executing statement 6. (i.e. just before attempting to insert a new record into the name and address file).

Step 5

For Options 1 and 2 only, the resulting display is the third "interruption" to the normal flow of events.

F@DEBUG02 Add Records to the Name and Address File Mode: DIS Select Function Variables to be Displayed/Printed at Breakpoints		
Search for Variat	ble :	
Act Field RPG M <u>X</u> DEMAD1 D	Name Description EMAD1 Address line 1	Type Len Dec A 25 0

X DEMAD2 DEMAD2 Address line 2	A 25 0		
_ DEMAD3 DEMAD3 Address line 3	A 25 0		
X DEMNAC DEMNAC Name Code	A 60		
_ DEMNAM DEMNAM Name	A 25 0		
_ DEMPCD DEMPCD Post Code	A 40		
_ IO\$STS IO\$STS LANSA data base operation	on sts cde A 2 0		

This screen states that function ADDRECS is in debug mode and requests that the Function variables (if any) that are to be Displayed or printed at an execution halt is to be selected.

Note: On this screen it is possible to:

- search for a variable. To do this, enter the required variable name into the 'Search for Variable' field and press Enter.
- Select all variables. To select all the available variables for display at breakpoint, use command key 20. The next screen will then be displayed.

If no RDML commands were selected in Step 4 then this display will not be shown.

In this case the Function variables DEMAD1, DEMAD2 and DEMNAC have been chosen to be displayed/printed at execution halt, in this example the variables will be displayed/printed just prior to the execution of statement 6.

Step 6

Function ADDRECS now begins to execute. Statements 3 and 4 are executed. When statement 5 is executed the normal display results.

ADDRECSAdd Records to the Name and Address FileName codeA00001NameACME ENGINEERING AND COAddress line1121 AUSTRALIA STREET

Address line 2NEWTOWNAddress line 3SYDNEY, AUSTRALIAPost code2042

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Into this screen, details of a new name and address are entered.

Step 7

When Enter is pressed in the preceding step function ADDRECS completes execution of statement 5. However, since debug is on it should halt before executing statement 6. If Option 1 from Step 3 was selected then the following debug display results otherwise the function continues processing.

F@DEBUG02	Add Rec	cords to the Name ar Mode: DIS	nd Addres	s File
Seq Lbl Comma	nd Paran	neters		
4 L1 CHANGE	E FIELI	D(#NAMESLIST) 7	O(*DEF	AULT)
5 DISPLAY	FIELDS	(#NAMESLIST)	,	~
6 INSERT	FIELDS(WAMESLIST) TO	FILE(DI	EMNAME)
	× ×	· · ·	- 、	,
Search For Variable	5:			
Enter action: $D = nc$	ormal disp	$\overline{\text{lay H}} = \text{hex display}$	P = Print	C = change
Act Field name RP	G name ¹	Description	Type I	Len Dec
DEMAD1 DE	EMAD1	Address line 1	Â	25 0
– C: 12	1 AUSTR	ALIA STREET		
Z: FF	FFCEEEI	DCDCCFEEDCCE		
D: 12	10142391	3910239553		
DEMAD2 DE	EMAD2	Address line 2	А	25 0
– C: NI	EWTOWN	1		
Z: D0	CEEDED			
D: 55	63665			
DEMNAC D	EMNAC	Name code	А	60
C: A()0001			

Z: CFFFFF D: 100001

F10=Services F11=Step Next F15=Vars Y/N F19=Hex Yes/No

Some points to note about this display are:

- The statement about to be executed is indicated at the top of the screen and is highlighted. It is possible to roll through the top display of three RDML lines to show statements which come before and after the current statement.
- The bottom half of the screen displays all the fields that were selected from Step 5. Shown are the LANSA field name, the RPG field name, the description, Value of the fields (in Character, Zone and Decimal formats), type and length. If the LANSA field name is more than 6 characters long it will be different to the RPG field name. LANSA internally renames fields that are more than 6 characters long so that the RPG program will compile.
- The "Search for Variable" allows input for a variable name to be searched for against the list of variables that were requested to be displayed from Step 5. The Search if found will position the Cursor against that variable or in the case of a Generic search the cursor will be positioned against the first variable meeting the Generic search criteria.
- The "Act" column allows a number of actions to be taken on any of the fields displayed. These are:
 - D Display the contents of the field.
 - H Display the contents of the field in hexadecimal form.
 - P Print the contents of the field.
 - C Change the contents of the field.
- The function key "F10=Services" allows the various Debug service options to be actioned from a POP UP style services menu. This will be explained in Step 8.
- The function key "F11=Step Next" allows the execution of the current statement and then halt at the next statement to be executed.

- The function key "F15=Vars Y/N" allows the display to be shown with:
 - variables and only three lines of RDML or
 - without variables but a full screen of RDML lines.
- The function key "F19=Hex Yes/No" allows the display to be shown with hex values or without hex values.

When enter is pressed on this screen and assuming that no extra statements were added to halt execution via the "Edit Services Menu", the function ADDRECS will execute statement 6 and statement 7 which causes statement 4 to be executed again. When statement 5 is executed the display described in Step 6 will result again.

So the function continues to execute - stopping whenever requested to display or change the contents of fields.

Step 8

If Function key 10 is used the "Edit Services" Menu will be displayed.

```
F@DEBUG02
                 Add Records to the Name and Address File
                              Mode: DIS
  Seq Lbl Command Parameters
   4 L1 CHANGE
                  FIELD(#NAMESLIST) TO(*DEFAULT)
   5
      DISPLAY
                FIELDS(#NAMESLIST)
                FIELDS(#NAMESLIST) TO_FILE(DEMNAME)
   6
      INSERT
Search F .....
Enter ac : F@DEBUG02 Add Records to the name and Address File : e
Act Field :
                                   : Dec
_ DEMAD : Select Debug Service Option and press Enter
                                                    : 0
    :
    : _ 1. Review/Change Debug statements
                                             :
    : 2. Remove ALL Debug statements
                                             :
_ DEMAD : 3. Review/Change Variables to display
                                                  : 0
    : 4. Terminate Function
                                :
    : F3=Exit Services
                                     : 0
DEMNA:
```

Z: CFFFFF D: 100001

F10=Services F11=Step Next F15=Vars Y/N F19=Hex Yes/No

The various options are:

- Review/Change Debug statements. This allows the review of statements that have been selected to halt the function to either add further statements or remove current statements. This process and the resulting display is identical to Step 4.
- Remove ALL Debug statements. This will automatically remove all statements that have been selected from Step 4 which will allow the function to continue processing without being halted.
- Review/Change Variables to display. This allows the review of Function variables that have been selected to be displayed at selected debug statements. This process and resulting display is identical to Step 5.
- Terminate Function.

This allows the function to be terminated and return control to the current process menu. Function key 3 will Exit from the "Services Menu" and return to the display as in Step 7.

The "Edit Services" menu is only ever available when the debug mode is on, debug option is DEBUG Interactively and when the function is at a debug statement halt.

Step 9

The final step in this example is to turn debug mode off and to display or print the Debug TRACE or COUNT report if selected.

4.12.6 Turn LANSA Debugging Mode OFF

When you have completed using LANSA DEBUG mode it should be turned off. To do this select the option on the process control menu to turn LANSA debugging mode off.

The system should respond with a message indicating that debugging mode has been turned off. If debugging mode was not on the message will indicate this fact instead.

If you forget to turn LANSA debugging mode off do not worry. Debugging mode is automatically turned off whenever you exit from LANSA in a normal manner.

4.13 Process Definition

The process definition menu is displayed whenever the options "Create a new process definition" or "Review or change a process definition" are chosen from the process control menu.

The method of accessing the process definition menu can be visualized like this:



|_____|

4.13.1 The Process Definition Menu

When displayed, the process definition menu looks like this:

pppPROREV Process Definition (xxxxx)

Enter number of function or place cursor on same line and press Enter.

- ____ 1. Create a new function
 - 2. Review or change a function's RDML commands
 - 3. Delete a function
 - 4. Review or change the function control table
 - 5. Review or change special entries on process menu
 - 6. Review or change multilingual special entries
 - 7. Review or change other processes on process menu
 - 8. Review or change process parameters
 - 9. Review or change miscellaneous process details
- 10. Review or change process multilingual attributes
- 11. Review or change the action bar control table
- 12. Exit from LANSA

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note: "(xxxxx)" on first line indicates Create or Review mode.

Working from this menu it is possible to define the various components of a new process or review/change the various components of an existing process. Each of the entries on this menu and their use are described in detail in following sections in this chapter.

4.13.2 Exit from the Process Definition Menu (Function Status)

Whenever the Cancel function key is used to exit from the process definition menu (and return to the process control menu) a function status display similar to the following example will be result:

STSFUNCTN Status of RDML Functions

Currently Function Use Description and messages ADDCUST YES Add new customer details to system BRWCUST NO Browse / change customer details The RDML commands have been manually changed. Compile/recompile this function before attempting to use ADDORD NO Add details of a new order to the system The RDML commands have been manually changed. Compile/recompile this function before attempting to use BRWORD YES Bowse order details

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen displays the current status of all functions defined within the process that was being worked on from the process definition menu.

The screen is presented as a "wrap up screen" when you have completed the specification of a new process or reviewed/changed an existing process. Details presented include:

- The name and description of every function defined in the process.
- A "currently usable" indicator. If this is set to YES then the function is currently in a usable state and can be executed whenever desired. If this is set to NO then some action(s) will have to be taken before attempting to execute the function. Usually the actions required are detailed in the associated messages section.
• Message(s) that indicate changes made to the function and any action(s) that are required before the function can be used again.

On the example screen shown it can be seen that functions BRWCUST and ADDORD have had their associated RDML programs modified. As such they must be re-compiled before they can be used. Functions ADDCUST and BRWORD have not been changed at all and can still be used.

4.14 Create a New Function

The following diagram visualizes the steps required to create a new function within a process:



1	1		
		1 1	

4.14.1 Specify Details of the New Function

When the option to create a new function is chosen from the process definition menu a screen similar to the example following will result:

DC@P301001	Define New Function
Process : XXXXXXXX	XXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
New function name Description	••
New function type 2. Copy 3. Manu	1. Generate from an Application Template versiting ProcessFunctionually define function
Initial public access	. <u>NORMAL</u>
Fnn=Help Fnn=Exit Fi	nn=Cancel Fnn=Messages

As soon as the details on this screen have been specified and the enter key pressed the function is created.

What happens next depends upon whether the function is to be

"AUTOMATICALLY" defined by an Application Template (function type 1) or "MANUALLY" defined by copying an existing function (function type 2) or by manually inputting the RDML commands (function type 3).

If a "manually" defined function is requested then RDML command editor will be invoked next. This facility allows RDML commands to edited and is described in much more detail in following sections.

If an "automatically" defined function is chosen then "Application Template" facility will be invoked next. This allows you to choose a "template", which will probably ask you questions, and using your answers, automatically generate the

RDML commands your require for your function. This facility is described in more detail later.

Input Options

The following input options apply when creating a new function within a process:

New Function Name

Specify the name that is to be assigned to the new function. The maximum length allowed for a function name is 7 characters. The name must also be a valid name as specified in Valid Names on the IBM i in *Common Input Options*. The name specified must also be unique within the process.

Certain function names are "reserved" for use by LANSA and cannot be specified as a valid function name. These are:

Name	Reserved Meaning / Description
MENU	Display process main menu
EXIT	Exit from LANSA
HELP	Display process HELP text
SELECT	Select next function from list of allowable function
EOJ	End all batch processing
ERROR	Abort process with an error
RETRN	Return control to calling process or function
*ANY	Any function name

For more details of why these names are "reserved" refer to 4.18 Function Control Table.

Description

Specify the description that is to be associated with the function. The entry made here is the description that is displayed on the process main menu. Mandatory field. Function descriptions can be changed by altering the process control table. Refer to later sections in this chapter for details of the process

control table.

Since this value will appear on the process menu it is recommended that upper and lower case characters are used. There is no need to centre the description. This is done automatically.

New Function Type

Select the way that the RDML commands associated with this new function are to be defined by entering the number corresponding to:

- 1 Generate from an Application Template.
- 2 Copy existing Process_Function_.
- 3 Manually define function

Initial Public Access

Specifies what access other system users are to be given to the function initially. Access to the function can be changed at any time later using the option available on the housekeeping menu Mandatory (if displayed). Pre-filled to NORMAL. Allowable values are:

- NORMAL Other users can use (execute) this function but they cannot modify its definition in any way.
- ALL Other users can use this function and can modify or even delete its definition.
- NONE Other users cannot use this function, nor can they modify its definition in any way.

Note: this option will only appear on the screen if the LANSA system being used has a flag set in the system definition data area DC@A01 to indicate that function level security should be implemented. Refer to the System Definition Data Areas DC@A01 for more details.

4.14.2 When You Use the Application Template Facility

When you create a new RDML function and specify a new function type of 1 (Generate from an Application Template) then the Application Template Facility will be invoked.

The Application Template Facility will initially place a "pop-up" window over the lower portion of the screen that you are currently using.

DC@P301001 Define New Function

Process : CUSTOMERS Work with Customers

New function name: CUST01Description: Enter Details of New Customers

New function type : 1 1. Generate from an Application Template

: DC@P309601 Application Template Selection

: Place cursor on same line as required template and press ENTER :

: Template Application Template Description

: ADD01 Data entry application - style 1 (simple)

: ADD02 Data entry application - style 2 (with confirmation):

```
: ADD03 Data entry application - style 3 (batched input) :
```

- : BROWSE01 Browse data style 1 (simple | single file)
- : BROWSE02 Browse data style 2 (simple | multi-file)

: INQUIRE01 Simple inquire - style 1 (flat screen display)

: INQUIRE02 Simple inquire - style 2 (with browse area as well) :

```
More ... :
```

:

:Enter Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Msgs Fnn=ChgSeq Fnn=Filter:

:....:

This pop up window is displaying a list of available Application Templates.

From this pop up window you can elect to:

- Select an Application Template to use. Position the screen cursor anywhere on the same line as the template required and press enter. This will cause the selected template to be invoked.
- Scroll the list of templates. More templates probably exist than can be displayed in one page of the pop up window. To view the others, use the roll up or roll down keys to scroll backwards and forwards through the list of Application Templates.

Note that only the application templates that you are authorized to use are actually displayed to you.

- Use the *ChgSeq* (Change Sequence) function key to toggle the template list between template display sequence and template name sequence.
- The *Filter* function key allows you to enter a full or partial template name. Only templates matching the entered value will be displayed. A filter value of blank will result in all templates being shown.
- **Get a lot more information about a template.** The short description of the template that is shown in the window only gives you a vague idea what sort of RDML function it would generate for you.

Most well constructed templates have a lot more information contained in their associated HELP panels (including fully worked examples of the type of RDML applications they produce).

To view these HELP panels, place the screen cursor on the same line as the Application Template you are interested in and press the HELP function key.

There is not a lot more that can be said about using Application Templates, because they are most often site defined, and even those that are shipped with the product have probably been modified to match your site's exact requirements.

However, some tips in using them are as follows:

- Make extensive use of the HELP function key and take your time. When using a template for the first time, read all the HELP panels associated with the template, especially the examples. Get a good idea of what the template will do and what the template won't do before you use it.
- Most templates work using a "question and answer" session. So after selecting a template to be used, a question will most likely appear in the pop up window area.
- If no question appears, and the "Process Definition Menu" is re-displayed,

you can assume that the template has generated your RDML program without having to ask you any questions at all. Some very simple templates work this way (e.g.: the one that generates a basic program layout).

- When a question appears, read the question very carefully, then read any additional prompting information very carefully, and finally, if you still have any doubts use the HELP function key.
- Follow instructions to the letter. The concept of an application template is to generate RDML programs for you in a very quick and very consistent matter. The generalized nature of the Application Template facility and its ability to be site definable, mean that absolutely precise validation of your answers is often not practicable.

Incorrect answers will cause no real problems, other than to cause RDML code that will either not compile, or not execute correctly, to be generated. This is not a real problem, but it means that you will most probably have to then fix it by manual editing of the RDML code.

• Most templates allow you to "back up" to a previously answered question and change your answer. To do this use the Cancel function key. Do not do this if the prompt indicates you should not do it.

4.15 Edit a Function's RDML Commands

This section of the user guide is intended to describe how RDML commands are input, reviewed or changed (i.e. edited).

It is NOT intended that this section describes:

- The RDML commands in detail.
- How RDML programs are written and the techniques involved.

For detailed information about the RDML commands themselves, writing RDML programs and RDML programming techniques, refer to the *Technical Reference Guide*.

4.15.1 What are RDML Commands?

As already described in this chapter, every function within LANSA has associated with it a set of commands.

The commands instruct the IBM i to what actions the user wishes to take when executing the function.

The commands are referred to as Rapid Development & Maintenance Language or "RDML" commands.

When a function is compiled into an executable program the commands are converted in RPG instructions which are then compiled into a program that can be executed on the IBM i.

Every command has 3 major components:

- A label or tag. This is a name associated with a particular command that allows control to be passed to it. For instance the command GOTO LABEL(L10) transfers control to the command that has label L10.
- The name of the command itself. This specifies exactly which command is to be executed.
- Additional parameters. These are specific to each type of command. For instance the GOTO command has parameter LABEL, which specifies the label to which control is to be passed. When a parameter on a command is not specified, it usually adopts a default value.

Whenever RDML commands are displayed they usually have the format described above. For instance a simple "function" (or "program") of RDML commands might look like this:

Label Command Additional Parameters

L10	REQUEST	FIELDS(#ORDERNO #PRODUCT #QUANTITY)
	INSERT	FIELDS(#ORDERNO #PRODUCT #QUANTITY) TO_FILE(ORDERS)
	GOTO	LABEL(L10)

What this RDML program does is:

• Request that the user input (at the workstation) an order number, a product

number and a quantity.

- Adds a record to file ORDERS using the values just input be the user.
- Transfer control back to the first command in this program.

The first question that springs to mind when looking at this RDML program is "How does it ever end?".

To answer that you must always remember that parameters that are not specified adopt their default values.

The REQUEST command has parameters EXIT_KEY and MENU_KEY which have default values EXIT_KEY(*YES *EXIT) and MENU_KEY(*YES *MENU).

This tells LANSA that the EXIT and MENU (or Cancel) function keys are to be enabled at the workstation. If the EXIT function key is used: exit from LANSA, if the MENU (or Cancel) function key is used: transfer back to the process's main menu.

This has been a very brief description of what RDML commands are and what RDML programs look like, the rest of this section describes topics like:

- How an RDML program such as this could be manually input.
- How the RDML program could be fully checked for errors before it is submitted for compilation.
- How the screen design facility could be used to modify any screen layouts used in the RDML program.
- How the report design facility could be used to modify any report layouts used in the RDML program.

For a complete list of RDML commands refer to the *Technical Reference Guide*. In the *Technical Reference Guide* you will find details of each RDML command and RDML programming techniques and examples.

4.15.2 Automatic Functions, Manual Functions and their RDML Commands

This chapter has already described how LANSA works with 2 different types of functions.

- The first is the "automatically" defined function. These have been automatically generated by the Application Template Facility.
- The second is the "manually" defined function. The RDML commands associated with a manually defined function are input directly by the user.





For either "automatic" or "manual" functions, it is the associated RDML commands that are actually compiled in an executable program. In addition, the RDML program associated with an automatic or a "manual" function can be reviewed and amended using the facilities described in this section.

4.15.3 Steps to Editing RDML Commands

The steps involved in editing the RDML commands associated with a function are as follows:



facility (if desired). | DESIGN | | DESIGN | | FACILITY | | FACILITY | |_____|

If the option to review or change the RDML commands associated with a function is chosen from the process or function definition menus, a screen similar to this example will result:

DC@P309901 Display RDML Commands
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2 REQUEST FIELDS (#ORDNO #PRODNO #QUANTITY) 3 INSERT FIELDS (#ORDNO #PRODNO #QUANTITY) TO_FILE(4 END_LOOP ************************************
Fnn=Help F5=W/C/P F6=Start F7=End F8=Jump F9=Audit F11=Tag Fnn=C F13=Top Fnn=Messages F16=F/Fwd F17=F/Bkwd F20=Indent F21=Change

Working from the Display RDML Commands screen it is possible to:

Review the RDML program associated with a function

Use the ROLL keys to roll backwards and forwards through the RDML program.

Change the RDML program using the LANSA editor

If the name of the editor shown at the top right of the screen is not LANSA change it and press enter. Then use the CHANGE function key to indicate that you wish change (edit) the RDML program. Using the LANSA editor is

described in more detail in the following sections.

Change the RDML program using the IBM SEU editor

If the name of the editor shown at the top right of the screen is not SEU, change it and press enter. Then use the CHANGE function key to indicate that you wish to change (edit) the RDML program. Use of the SEU editor is described in more detail in the following sections.

Run the function checker

Enter YES, LIST or DESIGN into the field at the top of the screen. Enter NO to avoid running the function checker. If YES or LIST are entered a full check of all commands in the function will be performed. Additionally, if LIST is entered, an indented listing of the function and any screen layouts will be printed. If DESIGN is entered only commands that relate to the layout of screen formats and/or report formats will be checked.

Run the screen design facility

If the function checker has been run and no errors were found it is possible to run the screen design facility. Enter SCREEN into the field which will appear at the top of the screen to run the screen design facility.

Run the report design facility

If the function checker has been run and no errors were found it is possible to run the report design facility. Enter REPORT into the field which will appear at the top of the screen to run the report design facility.

Find some text string within the function

Use function key 16 to scan forwards. Use function key 17 to scan backwards.

In both cases, the scan starts from the position directly adjacent to the current location of the screen cursor.

The scan facility also supports word wrapping over two lines, the use of "?" as a "wild card" in the scan string, and ignores upper and lowercase differences.

Position a line to the top of the display

To position any line to the top of the display, move the screen cursor onto the line and use function key 13. "Zoomed" lines cannot be positioned to the top of the display.

Display the RDML program in color coded format

Use function key 19 to turn color coding off or on. You must be on a properly configured color device for function key 19 to be enabled. This facility is best

used in conjunction with the indent function key.

Display the RDML function in an indented format

Use function key 20 to turn indenting mode off or on. The indented format displayed is similar to that produced in hardcopy reports that list RDML functions.

Ask "What is xxxx?" or use the "Cut and Paste" facility

To do this, position the screen cursor anywhere in the parameter area on a full (or partial) field or file name and use function key 5.

W/C/P stands for "What is xxxx? / Cut / Paste" and it is a very specialized function key that is designed to be used in two different ways:

To answer the question "What is XXXXX?" when browsing:

While browsing an RDML function you will often come across field and file names that you need further information about.

To get additional information about a field or file simply position the screen cursor on the field or file name and use the F5=W/C/P function key. The resulting pop-up window will show summary details of the object and then allow you to use function key 8 to display more detailed information.

To allow "Cut and Paste" operations when defining RDML logic:

While specifying RDML commands you may need to "remember" the name of a field or file. The F5=W/C/P function key may be used to aid in doing this on an ad-hoc" basis (to build lists of fields it is easier to use the normal "F" prompt facility).

As an example of using the W/C/P facility, imagine you have specified this much of the following command:

SELECT FIELDS(#SURNAME #FIRSTNAME #EM_

and then need to "remember" whether "Employee Number" is EMPNO or EMPNUM. To do this, use F5, position the cursor onto the correct name (EMPNUM, say) and then use F5 again. The selected value will be "pasted" back into your command:

SELECT FIELDS(#SURNAME #FIRSTNAME #EMPNUM

Similarly you may not know which view of a file called PSLMST is to be used on a FETCH command:

FETCH FIELDS(.....) FROM_FILE(PSL_

Using F5=W/C/P with a full or partial name you can build a list of the physical and logical files, examine their attributes and key details, and finally "cut and

paste" a selected name back into the RDML editor:

FETCH FIELDS(.....) FROM_FILE(PSLMST06

You can also "cut and paste" field names from within file details. For example you may wish to select "Employee Number" but have no idea of its name. However you do know that it comes from the PSLMST file, so you could type the following and then use the F5 function key:

SELECT FIELDS(#SURNAME #FIRSTNAME #PSLMST_

and then use F8 to show a list of all the fields in the PSLMST file. You may then position the screen cursor over "Employee Number" and use F5 to "paste" the name. This would alter your RDML command to:

SELECT FIELDS(#SURNAME #FIRSTNAME #EMPNUM

Some important points about the F5=W/C/P function key:

- The W/C/P facility determines the "search name" by searching backwards from the current screen cursor location for a non-blank character and then by attempting to "pick out" a name at the cursor position. The name location facility will handle a name that wraps around onto a following line (provided the line is on the current display page) but it will not handle wrap around onto a preceding line.
- When the W/C/P window first appears a summary of all objects generically matching the search criteria is shown. To view more details of an object place the screen cursor on it and use function key 8.
- When any W/C/P window is shown you can "cut and paste" the name of an object back into the RDML editor by placing the screen cursor on it and using function key 5.
- The "cut and paste" facility is very simple. It maps the name of the selected object back into the RDML function editor. It does not compress or expand commands. Only the original search string and any blanks to the right of it will be overlaid by the "cut and paste" string.
- The displayed pop-up window will be located as close as possible to the position of the cursor at the time that F5=W/C/P function key was used.
- The W/C/P window is deliberately made to be very small. It is a simple development aid and not designed to be a substitute for other major online inquiries or reports.
- All searches are terminated after 100 items have been selected. If the required object cannot be found, exit from the W/C/P window and then re-invoke it with a more precise search name.

• The W/C/P window will include details of function defined fields and data dictionary overridden fields. However the details shown for such fields are as they were at the time of the last full or partial function check or compile.

Jump to the start of the RDML function

Use function key 6 to jump to the start of the RDML function.

Jump to the end of the RDML function

Use function key 7 to jump to the end of the RDML function.

Jump to a pre-determined point within the function

Use function key 8 to invoke the jump screen panel. When it appears use the screen cursor to select to jump directly to:

- The start of the function.
- The end of the function.
- To any subroutine defined within the function. .
- To any line within the function that has been "tagged". (See the next point for details of how lines are tagged).

Tag a line to allow direct jumping to it

Use function key 11 when the screen cursor is positioned on an RDML line. The line will be "tagged" so that the next time that the "Jump" function key (see previous point) is used you may elect to jump directly to it.

Up to 50 lines may be tagged. To "un-tag" lines that have been previously tagged, use F8 (Jump) and then repeatedly use F11 (and the screen cursor) to select lines that are to be dropped from the list of tagged lines (i.e. to be "untagged").

Using F11 is a way of "temporary" tagging. The tagging is temporary because the tag is only remembered until you exit from the editor. Details of individual temporary tags are also dropped if you delete the tagged line, prompt it by using F4, or cause the entire function to be (re)loaded by performing some action such as editing via the SEU editor, screen painting or report painting.

However, there is a way to apply "permanent" tags to lines within the RDML function. Whenever an RDML comment line is processed by the editor the first 3 characters of the line are examined. If they contain "TAG", "Tag" or "tag" then the line is automatically considered to be a tagged line and loaded into the tagged list.

This means that a permanent tagging method may be established to allowing direct jumping to important sections of functions.

Zoom Back and In on the details of the function

To use the "zoom" keys, you must first use the "Indent" function key to place the display in an indented mode.

Repeatedly use function key 23 to "Zoom Back" and show less and less of the "details" of the current function.

Repeatedly use function key 24 to "Zoom In" and show more and more of the "details" of the current function.

By showing less of the "details" on the display, you can examine the overall structure of the function, or parts of the function.

This allows you to check the structure more easily, and also to move more quickly through the structure until the part that you are interested in is visible.

Once it has been found, repeatedly use the "Zoom In" function key to show more and more details of the program structure.

When scrolling through the function, look for the "zoom light" in the upper left of the screen panel. It indicates that something on the current display is in "zoom mode", and therefore not shown in full detail.

Display the RDML Audit Information

Use function key 9 to toggle the display of the audit date, user and, if used, the task identifier of RDML commands, overlaying the right hand portion of the RDML parameters. The audit date is displayed in installation date format.

DC@P309901	Display RDML Commands
XXXXXXXXXXX XXXX	XXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Invoke function checker.	YES, LIST, DESIGN, COMP, NO
No Act Lbl Command	Parameters Date User Task Id
******** Beginni	ng of RDML commands **************
1 BEGIN_LOOP	

2 REQUEST FIELDS (#ORDNO #PRODNO yyyy/mm/dd USERXX)

3 INSERT FIELDS (#ORDNO #PRODNO yyyy/mm/dd USERXXXX

4 END_LOOP

************ End of RDML commands *********

Fnn=Help F6=Start F7=End **F9=Audit off** Fnn=Cancel F13=Top Fnn=Messages F21=Change

Change Text String within the Function

This operation is possible only when in change mode. When function key 21 is pressed on the Display RDML Commands screen, the Change RDML Commands screen is displayed.

DC@P309902 Change RDML Commands
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
No Act Lbl Command Parameters
******** Beginning of RDML commands ************************************
 BEGIN LOOP REQUEST FIELDS(#ORDNO #PRODNO #OUANTITY)
3 INSERT FIELDS(#ORDNO #PRODNO #QUANTITY) TO FIL
4 END LOOP ****** End of RDML commands ************************************
Fnn=Help F4=Prompt F5=W/C/P F6=Start F7=End F8=Jump F11=Tag Fnn=Cancel F13=Top Fnn=Messages F16=F/Fwd F17=F/Bkwd

If the find text strings and change text strings are entered along with the single character change flag and function key 16 is pressed, the find and change operation will be activated. The single character change flag could be either N

or A.

If N is entered for the single character change flag, it will find the next occurrence of the string starting at the position of the cursor and replace it. The scan wraps to the beginning of the function when the end of the function is reached.

If A is entered for the single character change flag, it will find all occurrences of the string from the beginning of the function to the end of the function and replace them all.

The Find/Change facility supports word wrapping around over two lines and ignores upper and lowercase differences. It will not support the use "?" as a wild card in the scan string and find/change backwards using function key 17. Also if change text string is not entered and function key 16 is pressed, it will act as a scan forward operation.

For Best Performance in the LANSA Editor

To optimise response time when using the LANSA editor to enter new function control commands, do the following:

- Turn the online syntax checker OFF.
- Turn indenting off.
- Turn color coding off.
- Turn all zooming off.
- Turn audit information off.

Input Options

The following input options apply to the panel that allows the review or change of the RDML program associated with a function:

Editor

Indicates the RDML program editor that will be used if the CHANGE function key is used. Allowable values are:

LANSA Use the LANSA editor to edit the RDML program.

SEU Use the IBM i editor SEU to edit the RDML program.

The use of these program editors is described in more detail in the following sections of this guide.

The "default" editor that appears on the screen when you first begin to edit an RDML program is determined from the LANSA system definition data area. For

more details, refer to System Definition Data Area Overview and how to change it.

Roll

Specifies how many lines are to be "rolled" when either the ROLL up or ROLL down keys are used. The roll up and roll down keys are used when reviewing or changing the RDML program to roll back- wards and forwards through the program.

The default roll value is one less then the number of lines on the display. When the roll up key is used the last line on the current display will become the first line on the next display. Likewise, when the roll down key is used the first line on the current display will become the last line on the next display.

Entering a roll value of 999 then using the roll up key is a quick way of locating the end of an RDML program.

Invoke Function Checker

Specifies whether or not the function checker should be invoked. Allowable values are:

- YES Invoke a full check of the function.
- LIST Invoke a full check of the function. An indented listing of the RDML program and associated screen layouts is to be printed. The resulting output is similar to that produced when the "produce source listings" option is used when compiling the functions in a process.
- DESIGN Invoke a partial check of the function. Only commands that affect the design of screen and/or report layouts are to be checked. This option is mainly used as a quicker means of accessing the screen and/or report design facilities.
- NO Do not invoke the function checker. Using this option will cause the function definition menu to be re-displayed.
- COMP Invoke a full check of the function. If the full check completes with no errors then a screen will be displayed allowing the user the option of submitting a compile of the function. If the function is submitted for compilation then information from the full check is passed from the RDML program editor to the compiler, reducing the time needed to compile.

This option does not produce Documentor details, because it has been designed to be quick. If Documentor details are required, use normal compile.

Note: The function checker must be run BEFORE the screen design facility can be used. The function checker is described in more detail in 4.15.10 Function Checker.

Invoke the Design Facility

Specifies whether or not the screen or report design facility should be invoked. This option is only presented after the function checker has been invoked AND no errors were found in the RDML program. Allowable values are:

SCREEN invoke the screen design facility.

REPORT invoke the report design facility.

<blank> Do not invoke either design facility, return to the function definition menu.

The screen and report design facilities are described in more detail in the following sections.

Syntax

Specifies whether or not the online syntax checker should be used. Allowable values are:

- ON Turn the syntax checker on and syntax check each function control command that is entered or changed.
- OFF Turn the syntax checker off. Do not syntax check commands that are entered or changed.

Note that the syntax checker only checks command SYNTAX. For instance if the command GOTO LABEL(L10) was entered and the syntax checker was on it would detect the unbalanced parenthesis and highlight the error.

However, if label L10 did not exist in the RDML program it would not detect this problem. This type of problem is diagnosed by the full function checker that validates syntax AND logic.

The syntax checker can be turned OFF when entering large RDML programs to improve response time.

No

Indicates the line number or sequence number associated with the RDML command line. This field is analogous to the source statement numbers used in conventional programming languages on the IBM i such as RPG or COBOL.

The field is not input capable and is automatically generated by LANSA for every line in the RDML program. Since one RDML command can continue over many input lines it can have many line or sequence numbers associated with it.

The line or sequence number is used on the RDML program listings produced by LANSA and when using the LANSA debug facility to uniquely identify one RDML command.

Note: The maximum allowed number of lines of RDML code is 4096.

Act (Action)

Allows the input of line editing or prompting actions that are used to make the input or amendment of RDML programs easier. These are described in much more detail in the following sections.

Lbl (Label)

Specifies a label that is to be associated with a command. The following points apply to command labels:

- The specification of a label is optional. Only use a label on a command when you wish to transfer control to the command via a GOTO command or a similar transfer control parameter.
- When a command spans multiple input lines specify the label on the first input line only.
- Labels must conform to IBM i conventions. The first character must be in the range A->Z and subsequent characters must be in the range A->Z or 0->9. Imbedded blanks are not allowed in a label.
- Each label used in an RDML program must be unique. No 2 commands can have the same label within one RDML program.

Command

Specifies the command that is to executed. The command entered must be one of the LANSA RDML commands or a comment which is indicated by a first character of "*" (asterisk). For more information about RDML commands and RDML programming, refer to the *Technical Reference Guide*.

Parameters

Specifies any parameter(s) that are to be associated with the command.

Command parameters may be mandatory or optional.

If a mandatory parameter is omitted from a command the error will be highlighted by the syntax checker and/or function checker.

If an optional parameter is omitted from a command the default value of the parameter is used.

Parameters may be specified in keyword format or positional format. This type of parameter formatting is identical to that used by the IBM i for CL (control language) programs.

For full details about the **RDML** commands available and their associated parameters refer to the *Technical Reference Guide*.

Where the parameters associated with a command cannot be fitted onto the same line as the command continue on the next input line.

No "+" or "-" continuation character is required when continuing RDML commands onto another line under the LANSA program editor

4.15.4 Line Edit Actions When Using the LANSA Program Editor

When editing an RDML program via the LANSA program editor various line editing actions can be used (which are compatible with those that can be used with the IBM SEU program editors).

Line editing commands are input in the column labeled "Act" (action) beside the command(s) that are to be changed, moved, edited, etc.

The editing actions available in the current version of LANSA are:

Value Entered In "Act" Field	Enter In Pairs	Editing Action That Results / Description
Ι	No	Insert 1 line after this line
In	No	Insert n lines after this line
IB	No	Insert 1 line before this line
IA	No	Insert 1 line after this line
С	No	Copy this line.
CC	Yes	Indicate start or end of a block of lines that are to be copied.
Μ	No	Move this line.
MM	Yes	Indicate start or end of a block of lines that are to be moved.
D	No	Delete this line.
DD	Yes	Indicate start or end of a block of lines that are to be deleted.
А	No	Target for copy or move is after this line.
В	No	Target for copy or move is before this line.
An	No	Target for copy or move but with multiple copies inserted after this line.
Bn	No	Target for copy or move but with multiple copies inserted before this line.

P/PP	No	Formatted prompt of the command.
U/UP	No	Unformatted prompt of the command.
F/FL	No	Allow the FIELDS parameter of this command to be built or modified by choosing fields from the data dictionary or a file definition.
Т	No	Position this command line to the top of the edit display.
ET	No	Invokes the Application Template Facility. Any value keyed into the command field is used to generically locate the required template.
		If only one is found it is directly invoked. If more than one is found, a list is displayed for selection of the one required. If none are found, a list of all templates is displayed for selection of the one required.
		RDML code generated by the template is inserted directly after the line containing the ET action. The inserted code is not checked for syntax since it should be valid code.
EF	No	Invokes the Copy Function facility. This will allow you to copy an entire function or a part of a function. RDML code copied from another function is inserted directly after the line containing the EF function.
		The syntax of the inserted code is not checked because it is assumed only valid RDML will be copied.

Examples of Line Editing - LANSA Program Editor

The following examples apply to the line editing actions that can be used when entering or amending an RDML program:

Example 1: Inserting

Starting display:

No Act Lbl Command Parameters

1 IB	L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)	
2	INSERT FIELDS(#ORDNO #PRODNO #QUANTITY)	
3 <u>I</u> _	TO_FILE(ORDERS)	
4 I4	GOTO LABEL(L10)	

Resulting display:

No Act Lbl Command Parameters

1			
2	L10 REQUE	ST FIELDS(#ORDNO #PRODNO #QUANTITY	
3	INSERT	FIELDS(#ORDNO #PRODNO #QUANTITY)_	
4		TO_FILE(ORDERS)	_
5			
6	GOTO	LABEL(L10)	
7			
8			
9			
10			

Example 2: Single Line Copy

Starting display:

No Act Lbl Command Parameters

1 C	L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)	
2_	INSERT FIELDS(#ORDNO #PRODNO #QUANTITY)	
3 A	TO_FILE(ORDERS)	
4	GOTO LABEL(L10)	

Resulting display:

No Act Lbl Command Parameters

- 1 __ L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____
- 2 _____ INSERT FIELDS(#ORDNO #PRODNO #QUANTITY)_____
- 3 _____ TO_FILE(ORDERS)______
- 4 __ L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____
- 5 _____ GOTO LABEL(L10)______

Example 3: Block Copy

Starting display:

No Act Lbl Command Parameters

1 CC L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____

2 _____ INSERT FIELDS(#ORDNO #PRODNO #QUANTITY)_____

3 _ ____ TO_FILE(ORDERS)______

4 CC L11 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____

5 A ____ GOTO LABEL(L10)_____

Resulting display:

No Act Lbl Command Parameters

- 1 ___ L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____
- 2 _____ INSERT_FIELDS(#ORDNO #PRODNO #QUANTITY) _____
- 3 _ ____ TO_FILE(ORDERS)______
- 4 __ L11 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____
- 5 _____ GOTO LABEL(L10)______
- 6 ____L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____
- 7 _____ INSERT FIELDS(#ORDNO #PRODNO #QUANTITY) _____
- 8 _ ____ TO_FILE(ORDERS)______
- 9 __ L11 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____

Example 4: Single Line Move

Starting display:

No Act Lbl Command Parameters

1 M L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____

2_	INSERT FIELDS(#ORDNO #PRODNO #QUANTITY)	
3 A	TO_FILE(ORDERS)	
4	GOTO LABEL(L10)	

Resulting display:

No Act Lbl Command_Parameters

1	INSERT FIELDS(#ORDNO #PRODNO #QUANTITY)
2_	TO_FILE(ORDERS)
3	L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)
4	GOTO LABEL(L10)

Example 5: Block Move

Starting display:

No Act Lbl Command_Parameters

L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)
INSERT FIELDS(#ORDNO #PRODNO #QUANTITY)
TO_FILE(ORDERS)
L11 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)
CHANGE #ORDNO 0
CHANGE #PRODNO *BLANKS

GOTO LABEL(L10)_____

Resulting display:

7 B

No Act Lbl Command_Parameters

1		CHANGE #ORDNO 0
2_		CHANGE #PRODNO *BLANKS
3_	L10	REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)
4		INSERT FIELDS(#ORDNO #PRODNO #QUANTITY)
5_		TO_FILE(ORDERS)
6_	L11	REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)
7_		GOTO LABEL(L10)

Example 6: Single Line Delete

Starting display:

No Act Lbl Command_Parameters

1 D L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____

2 _____ INSERT FIELDS(#ORDNO #PRODNO #QUANTITY) _____

- 3 _____ TO_FILE(ORDERS)______
- 4 D ____ GOTO LABEL(L10)_____

Resulting display:

No Act Lbl Command_Parameters

1	INSERT FIELDS(#ORDNO #PRODNO #QUANTITY)
2	TO_FILE(ORDERS)

Example 7: Block Delete

Starting display:

No Act Lbl Command Parameters

1 DD L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____

2 _____ INSERT FIELDS(#ORDNO #PRODNO #QUANTITY) _____

3 _ ____ TO_FILE(ORDERS)______

4 __ L11 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____

- 5 DD ____ CHANGE #ORDNO 0_____
- 6 ____ CHANGE #PRODNO *BLANKS_____
- 7 ____ GOTO LABEL(L10)_____

Resulting display:

No Act Lbl Command Parameters_____

1	CHANGE #PRODNO *BLANKS	

2 ____ GOTO LABEL(L10)_____

Example 8: Positioning a Line to the Top

Starting display:

No Act Lbl Command_Parameters

1 __ L10 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____

- 2 _____ INSERT FIELDS(#ORDNO #PRODNO #QUANTITY) _____
- 3 _ ____ TO_FILE(ORDERS)______
- 4 T L11 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____
- 5 ____ CHANGE #ORDNO 0_____
- 6 ____ CHANGE #PRODNO *BLANKS_____
- 7 ____ GOTO LABEL(L10)_____

Resulting display:

No Act Lbl Command_Parameters

- 4 __ L11 REQUEST FIELDS(#ORDNO #PRODNO #QUANTITY)_____
- 5 ____ CHANGE #ORDNO 0_____
- 6 ____ CHANGE #PRODNO *BLANKS______
- 7 ____ GOTO LABEL(L10)_____

4.15.5 Prompt RDML Commands

When inputting or amending an RDML program it is possible to "prompt" an RDML command (or commands) in 5 different ways.

The first is called "formatted prompting" and allows the input or amendment of the command parameters in a formatted fashion. The formatted prompting facility uses the standard command prompting features of the operating system and in operation are identical to prompting CL commands from the IBM i command entry display screen.

The second is called "unformatted prompting" and allows the input or amendment of command parameters in an unformatted fashion. The unformatted prompting facility displays the prompted command on a single screen and allows it to be modified using the insert and delete keys.

The third is called "field list prompting" and allows the FIELDS (or equivalent) parameter of some commands to be dynamically built or modified by selecting fields from the LANSA data dictionary and/or a file definition.

The fourth is called "application template prompting" and allows whole blocks of RDML commands to be automatically generated from an application template.

The fifth is called "copy function prompting" and allows another entire RDML program or a part of an RDML program to be copied.

Prompting requests are input in the column labeled "Act" (action) beside the command(s) that are to be prompted.

- For formatted prompting enter the action P or PP.
- For unformatted prompting enter the action U or PU.
- For field list prompting enter the action F or FL.
- For application template prompting enter the action ET, and optionally the generic name of an application template.
- For copy function prompting enter action EF.

Examples of Prompting

The following examples apply to using formatted and unformatted prompting when entering or amending an RDML program:

Example 1. Formatted Prompting

Starting display:

No Act Lbl Command Parameters

1 P	L10	REQUES	ST FIELDS(#ORDER)
2		INSERT	FIELDS(#ORDER #PRODNO #QUANTITY) _
3			_ TO_FILE(ORDERS
4		GOTO	L10

Resulting display 1- Formatted prompt of REQUEST command:

Formatted display reque	est (REQUEST) Prompt ++-	⊦
Enter the following:		
Fields and attributes list : F	IELDS P	
Field name :	#ORDER	
Field control attribute 1 :	"ORDER	
2:		
3:		
4:		
5:		
6 :		
7 •		
+ for more		
Dflt field design methods.	DESIGN D *IDENTIEV	
Dift field design filethod. L	DESIGN P IDENTIFY	
Dflt identification method: I	DENTIFY P *DESIGN	
Down/vertical separation:	DOWN_SEP P *DESIGN	
Across/horizontal sep: A	CROSS_SEP P *DESIGN	
Name of browse list: BI	ROWSELIST P *NONE	
EXIT key control EX	KIT_KEY P	
Enable the EXIT key:	*YES	
If EXIT key used, goto:	*EXIT	
MENU key control M	IENU_KEY P	
Enable the MENU key:	*YES	
If MENU key used, goto: *MENU

If you make these changes:

Dflt field design method: DESIGN P *IDENTIFY
Dflt identification method: IDENTIFY P *COLHDG
Down/vertical separation: DOWN_SEP P *DESIGN
Across/horizontal sep: ACROSS_SEP P 10
Name of browse list: BROWSELIST P *NONE
EXIT key control EXIT_KEY P
Enable the EXIT key: *NO
If EXIT key used, goto: *EXIT
MENU key control MENU_KEY P
Enable the MENU key: *NO
If MENU key used, goto: *MENU

Resulting display 2 - Re-display of RDML program after changes:

No Act Lbl Command_Parameters

1	L10	REQUEST FIELDS(#ORDER) IDENTIFY(*COLHDG)
2		ACROSS_SEP(10)
3		EXIT_KEY(*NO) MENU_KEY(*NO)
4		INSERT IELDS(#ORDER #PRODNO #QUANTITY)
5		TO_FILE(ORDERS)
6		GOTO L10

Example 2. Unformatted Prompting

Starting display:

No Act Lbl Command Parameters

1	L10	REQUEST	FIELDS(#ORDER)
2 U		INSERT	FIELDS(#ORDER #PRODNO #QUANTITY)
3			_ TO_FILE(ORDERS)
4		GOTO	L10

Resulting display 1 - Unformatted prompt of the INSERT command: Insert new record into file (INSERT) prompt:

FIELDS(#ORDER #PRODNO #QUANTITY) TO_FILE(ORDERS)_____

Resulting display 2 - User changes the command: Insert new record into file (INSERT) prompt FIELDS(#ORDER #PRODNO #QUANTITY #DATDUE #TOTAL #ADD1 #ADD2 #ADD3 #ADD4 #TOTLIN) TO_FILE(ORDERS)______ VAL_ERROR(L10)_____

Note that the insert and delete keys can be used on this prompt which allows the command parameters to be "pushed" off one line and onto the next line. This is very useful when a lot of changes must be made in the middle of the command.

Resulting display 3 - Re-display of RDML program after changes:

No Act Lbl Command Parameters

1	L10 REQUEST	FIELDS(#ORDER)
n	INCEDT	

2	INSERT	FIELDS(#ORDER #PRODNO #QUANTITY #DATDUE
3		#TOTAL #ADD1 #ADD2 #ADD3 #ADD4 #TOTLIN)
4		TO_FILE(ORDERS)
4		VAL_ERROR(L10)
5	GOTO	L10

Note: If command key F12 is taken from the prompt screen following any changes to the command, all changes will be lost and the command will revert to its original value.

Example 3. Field List Prompting

Starting display:

No Ac	t Lbl Comman	d Parameters
1 F	GROUP_B	Y #ORDERDATA
2	INSERT	FIELDS(#ORDERDATA) TO_FILE(ORDERS)
3	GOTO	L10

Resulting display 1 - Field list prompt of a GROUP_BY command: Group fields into a named list (GROUP_BY) prompt

1	GROUP_BY	#ORDERDATA_	

+

+

Full/partial name of fields to be chosen from data dictionary: _____ Full/partial name of files from which fields are to be chosen: ORDERS

Resulting display 2 - Selection of fields to add to GROUP_BY command: Group fields into a named list (GROUP_BY) prompt

1 GROUP_BY #ORDERDATA_____

Full/partial name of fields to be chosen from data dictionary: Full/partial name of files from which fields are to be chosen: ORDERS

Sel_FieldDescription	TypeLenDec_Selected from
X ORDER Order number	A 6 ORDERS.QGPL
X PRODNO Product number	A 7 ORDERS.QGPL
X QUANTITY Quantity ordered	P 92 ORDERS.QGPL +

Resulting display 3: Field list prompt of a GROUP_BY command Group fields into a named list (GROUP_BY) prompt

1 GROUP_BY NAME(#ORDERDATA) FIELDS((#ORDER)(#PRODNO) (#QUANTITY))_______+ Full/partial name of fields to be chosen from data dictionary: Full/partial name of files from which fields are to be chosen: ORDERS

Sel Field Description	TypeLenDec Selected from
ORDER Order number	A 6 ORDERS.QGPL
PRODNO Product number	A 7 ORDERS.QGPL
QUANTITY Quantity order	red P 9 2 ORDERS.QGPL +

Resulting display 4 - Re-display of RDML program after changes:

No	Act Lb	l Comma	nd Parameters	
1_		GROUP_H	BY NAME(#ORDERDATA) FIELDS((#ORDER)	
3_			(#PRODNO) (#QUANTITY))	_
2		INSERT	FIELDS(#ORDERDATA) TO_FILE(ORDERS)	
3_		GOTO	L10	

Note - If command key F12 is taken from the prompt screen following any changes to the command, all changes will be lost and the command will revert back to its original value.

Example 4. Application Template Prompting

Starting display:

No Act Lbl Command Parameters

1	********* Display list of products (page_at
2	********* a time) and_allow the user to
2	********* select the one required
3 ET	BBPAT

Note the use of the name BBPAT in the command column. The programmer is indicating that an Application Template called BBPAT is to be invoked by this request. The name can be omitted to cause a list of all templates to be displayed, or specified generically to cause a partial list of templates to be displayed. Resulting display 1: Code generated by the template is inserted Note that any "question & answer" interaction that the Application Template used is omitted from this example. Any such interaction would take place in a pop-up window overlaying the lower half of the RDML edit screen, just as it does when an application template is invoked when creating a new function.

1	 ******	Display list of products (page at
2	 *****	at a time) andallow the user to
4	 *****	select the one required
5	 	
6	 CHANGE	FIELD(#IO\$KEY) TO(UP)
7	 SELECT	FIELDS(#PRODLIST) FROM_FILE(PRODMST)_
8	 EXECUTE	SUBROUTINE(PROD\$DISP)
9	 ADD_ENTR	Y TO_LIST(#PRODLIST) WITH_MODE(*DISPLAY)

etc, etc...

Example 5. Copy Function Prompting

Starting display:

No Act Lbl Command Parameters

1	DEFINE	FIELD(#ORDNO) TYPE(*CHAR) LENGTH(6)_
2	DEFINE	FIELD(#PRODNO) TYPE(*CHAR) LENGTH(7)
3 EF _	DEFINE	FIELD(#QTY) TYPE(*DEC) LENGTH(9)

Resulting display 1: Copy selection screen:

DC@P309	905	Copy I	RDML source	by LAN	ISA Editor
Process Function	: XXXX : XXXX	XXXXX XXX	XX XXXXXX XXXXXXXX	XXXXXX XXXXXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Specify details of the function that you may wish to copy RDML code from.					
Copy all RDML code NO YES, NO Process name XXXXXXXXX Function name XXXXXXX					

The Copy RDML source by LANSA Editor screen is requesting that you indicate whether to copy an entire or a part of a function.

If you specify YES for the *Copy all RDML code* input field and valid process and function names, it will copy the entire RDML code from the specified function. If you wish to copy a part of another function or to display another function specify NO for the 'Copy all RDML code' input field and valid process and function names.

The process and function names nominated for copy or display are checked for existence and access rights. If any error occurs the screen will be re-presented with an error message.

IF 'YES' is entered for the 'Copy all RDML code' input field the following screen will appear after copying the entire RDML code.

Resulting display 2 - Entire code copied from another function is inserted:

No	Act Ll	ol Comman	d Parameters
1_		DEFINE	FIELD(#ORDNO) TYPE(*CHAR) LENGTH(6)_
2_		DEFINE	FIELD(#PRODNO) TYPE(*CHAR) LENGTH(7)
3_		DEFINE	FIELD(#QTY) TYPE(*DEC) LENGTH(9)
4_		BEGINLO	OP
5_		REQUEST	FIELDS(#ORDNO)
6_		INSERT	FIELDS(#ORDNO #PRODNO #QTY)
7			TO FILE(ORDERS)

etc, etc...

If 'NO' is entered for the 'Copy all RDML code' input field, the indicated copy range will have to be entered manually by the user.

Resulting display 3 - Display/Copy RDML commands:

DC@P309906 Display/Copy RDML Commands

No Act Lbl Command Parameters			
********* Beginning of RDML commands *********			
1 BEGINLOOP			
2 CC REQUEST FIELDS(#ORDNO #PRODNO #QTY)			
3 INSERT FIELDS(#ORDNO #PRODNO #QTY) TO_FILE(ORD			
4 CHANGE #ORDNO *BLANKS			
5 CC CHANGE #PRODNO *BLANKS			
6 ENDLOOP			
********* End of RDML commands *********			
F1=Help F3=Exit F12=Cancel F14=Messages			

This *Display/Copy RDML Commands* screen will show the RDML commands of the function you requested to copy or display. If you need to copy one single line, enter action code **C** or to copy a range indicate **CC** as a pair. Only C or CC are allowed for the action code.

F6 and F7 can be used to position the display at the start or end of the RDML function.

A Find string with F16 can be used to scan and position the display.

Resulting display 4 - Part of the RDML code is copied and inserted:

No Act Lbl Command Parameters

1	 DEFINE	FIELD(#ORDNO) TYPE(*CHAR) LENGTH(6)
2	 DEFINE	FIELD(#PRODNO) TYPE(*CHAR) LENGTH(7)
3	 DEFINE	FIELD(#QTY) TYPE(*DEC) LENGTH(9)
4	 REQUEST	FIELDS(#ORDNO #PRODNO #QTY)
5	 INSERT	FIELDS(#ORDNO #PRODNO #QTY) TO_FILE(ORDE
6	 CHANGE	#ORDNO *BLANKS
7	 CHANGE	#PRODNO *BLANKS

4.15.6 Function Keys Available When Using Formatted Prompt

As already mentioned the formatted command prompting facility of LANSA uses a standard feature of the IBM i operating system.

When a command is being displayed in a formatted prompt it is under the control of a program provided by IBM and thus has different function key assignments. These are:

Key Description / Use

- 3 End/cancel command prompt. If used command prompting is canceled and the command remains unchanged by the prompt. A message is issued to indicate the command was canceled.
- 12 Display previous prompt screen. Where a command prompt uses multiple screens use this function key to re-display the previous screen.
- 16 Prompting is complete. Use this function key when the desired changes have been made and there is not need to display any following prompt screen.
- 5 Revert to default values/Refresh Use this function key to cause all parameters specified for a command to revert to their default values.

4.15.7 About the IBM Program Editor SEU

It is possible to edit an RDML program using the IBM supplied source editors called SEU. If the "editor" nominated at the top right of the RDML program display screen is SEU when the CHANGE function key is used, then the respective IBM utility will be used to edit the RDML program, rather than the LANSA program editor.

SEU is universally used by programmers and very widely used by other IBM i users. It is beyond the scope of this guide to describe SEU in any detail. If you are unfamiliar with the features and functions of SEU refer to the appropriate IBM manual.

In addition, SEU have online HELP facilities of their own. Once you have started to use SEU press the HELP key and review the extensive HELP text available.

By using SEU, an RDML program can be edited just like any other RPG or CL program. All the features of SEU are available and can be used.

Some points about the way SEU is used by LANSA are:

- LANSA does not store RDML programs in a source file. Thus LANSA creates a "work" file called FUNCWORK in library QTEMP and re-formats the RDML program into it to allow it to be edited as source. When the edit session is completed LANSA then re-formats the source RDML program back into the internal LANSA format.
- The re-formatting procedure may change the program layout. Since LANSA must re-format the RDML program to convert it to source and then to re-format it back into the LANSA internal format, the layout of the actual source lines may vary slightly. This change will never affect the commands used or the program logic in any way merely the way the commands are formatted into lines.
- The RDML Audit information associated with each RDML command is not preserved when using SEU. When the edit session is successfully completed, all RDML commands for the function will be updated to have the same RDML Audit information.: date, user and, if in use, task identifier.

4.15.8 IBM Program Editor SEU

If you have elected to use the IBM source editor SEU and used the CHANGE function key a screen similar to the following example will be presented initially:

Specify details of any **other** functions that you may wish to **browse** or **copy** from while using SEU on this function

Process name Function name

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen is requesting that you nominate any other functions that you may wish to browse or copy from while using the SEU utilities.

The reason that they must be nominated before beginning to use SEU is that LANSA must also re-format them into source format and copy them to the FUNCWORK work file, described in 4.15.7 About the IBM Program Editor SEU.

If you do not wish to browse or copy any other functions press enter without nominating any process or function names. The SEU utility will be then invoked.

If you wish to browse or copy other functions then specify their names. If you do not nominate the process to which the function belongs, the current process

(shown at the top of the screen) is assumed.

All functions nominated for browse or copy are then checked for existence and access rights. If any error occurs the screen will be re-presented with an error message.

When functions are placed into the FUNCWORK work file for browse or copy they are added as members that have a name like:

BRWfffffff

where "fffffff" is the function name and the "BRW" indicates they have been added for browse/copy purposes.

The function that is actually being edited in the FUNCWORK file (shown at the top of the screen) always has a name like:

EDTfffffff

where "ffffffff" is the function name and the "EDT" indicates it has been added for edit purposes.

Note: Browse/copy members are not removed from the FUNCWORK work file when the edit session ends. Thus they are available at all times until you sign off. As such it is not necessary to keep specifying the same browse/copy functions each and every time the SEU utility is invoked.

4.15.9 Exit from the IBM Editor SEU

When you have finished editing the RDML program, use function key 3 to exit from the SEU utility. A screen similar to this will be presented:

Exit Type choices, press Enter.

Change/create member <u>N</u>	Y=Yes, N=No
Member <u>EDTffff</u>	<u>fff</u> Name
File FUNCWO	RK Name
Library QTEMP	Name
Text	
Resequence member $\dots $	Y=Yes, N=No
Start <u>0001.00</u>	0000.01 - 9999.99
Increment <u>01.00</u>	00.01 - 99.99
Print member <u>N</u>	Y=Yes, N=No
Return to editing $\dots \dots \dots$	Y=Yes, N=No
-	
Go to member list <u>N</u>	Y=Yes, N=No
F3=Exit F5=Refresh F12=Can	ncel

The SEU exit option will be pre-filled to N (change/create member) if you did not change the RDML program in any way. Otherwise it will be pre-filled to Y (change/create member).

Normally you would only ever use Y or N in the "change/create member" option to exit from SEU.

A Y in the "Return to editing" prompt may be used to return to SEU and continue editing. Other options can be used if desired, but you may find that any changes made are lost and the RDML program remains unchanged within LANSA.

Note that the file and library names are always FUNCWORK and QTEMP respectively. This applies no matter what function is being edited. This is a special temporary source file that LANSA has created in library QTEMP. These values can be changed if required, but again you may find that any changes made are lost and the RDML program remains unchanged within LANSA.

4.15.10 Function Checker

After you have edited an RDML program (using either the LANSA or the IBM SEU editor) you may choose to use the online function checker.

You have the option to use the function checker whenever a message like this appears at the top of the current screen:

Invoke the function checker_____ YES,LIST,DESIGN,NO

The function checker performs the following tasks:

- It checks the syntax of the RDML commands.
- It checks the validity of the RDML commands.
- It creates default screen/report designs and extracts information that is required by the screen/report design facilities.

In addition, the function checker can produce a report that details all the screen designs being used by the function and fully indented listing (with imbedded error messages) of the RDML program.

To invoke the function checker enter YES, LIST or DESIGN in response to the prompt that appears at the top of the screen. To avoid using the function checker enter NO.

When the function checker has completed execution the RDML program will be re-displayed. Some points to note about this are:

- If the screen is "input capable" then the function checker has found some fatal error(s) in the RDML programs. Each command that is in error will be displayed in reverse image and the associated error message(s) will be displayed at the bottom of the screen. Correct the errors and run the function checker again.
- If the screen is not "input capable" then no fatal errors were found by the function checker. However, some warning messages may have been issued. Review any warning messages issued and take any appropriate action. In addition there will be a prompt at the top of the screen that allows the screen design facility to be used. This is described in detail in the following section.
- If the LIST option was used then 2 separate reports will have been produced. The first details every screen format that is designed as a result of REQUEST or DISPLAY commands (or as far as possible considering the errors in the RDML program). The second lists the RDML program in full with all associated error messages. The RDML program listing is very useful because the error messages appear directly after the command which caused

them and include the second level message text. The same messages will be displayed at the workstation, but they appear in a group at the bottom of the screen and if there are a lot of messages it is sometimes difficult to determine which command(s) they relate to.

• If the DESIGN option was used, only commands that affect the design of screen and/or report layouts will have been checked. This means that some commands that contain errors may not be shown as containing errors. This is not a problem however, because when the job to compile this function is eventually submitted, a full check of the function will be performed. This will pick out the errors and print details of them in the resulting function listing.

It is easy to code an RDML program that will not "compile". Such errors usually result from spelling or syntax problems. The function checker will locate any error that will prevent an RDML program from "compiling" and highlight it with the appropriate error message(s). In such situations you should correct the error and run the function checker again.

Note the use of the word "compile" in this context. If the RDML program contains errors that prevent it from "compiling" then it must be corrected by you. This is because the error exists at the RDML level.

However, if no errors are found in the RDML program by LANSA, but the resultant RPG program generated by LANSA will not compile, it indicates a problem in LANSA itself. In such situations report the problem to your LANSA product vendor.

The function checker will never operate quickly. A typical RDML program takes 10 - 30 seconds to check. Many factors influence the speed of the function checker, however the most significant is the number of DISPLAY and REQUEST commands in the RDML program. Since each DISPLAY or REQUEST command used causes the function checker to validate and design a screen format it is easy to see how they slow it down.

4.15.11 Screen Design Facility

Once the function checker has been run and no fatal errors were found in the RDML program it is possible to use the screen design facility. The function checker is described in detail in preceding sections of this chapter.

It is easy to tell when the screen design facility can be used because a message like this will appear at the top of the screen:

Invoke the design facility for _____ SCREEN, REPORT

If this message does not appear at the top of the screen it will be for one of the following reasons:

- You are currently inputting or changing the RDML program. In this case complete the required input, run the function checker and the message should appear.
- The function checker has just been run but has found fatal error errors in the RDML program. In such cases correct the error(s), run the function checker again and the message should appear.

To invoke the screen design facility enter SCREEN in response to the message previously described and press enter.

Sel	Seq	Lbl Command Parameters and Screen Format Details
	7	REQUEST FIELDS((#ORDNO) (#PRDNO) (#QTY)
		Format name : R@F00001 Type : Single display
	15	DISPLAY FIELDS((#MAINHEAD))
		Format name : R@F00002 Type : Single display
		Format name : R@F00003 Type : Single display
	22	DISPLAY FIELDS((#ORDERHEAD)) BROWSELIST (#ORDER)
		Format name : R@F00004 Type : Browse hdr/list

This screen is a list of all the DISPLAY and REQUEST commands that were found in the RDML program and information about the screen format(s) that have been designed for them. In this example, it can be seen that:

- RDML program statement number 7 is a REQUEST command and one record format named R@F00001 has been designed to accommodate it. The record format is a single record display (i.e. it has no browse component).
- RDML program statement number 15 is a DISPLAY command and two record formats named R@F00002 and R@F00003 have been designed to accommodate it. Both are single record displays (i.e. they have no browse component). The fact that 2 record formats have been designed indicates that not all the fields in group #MAINHEAD would fit on one screen. However, at the RDML program level the 2 screens are treated like one "long" screen.
- RDML program statement number 22 is a DISPLAY command and one record format named R@F00004 has been designed to accommodate it. The record format contains a header portion and a browse portion.

The record format names are only provided as a "link" between the screen design facility and the screen design listing produced by the function checker. When examining the screen design listing produced by the function checker it is possibly to directly relate a screen design to information on this display by the screen format name.

After considering the information on the display you can do one of two things:

- Press the enter key without selecting any of the screen formats. In this case the RDML program will be re-displayed for amendment.
- Select one of the screen formats to be (re)designed by entering any nonblank character beside it in the column headed "Sel".

4.15.12 Design a Screen Format

Once a screen format has been selected for design purposes an "image" of what the screen format currently looks like is presented.

The content and format of the "image" will of course vary widely according to the fields that are used on the format, so for purposes of explanation a specific example is used.

The example screen is the result of a DISPLAY command that was coded into the RDML program as:

DISPLAY FIELDS(#CUSNUM #CUSNAM #ADDRL1 #ADDRL2 #ADDRL

The "image" of the screen that results from the DISPLAY command might look like this when presented by the screen design facility:

Note the following points about this example:

- Lines 1 -> 23 of the "image" are just that. They are an "image" of what the screen format will look like when it is actually being used in the function.
- On line 24 is the only input capable field on the entire screen. It is called the "action" field and is used to "re-design" or "paint" the screen "image" that is displayed on lines 1 to 23.

Using the "action" field is very simple. You enter one of the "action codes" below to indicate what you wish to do and then follow the instructions that are displayed on line 24.

When you are comfortable with using "action" codes you can then begin to work more quickly using the associated function key to reduce the number of actions that you must take to make the painter modify the screen layout.

The list of valid "action" codes and function keys for screen design is as follows:

Action Code	Function Key	Action That This Initiates
<blank></blank>	F5	Cancel any pending actions or errors.
EX	F3 / F12	Exit from the screen design facility.
GR		Place a grid over or remove a grid from the image.
Μ	F16	Move field(s) or text string(s).
TI	F6	Insert a text string.
TD	F22	Delete a text string.
TC	F21	Change a text string.
FI	F6	Insert a field.
FD	F22	Delete a field.
ID	F17	Change the method used to identify a field.
Pxxx	N/A	Change shape/size/location of Pop-Up Window. Refer to following sections for details.

4.15.13 Report Design Facility

Once the function checker has been run and no fatal errors were found in the RDML program it is possible to use the report design facility. The function checker is described in detail in preceding sections of this chapter.

It is easy to tell when the report design facility can be used because a message like this will appear at the top of the screen:

Invoke the design facility for _____ (SCREEN, REPORT)

If this message does not appear at the top of the screen it will be for one of the following reasons:

- You are currently inputting or changing the RDML program. In this case complete the required input, run the function checker and the message should appear.
- The function checker has just been run but has found fatal errors in the RDML program. In such cases correct the errors, run the function checker again and the message should appear.

To invoke the report design facility enter REPORT in response to the message previously described and press enter.

The resulting display will look something like this:

Opt_Rpt_Seq_Command___Parameter(s)_____Cycles: 1 1 1 DEF_HEAD NAME(#HEADING) FIELDS((#ORDNUM) (#DIVNUM (#CUSNUM) (#CUSNAM)) X 1 2 DEE LINE NAME(#DETAIL) FIELDS((#DDODNO)(#DDODES)

X 1 2 DEF_LINE NAME(#DETAIL) FIELDS((#PRODNO)(#PRODES) (#QANTY) (#VALUE))

This screen displays all the DEF_HEAD, DEF_FOOT, DEF_LINE and DEF_BREAK commands that were found in the RDML program. In following sections these commands are referred to as the DEF_XXXXX report commands.

In this example you can see:

- RDML program statement number 1 is a DEF_HEAD (define report heading) command and thus is eligible for "painting" via the report design facility.
- RDML program statement number 2 is a DEF_LINE (define report line)

command and thus is eligible for "painting" by the report design facility. After considering the information on the display you can do one of two things:

- Press the enter key without selecting any of the DEF_XXXXX report commands. In this case the RDML program will be re-displayed for amendment.
- Select one of the DEF_XXXXXX commands as the command to be "edited" or "painted" by entering a "1" beside it in the column headed "Opt". Optionally, additional DEF_XXXXX commands can be chosen for display by entering an "X" beside them. Additional commands chosen in this way cannot be edited by the report design facility (only one DEF_XXXXX command can be edited at a time), but they will be displayed. This is useful when checking and comparing the layouts of different components of the report (i.e. heading section, details section, etc).

In addition, a "Cycle" (report cycles to be emulated) value may be specified at the top of the screen and a "Repeat" (number of times to repeat the image) value may be specified for each individual DEF_LINE command beside the command in the column headed "Rpt". These values are used to enhance the "image" of the report and are described in more detail in 4.15.15 Use Cycle and Repeat Values to Enhance Report Image.

4.15.14 Design a Report

Once one or more DEF_XXXXX report commands have been chosen for design/display purposes an "image" of what the report line(s) will look like when they are printed is presented.

The content and format of the "image" will of course vary widely according to the commands selected and the fields used within the commands, so for purposes of explanation a specific example is used.

The example screen is the result of selecting a DEF_HEAD command (by entering option 1) and a DEF_LINE command (by entering option X) like this:

Op	t_Rpt_Seq_Command_	_Parameter(s)	Cycles:1
1	1 DEF_HEAD NAM	E(#HEADING) FIELDS	S((#ORDNUM) (#DIVNUM)
	(#CUSNUM)(‡	#CUSNAM))	
Х	1 2 DEF_LINE NAM	E(#DETAIL) FIELDS((#PRODNO) (#PRODES)
	(#QUANTY)(#	#VALUE))	

The "image" of the report resulting from this selection might look like this when presented by the report design facility:

Order no DDDD Division no DD Customer no DDDD Name D

Act :

This example is for an 80 column workstation. Note how field #CUSNAM does not entirely fit onto the display. To view field #CUSNAM you would have to use the commands available to move screen "window" from left to right. These are described in detail in the following sections.

If you are using a 132 column workstation (i.e. a 3180) then 132 columns of the report will be displayed. Of course, if you are designing a report wider than 132 columns it will still be necessary to window the display from left to right.

Note the following points about this example:

- Lines 1 -> 23/26 of the "image" are just that. They are an "image" of what the lines will look like when actually printed. They are not input capable and cannot be directly changed in any way.
- On line 24 is the only input capable field on the entire screen. It is called the "action" field and is used to "design" or "paint" the report "image" that is displayed on lines 1-23.

Using the "action" field is very simple. You enter one of the "action codes" below to indicate what you wish to do and then follow the instructions that are displayed on line 24.

When you are comfortable with using "action" codes you can then begin to work more quickly using the associated function key to reduce the number of actions that you must take to make the painter modify the report layout.

The list of valid "action" codes and function keys for report design is as follows:

Action Code	Function Key	Action That This Initiates
<black></black>	F5	Cancel any pending actions or errors.
EX	F3 / F12	Exit from the report design facility.
GR		Place a grid over or remove a grid from the image.
Μ	F16	Move field(s) or text string(s).
TI	F6	Insert a text string.
TC	F21	Change a text string.
TD	F22	Delete a text string.
FI	F6	Insert a new field.
FD	F22	Delete a field from the image.
ID	F17	Change the method used to identify a field.
WLn		Window left "n" characters
WRn		Window right "n" characters
Wn		Window to column "n"

W?	Display current window details
Run	Roll image up "n" lines
RDn	Roll image down "n" lines
SP	Change line space before/after values

4.15.15 Use Cycle and Repeat Values to Enhance Report Image

When using the report design facility 2 special values appear on the command selection screen. These are called the "cycle" value and the "repeat" value.

The "cycle" value appears once in the upper right of the screen and is usually pre-filled to 1. The value specified here indicates the total number of report "cycles" that are to be emulated by the screen design facility when building an image of the report commands.

The "repeat" value appears in the column headed "Rpt" in the lower half of the screen (for DEF_LINE commands only) and is usually pre-filled to 1. The value specified here indicates the number of times the image should be repeated by the screen design facility when building an image of the DEF_LINE command.

Both values are used to enhance the report image that is built by the screen design facility. The way these values are used is best demonstrated by example.

Consider a report that is defined by the following 3 commands:

DEF_HEAD NAME(#HEADING) FIELDS(#COMPANY)

DEF_LINE NAME(#DIVSALES) FIELDS(#DIVNAM #PRODNO #PRODES #QUANTITY)

DEF_BREAK NAME(#DIVTOTAL) FIELDS(#DIVQTY) TRIGGER_BY(#E

If all lines were selected for display by the report design facility and the default of 1 used for the "cycle" and "repeat" values the image might look something like this:



If the "repeat" value associated with the DEF_LINE command was set to 3. The report image would be enhanced to look like this, which is closer to what the report will look like when it is actually printed:

Company XXXXXXXXXX	
Divn	
Name Product Description Quantity	
XXXXX XXXXXXXX XXXXXXXXXXXXXXXXXXXXX	K 99999.9
XXXXX XXXXXXXX XXXXXXXXXXXXXXXXXXXXX	K 99999.9
XXXXX XXXXXXXX XXXXXXXXXXXXXXXXXXXXX	K 99999.9
Division total 99999.99	
	/

Note how the DEF_LINE command image is repeated 3 times.

If the "cycle" value was set to 2 and the "repeat" value associated with the DEF_LINE command was set to 3. The report image would be enhanced to look like this, which is even closer to what the report will look like when it is actually printed:

Company XXXXXXXXXX Divn Name Product Description Quantity 99999.9 99999.9 Division total 99999.99

/

Note how the DEF_LINE and DEF_BREAK command are subjected to 2 emulated report cycles, and the DEF_LINE command is repeated 3 times every time it occurs.

4.15.16 A Note for Right-to-Left Language Users

The following sections detail the "actions" that are used to control the LANSA screen and report design facilities.

The actions that move and insert fields on screen panels or report layouts often rely on the location of the screen cursor for final positioning.

Where the following sections say that you should use the screen cursor to specify the LEFT or UPPER LEFT location of a field you should ignore the word LEFT and substitute the word **RIGHT** instead.

4.15.17 Action EX, F3 or F12 (Exit)

This action is used in both the screen and report design facilities.

When an action code of EX (or function key F3 or F12) is used the following events occur:

• A pop-up appears in the centre of the current screen panel requesting that the type of exit required be selected:

Exit from Painter :
Use cursor to make a selection :
Exit and update design :
Exit and do NOT update design :
Ignore Exit, Continue designing :
Restart from initial design :
Fnn=Help Fnn=Messages Fnn=Cancel :

Move the screen cursor onto the same line as the option that you wish to use and then press the enter key.

• If you choose to actually exit from the painter, then the screen or report command selection prompt will be re-displayed.

From either of these prompts another screen format or report command can be selected for modification.

The RDML screen commands DISPLAY and REQUEST and the DEF_XXXXX report definition commands allow the nomination of specific row and column numbers to position fields. Once the screen or report design facility has been used the associated DISPLAY, REQUEST or DEF_XXXXXX commands will be automatically re-written to include the desired row and column numbers.

Subsequently the specific row and column numbers can be amended using the screen or report design facilities again or by direct amendment of the DISPLAY, REQUEST or DEF_XXXXXX commands in the RDML program. Refer to the *Technical Reference Guide* for more details of the DISPLAY, REQUEST and DEF_XXXXX report commands.

Remember that the RDML program is completely self contained. All program logic and screen/report designs are stored in the RDML program only.

4.15.18 Action GR (Add/Remove Grid)

Action GR is used in both the screen and report design facilities

Action code GR is used to generate or remove a "grid" from the screen or report image. The grid consists of 2 axes. The horizontal axis indicates the column positions across the screen/report and the vertical axis indicates the line numbers down the screen/report. When action code GR is used the following events occur:

- 1. If there is already a grid on the screen/report image it is removed.
- 2. If there is no grid currently on the screen/report image a grid is generated. The horizontal and vertical axes are placed on the screen/report image so that they "cross" at the position the screen cursor was located when the request was made.

4.15.19 Action M or F16 (Move)

This action is used in both the screen and report design facilities.

When action code M is used the following occurs:

- 1. The user is asked to select the field or text that is to be moved by placing the screen cursor anywhere on the field and pressing the enter key.
- 2. The user is then requested to position the screen cursor at the location the field or text is to be moved to and press the enter key.
- 3. If the field or text can be fitted at the required location it will be moved. If not, an error will be displayed and step 2 can be repeated.

To use function key F16 complete one of the following:

- 1. Position the screen cursor anywhere on the field or text that is to be moved and press function key F16.
- 2. Position the screen cursor at the location the field or text is to be moved to and press the enter key.
- 3. Move the screen cursor to another field in the same screen portion (header or browselist) or the same report line and use function key F16 again. This will select another field to be moved. By repeatedly using F16 you may choose up to 50 fields to be moved. When all of the required fields have been selected they form a logical "box" on the screen. By moving the cursor to where you want the upper left (or upper right for RLTB users) of the "box" to be moved to, and pressing enter you can cause all selected fields within the "box" to be moved in one go.

This type of operation is called a "multiple move" because it can move multiple fields in one operation. It is not strictly a "block" move because fields within the boundaries of the logical "box" that have not been selected are not moved.

Some important points about moving fields or text are as follows:

To Select the Field to Be Moved

To select the field to be moved, place the screen cursor anywhere on the first occurrence of the field or its associated label, description or column headings and press the enter key or F16.

When a Field is Being Moved

When a field is being moved the field and any associated identification text is

moved. For instance, if the field is identified by column headings like this:

Company Order Number 999999999

or identified by a label such as:

Order number : 999999999

or identified by a description such as:

Order number : 999999999

then a "block" that contains the field and its associated identification text is what will be moved. The "blocks" associated with the 3 examples above can be visualized as:



When a field has been selected for a move the associated "block" that contains the field will be displayed in reverse image. Thus, it is very easy to see exactly what is to be moved and how it will fit into the desired new location.

Indicating Position Where to Move a Field

When indicating the new position to which a field should be moved you should carefully note the following:

• When there is only one occurrence of the field possible on the screen panel or report page (which means the field is in the main or header area of a screen panel or in the header or footer area of a report) the "target" position that you indicate with the screen cursor is for the upper left corner of the "block" containing the field and its identification details (right-to-left users should indicate the "mirrored" upper right corner position):



• When multiple occurrences of the field are possible on the screen panel or report page (which means the field is in a browse list on a screen panel or in a detail or break line on a report) the "target" position that you indicate with the screen cursor is interpreted as a left hand side position as in the following examples (right-to-left users should indicate the "mirrored" right hand side positions):

| Company |



Additionally, you should only attempt to move and target the first occurrence of a field. Additional occurrences of a field in a browse list or detail/break line will be automatically moved to the same relative location as the first occurrence.

Performing a Multiple Move

When performing a multiple move all selected fields must come from the same screen panel portion (header or browselist) or the same report line. Attempts to select fields in another portion do not cause an error as they are simply ignored.

Using a Multiple Move

When a multiple move is used all selected fields/text form a logical "box". To move the "box" position the cursor at the upper left (or right for RLTB users) position and press enter. All selected fields within the logical box will then be moved (providing that they can be).

For example:

UL

```
---->Customer Number : XXXXXXXXX
```

--

By positioning the cursor on all 3 of these fields (or their labels) and pressing F16 each time you can create a logical "box" that can be visualized as having its

upper left corner on the "C" in the string "Customer Number". By moving the cursor to where you want the upper left corner of the "box" to be moved to, and pressing enter, you can move the entire "box".

Note that if all fields and text were selected in this example, then the upper left corner of the logical "box" is now two lines above the "C" in "Customer Number":

UL

----> CUSTOMER DETAILS AS AT 1/09/91

--

Note also that not all fields inside the logical "box" need to be selected and therefore moved. This facility can be very useful in situations like this:

UL

>BASIC DETAILS	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
ADDRESS DETAILS	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
CURRENT SHIPMENT DETAILS<	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	

By using F16 to select "BASIC DETAILS", "ADDRESS DETAILS" and "CURRENT SHIPMENT DETAILS" you can form a logical "box". The upper left corner is on the "B" in "BASIC DETAILS".

Moving the cursor so that it is over the "S" in "BASIC DETAILS" and pressing enter would cause the following change:

Note that even though the logical "box" contained parts of the "XXXXXXX" values, they were not moved. That is because they were not selected via F16 to be within the logical "box".
4.15.20 Actions TI or F6 and TC or F21 (Text Insert/Change)

Actions TI/TC are used in both the screen and report design facilities.

When action TI is used the following steps occur:

• A field appears at the bottom of the screen that allows up to 80 characters (screen design facility) or 120 characters (report design facility) of text to be specified.

Additionally (in the screen design facility only) an area to specify CUA emphasis, attributes and colors also appears.

- Once the desired text (and optional attributes) have been specified, and the enter key pressed, the user is asked to nominate, by positioning the screen cursor, the position on the screen at which the text string is to be inserted.
- If the text will fit at the position nominated it is inserted into the screen/report image.

To use function key F6 do the following:

- 1. Use function key F6 at any time.
- 2. In the resulting action box choose the option to insert text by positioning the screen cursor and pressing enter.
- 3. Specify and position the new text as previously described.

When action TC is used the following steps occur:

- The user is requested to position the screen cursor on the text to be changed and press enter.
- The selected text then appears at the bottom of the screen in exactly the same format that it had when originally input.
- The user can then edit the text and its attributes (screen painter only) and press enter.
- The user is asked to indicate with the screen cursor where the text string is to be (re)inserted back into the screen or report layout and press the enter key.
- If the text will fit back at the position nominated it is (re)inserted into the screen/report image.

To use function key F21 do the following:

- 1. Position the cursor on the text to be changed and press F21.
- 2. Edit and (re)position the text as previously described.

Some important points to note about inserting text are:

- During the first text operation a warning about unconstrained text usage will appear. Press enter to proceed after this warning has been read.
- Up to 50 text strings can be inserted into one screen image or report line.
- The maximum length for a text string on a screen panel is 80 characters. For a report line it is 120 characters.
- Do not position text strings that "wrap around" onto following lines.
- RLTB (right to left users) are strongly advised against inserting text strings longer than 80 characters into reports. Failure to observe this warning may complicate the subsequent movement of the text strings around the report.
- In the screen design facility, options appear to allow attributes of the text string to be specified, these include:
 - CUA emphasis
 - Attributes (underline, reverse video, etc)
 - Colors

The following should be noted about these options:

- The use of "CUA Normal" or "CUA Emphasized" is best, because the final attributes assigned to the text string are derived from the site standards for these types of text strings. This avoids a "rainbow" system and produces consistent results.
- If either CUA option is chosen all other options are reset to N (not required).
- Color options are only used when on a color device. This applies to both the screen design facility and the resulting end user screen panel.
- IBM i restrictions and complications regarding the use of colors and attributes in combination exist. Refer to the IBM Manuals for more details. If these complications are not understood, use the CUA options only.
- Where a DISPLAY or REQUEST command results in multiple screen formats, text strings can only be inserted onto the first screen format in the set. In such situations modify the RDML program to use multiple DISPLAY or REQUEST commands (each with only 1 screen panel) instead.
- Text strings cannot be inserted into the "browse" portion of a screen image in the current version. To achieve this result use a field in the browselist that contains the required text string value.

4.15.21 Action TD or F22 (Text Delete)

Action TD is used in both the screen and report design facilities When action TD is used the following steps occur:

- 1. The user is requested to position the cursor anywhere on the text string that is to be deleted.
- 2. When the enter key is pressed the text string selected is erased from the screen/report image.

3.To use function key F22, position the cursor on the text to be deleted and press F22.

4.15.22 Action FI or F6 (Field Insert)

Action FI is used in both the screen and report design facilities When action FI is used the following steps occur:

- The user is requested to name the field that is to be inserted into the report image and the identification method (C,L,D or N) to be used when inserting the field.
- When a screen panel is being designed, and it has a browselist area, an action box will appear asking the user whether the field is going to be inserted into the header/flat panel area, or, into the browse list area.
- The user is then requested to position the screen cursor so as to specify the position the field is to be inserted at.
- If the field (using the nominated identification method) will fit into the report image, at position nominated, it is inserted, else an error message is displayed and step 2 can be repeated.

To use function key F6 do the following:

- 1. Use function key F6 at any time.
- 2. In the resulting action box choose the option to insert a field by positioning the screen cursor and pressing enter.
- 3. Specify and position the field as previously described.

Some important points about inserting fields are:

- When asked for the name of the field to be inserted you may use F4=Prompt to choose the field from Data Dictionary or from any field defined within the current function.
- The specification of the identification method for fields being positioned into the browse lists area is unnecessary. The identification method is automatically derived from the fields location in the browse list.
- When attempting to insert a field into a screen panel (with any identification method) you may cause this error message:

"Field description too wide. Insert via editor and redesign screen"

This error may be annoying at first. However, it will help if you understand the reasons for this error, and the implications of what would happen were it not issued: When fields on a screen panel are identified by their descriptions the longest description of any field is used to "align" all the other fields. All shorter descriptions are padded out with aligned and double spaced "leader dots" as per the CUA guidelines.

If you were permitted to insert a field with a still longer description, then all the alignment and leader dot padding becomes incorrect and needs to be recalculated. If this recalculation was performed then many fields would have to be (re)positioned.

The complexity of this operation requires that you do the insert operation, via the RDML editor, directly into the associated command, then use the full function checker's design capability to (re)design the screen panel.

4.15.23 Action FD or F22 (Field Delete)

Action FD is used in both the screen and report design facilities When action FD is used the following steps occur:

- The user is requested to position the cursor anywhere on the field that is to be deleted from the screen/report image.
- When the enter key is pressed the field selected is erased from the screen/report image.

To use function key F22 position the cursor on the field to be deleted and press F22.

Some important points about deleting fields are:

- To select the field to be deleted place the screen cursor anywhere on the first occurrence of the field or its associated label, description or column headings and press the enter key (when using FD) or F22.
- When a field is deleted the field and any associated identification text is deleted.

4.15.24 Action ID or F17 (Identification Change)

This action is used in both the screen and report design facilities. When action ID is used the following steps occur:

- The user is requested to position the cursor anywhere on the field that is to have its identification method changed and press the enter key.
- When a field is selected a pop-up like this will appear in the centre of the screen:

- Change Identification
- : Use cursor to make selection. :
- : Identify by Label
- : Identify by Column Headings
- : Identify by Description
- : No identification is required
- : Ignore request, continue designing :
- : Fnn=Help Fnn=Messages Fnn=Cancel :

:....:

• If the identification method can be changed successfully the field is changed and the screen/report image is then redisplayed with the changes.

:

:

To use function key F17 do the following:

- 1. Position the screen cursor on the field that is to have its identification method changed and then press function key F17.
- 2. Select the type of change required from the resulting pop-up (as previously described).

Some points about changing a field's identification method are:

- To select the field to be changed place the screen cursor anywhere on the field or its associated label, description or column headings and press the enter key or use function key F17.
- Fields in the browse list area of a screen panel cannot have their identification method changed. A field in the browse area has an identification method of "column headings" if it is in the first browse line or an identification method of "label" if it is on the second or subsequent browse line.

• Altering the identification method used for a field will alter the shape of the "block" that contains the field. The section that details how fields are moved (action M or F16) describes these "blocks" in more detail.

For instance a field #ORDNUM (order number) has the following "blocks" associated with each identification method:

When using an identification method of Column Headings:



When using an identification method of Label:

_____ |Order number | 999999999|

When using an identification method of Description:

| | Order number | 999999999 |

When using no identification method:

| | |9999999999| _____

So if field #ORDNUM was in a screen image and currently using an identification method of label it might look like this:

Date : 99/99/99 Company no : 99

| | |Order number | 999999999| |_____| Part number | 99999 Reference : XXXXXXXXXXXX

If the identification method of field #ORDNUM was changed from label to no identification the screen image would look like this:

Date : 99/99/99 Company no | 99 ______ | | |9999999999| |______| Part number | 99999 Reference : XXXXXXXXXXX

Note that when a change to the identification method is made (thus changing the shape of the "block" containing the field) LANSA will attempt to position the new "block" so that its UPPER LEFT HAND CORNER is in the same position as the old "block's" upper left hand corner. This means that it is sometimes necessary to move a field before attempting to change its identification method or after changing its identification method. In the example above it would not be possible to change the identification method of field #ORDNUM to column heading as the new "block" would not fit in the space available. If however, field #ORDNUM was moved (using action M) like this:

Date : 99/99/99 Company no : 99

Order number | 9999999999

Part number : 99999 Reference : XXXXXXXXXXXX

It would now be possible to change the identification method to column heading because the new "block" can be fitted into the space available. The resulting screen image might look like this:

Date : 99/99/99 Company no : 99 | | | | Company | | Order | Part number | 99999 | Number | Reference | XXXXXXXXX |9999999999 | |______|

4.15.25 Pxxx Actions for Manipulating Pop-Up Windows

These actions are used in the screen design facility only.

A number of actions can be specified to manipulate the size, shape and location of pop-up windows while working within the screen design facility. These actions are:

PM Used to move the current pop-up window. Enter action code PM and press enter. Then use the screen cursor to indicate the new position for the upper left hand corner of the pop-up window. The position being indicated with the cursor is the location of the uppermost left most character within the pop-up window border (even if RLTB mode is being used).

PM is an abbreviation of "Pop-Up Move".

- PML Use this action code to move the current pop-up window as far as possible to the left hand edge of the screen image.PML is an abbreviation of "Pop-Up Move Left".
- PMR Use this action code to move the current pop-up window as far as possible to the right hand edge of the screen image.PMR is an abbreviation of "Pop-Up Move Right".
- PMU Use this action code to move the current pop-up window as far as possible to the top (upper) edge of the screen image.PMU is an abbreviation of "Pop-Up Move Up".
- PMD Use this action code to move the current pop-up window as far as possible to the bottom (down) edge of the screen image.PMD is an abbreviation of "Pop-Up Move Down".
- PMAX Use this action code to maximize/maximize the size of the pop-up window currently being displayed. The location of left hand upper corner of the pop-up is not changed, but the width and length of the pop-up are increased to fill all the rest of the screen image. In other words, the pop up is expanded to the right and downwards until it fills all available screen image space.

PMAX is an abbreviation of "Pop-Up Maximize".

PMIN Use this action code to reduce the size of the pop-up window to its theoretical minimum size. The window is "trimmed down" from the right and the bottom until its minimum theoretical size has been reached. The theoretical minimum size only accounts for information in the "body" of the pop-up and it excludes title and function key areas. Later use of PWID/PLEN actions may be required to ensure that the title and function key areas (which may vary at execution time) will be correctly fitted into the pop-up. PMIN is designed to be used before a series of PWID/PLEN actions are used to define the "practical" minimum size of a pop-up window.

Warning If the PMIN action is used last before the screen painter is exited then any subsequent use of the compiler or online full function checker (including the design only option) may reject the POP_UP command with an error stating that the window is actually too small. PMIN is an abbreviation of "Pop-Up Minimize".

- PWID Use this repeating action code to increase the width of the current pop-up window by 1 character. When the desired width has been obtained remove PWID from the action code input area. by typing over it or using "F5=Cancel Pending" function key.
 PWID is an abbreviation of "Pop-Up Width (Increase)".
- PLEN Use this repeating action code to increase the length of the current pop-up window by 1 line. When the desired length has been obtained remove PLEN from the action code input area by typing over it or using "F5=Cancel Pending" function key.

PLEN is an abbreviation of "Pop-Up Length (Increase)".

4.15.26 Action WLn (Move Window Left)

Action WLn is used in the report design facility only. When action WLn is used the following steps occur:

- The current report image "window" is shifted to the left by "n" characters.
- A message is displayed on line 24 indicating what columns of the report image are now being viewed.

- The "window" over the report image is required because very often report image dimensions exceed the dimensions of a workstation screen in both width and length.
- The workstation screen is considered to be a "window" through which the full report image can be viewed. Since the report image is larger than the workstation "window" it is necessary to move the "window" around to view various parts of the report image.
- WLn is the action used to move the workstation window to the left by "n" characters/columns.
- If no value for "n" is specified (i.e. action WL is entered) a default value of (workstation width 1) is used. This default value causes the left most column in the current window to become the right most column in the new window.
- Where the "n" value specified (or defaulted) would cause a movement beyond column 1 of the report, column 1 of the report will be displayed as the first column in the new window.

4.15.27 Action WRn (Move Window Right)

Action WRn is used in the report design facility only. When action WRn is used the following steps occur:

- The current report image "window" is shifted to the right by "n" characters.
- A message is displayed on line 24 indicating what columns of the report image are now being viewed.

- The "window" over the report image is required because very often report image dimensions exceed the dimensions of a workstation screen in both width and length.
- The workstation screen is considered to be a "window" through which the full report image can be viewed. Since the report image is larger than the workstation "window" it is necessary to move the "window" around to view various parts of the report image.
- WRn is the action used to move the workstation window to the right by "n" characters/columns.
- If no value for "n" is specified (i.e. action WR is entered) a default value of (workstation width 1) is used. This default value causes the right most column in the current window to become the left most column in the new window.
- Where the "n" value specified (or defaulted) would cause a movement beyond the width of the report image, the last column of the report image will be displayed as the last column in the new window.

4.15.28 Action Wn (Move Window to n)

Action Wn is used in the report design facility only.

When action Wn is used the following steps occur:

- The current report image "window" is moved so that column "n" of the report image is displayed as the left most column in the "window".
- A message is displayed on line 24 indicating what columns of the report image are now being viewed.

- The "window" over the report image is required because very often report image dimensions exceed the dimensions of a workstation screen in both width and length.
- The workstation screen is considered to be a "window" through which the full report image can be viewed. Since the report image is larger than the workstation "window" it is necessary to move the "window" around to view various parts of the report image.
- Wn is the action used to move the workstation window to a specific column ("n") in the report image.
- If no value for "n" is specified (i.e. action W is entered) a default value of 1 is assumed.
- Where the "n" value specified (or defaulted) would cause a display beyond the width of the report image, the last column of the report image will be displayed as the last column in the new window.

4.15.29 Action W? (Display Current Window Details)

Action W? is used in the report design facility only.

When action W? is used:

A message is displayed on line 24 indicating what columns of the report image are being viewed via the current "window".

- The "window" over the report image is required because very often report image dimensions exceed the dimensions of a workstation screen in both width and length.
- The workstation screen is considered to be a "window" through which the full report image can be viewed. Since the report image is larger than the workstation "window" it is necessary to move the "window" around to view various parts of the report image.
- W? is the action used to inquire about what columns of the report image are being displayed in the current screen window.

4.15.30 Action Run (Roll Image up n Lines)

Action RUn is used in the report design facility only.

When action RUn is used the following steps occur:

- The current report image is "rolled upwards" by "n" lines within the current workstation window.
- A message is displayed on line 24 indicating what lines of the report image are now being viewed.

- The "window" over the report image is required because very often report image dimensions exceed the dimensions of a workstation screen in both width and length.
- The workstation screen is considered to be a "window" through which the full report image can be viewed. Since the report image is larger than the workstation "window" it is necessary to move the "window" around to view various parts of the report image.
- RUn is the action used to move the report image upwards by "n" lines within the current window.
- If no value for "n" is specified (i.e. action RU is entered) a default value of (workstation length 2) is used. This default value causes the lowermost line in the current window to become the uppermost line in the new window.
- Where the "n" value specified (or defaulted) would cause a movement beyond the length of the report image, the last line of the report image will be displayed as the last line in the new window.

4.15.31 Action RDn (Roll Image down n Lines)

Action RDn is used in the report design facility only.

When action RDn is used the following steps occur:

- The current report image is "rolled downwards" by "n" lines within the current workstation window.
- A message is displayed on line 24 indicating what lines of the report image are now being viewed.

- The "window" over the report image is required because very often report image dimensions exceed the dimensions of a workstation screen in both width and length.
- The workstation screen is considered to be a "window" through which the full report image can be viewed. Since the report image is larger than the workstation "window" it is necessary to move the "window" around to view various parts of the report image.
- RDn is the action used to move the report image downwards by "n" lines within the current window.
- If no value for "n" is specified (i.e. action RD is entered) a default value of (workstation length 2) is used. This default value causes the uppermost line in the current window to become the lowermost line in the new window.
- Where the "n" value specified (or defaulted) would cause a movement beyond line 1 of the report image, the first line of the report image will be displayed as the first line in the new window.

4.15.32 Action SP (Change Space before/after Values)

Action SP is used in the report design facility only.

When action SP is used the following steps occur:

- The "space before" and "space after" values associated with the DEF_XXXXXX command currently being edited are displayed on line 24 of the workstation and can be changed if desired.
- If the new values are acceptable they are changed within the DEF_XXXXX command and the current report image window is re-displayed to reflect the changes.

A point about altering space before and after values are:

• Space before and space after values can only be changed when editing a DEF_LINE or DEF_BREAK print definition command. Attempting to use action SP when editing a DEF_HEAD or DEF_FOOT command will result in an error message being displayed on line 24.

4.15.33 Canceling Pending Screen/Report Design Actions

You may change your mind after initiating an "action" in the screen or report design facilities.

For instance an action of "FD" (field delete) is entered and then you decide not to delete the field. Or an action of "TI" (text insert) is entered and then you decide no text is required.

In such cases, to cancel the pending action do either of the following:

- Enter a new action code into the action field to initiate a new action, thus canceling the current action even if it has not completed.
- Change the action code field to blanks. This will cancel the current action, even if it has not completed, and then request that a new action be specified.
- Use function key F5 which is functionally identical to entering blanks into the action field.

4.15.34 Some Tips for Inputting / Changing RDML Programs

This section provides some techniques that can be used to make the input or amendment of RDML programs easier, quicker and more accurate.

Use the Prompter

When it is necessary to code a number of commands that may span multiple lines it is easier to use the command prompter than input the lines directly. For example if the commands CHANGE, CHANGE, DISPLAY, INSERT, CHANGE, GOTO had to be coded with all their associated parameters it is easiest to enter them as:

No	_Act_Lbl_Com	nandParameters
1 P	CHANGE	
2 P	CHANGE	
3 P	DISPLAY	
4 P	INSERT	
4 P	CHANGE	
5 P	GOTO	

When the enter key is pressed a fully formatted prompt will be displayed for each command. All that has to be done then is "fill in the boxes" for each command and let the command prompter automatically assemble and format the resulting RDML commands which may span multiple input lines.

Handling Quotes

Some RDML commands require that associated parameters appear as "quoted strings" because LANSA uses the IBM i operating system command definition and prompting facilities.

For example, to increment field #COUNT by 1 the correct format is:

```
CHANGE #COUNT '#COUNT + 1'
or
CHANGE #COUNT ('#COUNT + 1')
```

or

```
CHANGE #COUNT '(#COUNT + 1)'
but NOT
```

```
CHANGE #COUNT (#COUNT + 1)
```

This is because the IBM i command facilities demand that a command parameter be enclosed in quotes if it contains imbedded blanks. In this case the string "#COUNT + 1" definitely contains imbedded blanks and thus must be enclosed in quotes (i.e. made into a "quoted" string).

When LANSA processes the command only the part between the quotes (but not the quotes themselves) are passed to LANSA by the operating system.

The matter is complicated even further if you wish to use quotes within a "quoted" string. This situation usually arises when coding IF or CASE conditions.

The rule for using quotes inside a quoted string is: use 2 quotes instead of just one.

For instance, to check if #FIELD contains a lowercase "a" you would have to code:

IF '#FIELD = "a'"

What is passed to LANSA by the IBM i operating system command processor as the expression associated with the IF command is actually:

#FIELD = 'a'

because the operating systems do not pass the outer quotes and replaces occurrences of 2 quotes within the string with just one quote.

However, the handling of quotes within LANSA can be made much easier by following 2 simple guidelines:

- Only use quotes inside a quoted string when absolutely necessary.
- Use the formatted prompting facilities to input complex quoted strings.

With regard to point 1, LANSA does not require that alphanumeric literals be quoted. Thus the following are identical conditions because alphanumeric literals that are not enclosed in quotes are converted to uppercase:

```
IF '#FIELD = A'
```

```
IF '#FIELD = a'
```

```
IF '#FIELD = "A"
```

Only use quotes around alphanumeric literals if the test involves lowercase characters. For instance the test for a lowercase "a" in #FIELD:

```
IF '#FIELD = "a'"
```

With further regard to point 2, you will find that the formatted prompting facility will automatically insert the required outer quotes. For instance if you

prompt an IF command and enter the condition as:

#FIELD = A

then the prompter will automatically re-format the condition so that it is a valid "quoted" string. The final version of the command created by the prompter would look like this:

IF COND('#FIELD = A')

The same applies when it is necessary to use quotes within the expression. For instance if you specify to the prompter the following condition:

#FIELD = 'a'

then it will re-format the command automatically and insert the necessary inner and outer quote symbols. The command created by the prompter would look like this:

IF COND('#FIELD = "a'")

Remember the HELP Key

A common problem that occurs when coding RDML programs is the need to look up the definitions of fields or files in the LANSA data dictionary. Questions like:

- Is field #ADDR1 35 characters long or 30 characters long?
- Is field #PRODNO alpha or numeric?
- What are the keys of logical file CUSMSTV6?
- What are the fields in file CUSTMST?
- etc, etc....

most frequently arise when coding RDML programs.

To answer any of these questions PRESS THE HELP KEY and use the help facilities to look up the answer. The HELP key can be used from any screen in the LANSA system. Refer to *Messages and the Help Key* for specific details about the HELP key facilities and how they are used.

Read All the Messages and the Second Level Text

When inputting or amending an RDML program or running the function checker messages may appear on line 22/24 of the screen. When a message appears remember the following points:

• If there is a "+" sign at the end of the message then more messages follow. These should be reviewed using the messages function key or the ROLL keys. • If you don't understand exactly what the message refers to, read the associated second level text which will explain the message in much more detail. To do this place the screen cursor on the same line as the message and press the HELP key.

For detailed information about messages and message handling within LANSA refer to the relevant section in the *Messages and the Help Key*.

4.16 Rename a Function

4.16.1 Steps to Rename a Function within a Process

4.16.2 When you Rename a Function

4.16.1 Steps to Rename a Function within a Process

The steps involved in the renaming of a function from a process can be visualized as follows:



4.16.2 When you Rename a Function

When a function has been selected for renaming a display similar to the following example will be presented:

DC@P301401 Rename Existing Function

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

To rename the function within the process enter a new function name. The function name must not already exist within the process, and if the function being renamed is a *DIRECT function, the new function name must not exist within the partition.

All records associated with the function will be renamed to the new function name, including any compiled objects such as programs and display files (in the case of *DIRECT functions).

Note: The function will be flagged as unusable once it has been renamed. The process will also be flagged as not compiled.

4.17 Delete a Function

4.17.1 Steps to Deleting a Function from a Process4.17.2 Confirm a Function Is to Be Deleted

4.17.1 Steps to Deleting a Function from a Process

The steps involved in the deletion of a function from a process can be visualized as follows:



4.17.2 Confirm a Function Is to Be Deleted

When a function has been selected for deletion a display similar to the following example will be presented:

DC@P301201 Delete Existing Function

Process : CUSTOMERS Work with Customers Function : Maintain Customers

Confirm that this function is to be deleted

Delete this Function YES, NO

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

To delete the function from the process enter YES. To avoid deleting the function enter NO.

When attempting to delete a function from a process that uses the "anticipated usage" HEAVY option the system may respond with a message indicating that not all IBM i objects associated with the function could be deleted and thus the function could not be deleted.

This occurs because the function leaves the associated IBM i display file open and LANSA cannot allocate it for deletion.

In such cases ask all users who have been using the process to sign off and sign on again. This will close the display file and release it for deletion. Remember that the workstation attempting to delete the function may be one of the workstations that has the associated display file open.

Once all users have closed the display file attempt to delete the function again.

For more details of the "anticipated usage" option refer to 4.4 Create a New Process Definition or 4.23 Miscellaneous Process Details.

4.18 Function Control Table

The function control table is a table that controls how and when the various functions within a process are used and the ordering sequence that the functions should be in when displayed on the process menu.

Every function defined in a process has an entry in the function control table. It is automatically created with default values when the function is created and cannot be removed. However it can be changed from its default values to suit the specific requirements of a user application of LANSA.

Every entry in the function control table contains the following elements:

- The name of the function to which the entry belongs.
- The description of the function to which the entry belongs.
- An indication of whether or not the function should be displayed on (and therefore directly accessible from) the process's main menu.
- The Sequence that the Function should be ordered in on the process menu when displayed.
- The name of the associated "default next function".
- A list of up to 20 "allowable next functions".

The function control table associated with a process called EXAMPLE which has 3 functions called FUNC1, FUNC2 and FUNC3 would look like this before any user changes were made to it:

Func	Dis	on Def	fault		
Name I	Desc I	Menu	Next Func	Allowable	next funcs
FUNC1	Function	1 YES	MENU	*ANY	
FUNC2	Function	2 YES	MENU	*ANY	
FUNC3	Function	3 YES	MENU	*ANY	

In the section that dealt with "Creating a new function" a list of "reserved" function names was mentioned. This was a list of function names that were "reserved" for internal LANSA use and could not be used as valid user created function names:

Name Reserved Meaning / Description

MENU Display process main menu

EXIT Exit from LANSA

HELP Display process HELP te	ext
------------------------------	-----

SELECT Select next function from list of allowable function

- EOJ End all batch processing
- RETRN Return control to calling process or function
- ERROR Abort process with an error
- *ANY Any function name

Taking the function control table for process EXAMPLE and the list of reserved function names together it can be seen that what the table is actually specifying to LANSA is:

- When using FUNC1 the most common (or default) next function is to display the process's main menu.
- After using FUNC1 the user can go to any other function in the process.
- Exactly the same conditions apply to using FUNC2 and FUNC3.
- All of the functions are to appear on the process's main menu and are therefore directly and independently accessible from the menu.

However, if the function control table for process EXAMPLE was changed as follows:

Func	Dis	on Def	fault		
Name I	Desc l	Menu	Next Func	Allowable	next funcs
FUNC1	Function	1 YES	FUNC2	FUNC2	
FUNC2	Function	1 2 NO	FUNC3	FUNC3	FUNC1 SELECT
FUNC3	Function	1 3 NO	MENU	MENU	

The following conditions now apply:

- Once FUNC1 is used it is only possible to go to FUNC2.
- In FUNC2 we can only go to FUNC3 or back to FUNC1 and the usual thing to do next is to go to FUNC3. In addition, the "reserved" function SELECT can be used. This will result in a list of the allowable next functions (FUNC1 and FUNC3) being displayed as a "sub menu". From this the user can select

the desired next function.

- In FUNC3 we can only go back and re-display the process's main menu.
- Only FUNC1 is to appear on the process's main menu. Thus only FUNC1 can be accessed from the process main menu. FUNC2 can only be accessed from FUNC1 and FUNC3 can only be accessed if FUNC2 (and therefore FUNC1) have been accessed first.

Hopefully this example illustrates the main purpose of the function control table which is to provide a "path" through the functions.

The path may be "enforced" (e.g.: From FUNC1 you must go to FUNC2) or it may be "suggested" (e.g.: from FUNC2 you would normally go to FUNC3, but can go to FUNC1 if desired).

4.18.1 Function Control Table and Menu Selection Style: FUNCTION

4.4 Create a New Process Definition describes the menu selection style options available in LANSA. This allows the style of menu selection used for the process's main menu to be specified as one of the following (ignoring ACT/BAR which causes the process to act as an action bar):

- SAA/CUA Design menu according to IBM's SAA/CUA guidelines. This is the recommended style for LANSA. As such, much of this section of the manual is now redundant, and is retained purely for the continuance of older style systems.
- CURSOR Select entries using the screen cursor.
- NUMBER Select entries by entering the associated number.

FUNCTION Select entries by entering the "next function" name.

While the function control table can be used effectively with all 3 menu selection styles it is more "visible" and more flexible when using the FUNCTION selection style and is thus easiest to explain.

FUNCTION style selection basically means that you indicate what you wish to do next by entering the name of the function into a field at the bottom of the current screen.

If you are using FUNCTION style selection then at the bottom of the process's main menu and every screen used by the functions associated with the process a field called "the next function field" will appear like this:

Next function : FUNC2

F1=Help F3=Exit F12=Cancel F14= Messages

What this field is asking is "When you have finished here what do you want to do next?"

The "next function field" will always be pre-filled with the default next function from the function control table. The value is "suggesting" what you would

normally do next (and it may be the only thing that can be done next) but it can be changed if desired.

This is what makes FUNCTION style selection more flexible than the other selection styles. It effectively allows a choice about what selection can next to be made from within a function as well as from the process's main menu.

This means that it is possible to go from FUNC1 to FUNC2 then to FUNC3 without once displaying the process's main menu.

Of course the process control table can be used in exactly the same way when using selection styles CURSOR or NUMBER. The user can go directly from FUNC1 to FUNC2 and then to FUNC3 without displaying the process's main menu. The difference is that because the "next function" field is not displayed when using these menu selection style he/she is forced to follow the path and cannot change it.

When using FUNCTION style selection in FUNC1 the "next function field" might be "suggesting" next function FUNC2. But, because it is displayed, it can be changed to something else (FUNC17 for example) thus altering the path that is being followed. Whether or not FUNC17 is actually a valid choice depends upon the allowable next functions list in the associated function control table entry.

4.18.2 What Do the Reserved Functions Do?

The list of function names that are "reserved" and cannot be used as the name of a user defined function has already been described as being the names:

Name	Reserved Meaning / Description
MENU	Display process main menu
EXIT	Exit from LANSA
HELP	Display process HELP text
SELECT	Select next function from list of allowable function
EOJ	End all batch processing
RETRN	Return control to calling process or function
ERROR	Abort process with an error
*ANY	Any function name

While it is not possible to use any of these names as a user created function name it is possible to use any of these names in the process control table. It is easiest to think of these "reserved" functions as "real" functions that are pre-defined and exist in every process that is created. They can be used in exactly the same way as any user defined function.

The effect of using these "reserved" functions is as follows:

- MENU : Causes the process's main menu to be displayed. The menu will contain an entry for every user defined function in the process, every "reserved" function in the process and any additional menu options. Specifying a "next function" of MENU or using the MENU function key are identical operations.
- EXIT : Causes an immediate exit from the LANSA system. Control is returned to the application that invoked LANSA. Specifying a "next function" of EXIT or using the EXIT function key are identical operations.
- HELP : Causes any process HELP text that exists to be displayed. From this display any associated function or field HELP text can be displayed.
Specifying a "next function" of HELP or using the HELP key are identical operations.

- SELECT : Causes a list of all allowable next functions to be displayed. The effect is a "mini menu" of valid next functions. Note that MENU causes all next functions to be displayed whereas SELECT only displays the allowable next functions.
- EOJ : Causes a process executing in a batch environment to end all processing and return control to the application that invoked LANSA. EXIT and EOJ are identical in processing and can be used interchangeably. The EOJ function is provided for convenience only.
- RETRN : Causes the current process to end. Control is returned to the process, function or user application program that invoked the process. Specifying a "next function" of RETRN is equivalent to executing the RETURN command in an RDML program mainline. Refer to the *Technical Reference Guide* for more details of the CALL and RETURN commands.
- ERROR : Cause a process to end in error and display/print error details. Control is then returned to the application that invoked LANSA. Use in user defined function control tables is not recommended.
- *ANY : is not a function at all. It can only be used in the "allowable next functions" list in a function control table. It is provided to save having to enter the names of all possible next functions when all next functions are valid.

4.18.3 Experiment with Process Control Table

The process control table associated with any process can be changed dynamically. There is no need to recompile the associated functions.

As such it is easy to experiment with the process control table entries to learn how they work and what can be achieved using them.

No harm can result from altering process control table entries. The worst thing that can happen is that you will lock yourself into a function and not be able to get out.

The actual procedures used to review and change the process control table entries are described in 4.18.4 Steps to Review or Change the Function Control Table.

4.18.4 Steps to Review or Change the Function Control Table

The steps involved in review or change of the function control table can be visualized as follows:



4.18.5 Function Control Table Maintenance

When the option to review or change the process control table is selected from the process definition menu a screen similar to the example following will be presented:

DC@P301501	Functi	on Contro	l Table	
Process : FUNCFLOW	FUN	CTION F	LOW TE	ST
Function Description FUNC1 Function 1 Default next function	ע ו : Fl	usplay on pro <u>YES</u> UNC2	$\frac{1}{5}$	ı Seq <u>001</u>
Allowable next funct	ions:	FUNC2	FUNC3	SELECT
FUNC2 Function 2 Default next function	ı:Fl	NO UNC3		
Allowable next funct	ions:	FUNC3	SELECT	

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

From the *Function Control Table* screen it is possible to:

- Review the entries in the process control table. Use the roll keys to roll backwards and forwards through the entries.
- Make changes to the process control table entries using the CHANGE function key.

Input Options

The following options apply to reviewing or changing entries in a process control table.

Function

Indicates the function to which this entry in the function control table belongs/is associated with. This is an output field and cannot be changed.

Description

Specifies the description associated with the function above. This is the description of the function that was specified when the function was created and is the description of the function that will be displayed on the process's main menu.

The description can be changed if desired, but a description must always exist for each and every function in the function control table.

Display on Process Menu

Specifies whether or not this function should be displayed (and therefore be accessible from) the process's main menu. Allowable values are YES indicating that the function should be displayed on and be accessible from the process's main menu, or NO indicating that the function should not be displayed on or be accessible from the process's main menu.

Menu Seq (Menu Sequence)

Specifies the order of the functions on the Process Menu If the function is to be displayed on the process menu then the order in which it appears on the process menu can be governed by an entry in this alpha field.

Note: This field is an alphanumeric field. Care should be taken when ordering the functions. See examples in the next section.

Default Next Function

Specifies the function that is "usually" invoked after this function has been completed. Allowable values are any function defined in the process or any of the "reserved" function names except for *ANY which makes no sense in this context.

Allowable Next Functions

Specifies up to 20 functions that are allowed to be invoked after this function has been completed. Allowable values are any function defined in this process or any of the "reserved" function names.

The use of special value *ANY is recommended if any function can be invoked

rather than listing all function names.

If *ANY is used it should be the only entry in the list.

If *ANY is not used, ensure that the "default next function" is included into the list.

Lists that do not include "reserved" names EXIT and MENU are effectively disabling the use of the EXIT and MENU function keys. The EXIT and MENU function keys are processed by simulating the entry of "next functions" of EXIT and MENU respectively.

Example - Function Controls

Assuming that a process has four functions:

HEADER Input order header details.

LINES Input line items.

SUBMIT Submit order for processing.

HOLD Hold order for processing later.

The function control table can be used to allow the input of order header details and then allow a choice of submitting or holding the order. The entries in the function control table might look like this:

Function Display on Default Name Process menu Next Func Allowable next funcs HEADER YES LINES LINES LINES NO SELECT SELECT SUBMIT HOLD SUBMIT YES MENU **MENU** HOLD NO MENU **MENU**

Note the following points about this table:

- Only HEADER and SUBMIT actually appear on the process's main menu. HEADER allows the user to "start" on the path required to create an order. The path must eventually terminate with functions HOLD or SUBMIT. SUBMIT is accessible from the process's main menu to allow the submission of orders that were previously held.
- When the user completes using function LINES, the default next function is the "reserved" function SELECT which will display a "mini menu" of functions SUBMIT and HOLD. This is allowing the user to choose which of the 2 functions he/she wishes to use.

Example of Using Menu Sequence in Function Control Table

Assume that HOLD function will be displayed on the resulting process menu. However we want it to be the first entry on the process menu.

	Display on	Menu		
Function 1	Description pi	rocess menu	Seq	
HEADER	Input order header details	YES		_2
Defa	ult next function : LINES			
Allow	wable next function : LINE	S		
HOLD	Hold order for processing la	ater YES	1	

Default next function : MENU Allowable next function : MENU

Specifying the 'Menu Sequence' as shown in the above example will cause the function HEADER to be displayed first, and this is not the order that was required.

To ensure that the order is correct when ordering by number, place the digit in the right most part of the field or precede number with zeros.

	Disp	lay on M	lenu	
Function D	escription	proce	ess menu	Seq
HEADER	Input order head	er details	YES	000002
Defaul	t next function	: LINES		
Allowa	able next function	n : LINES		

HOLD Hold order for processing later YES 000001 Default next function : MENU Allowable next function : MENU

OR

Use a word that is a combination of alpha characters and numeric characters to determine the ordering of the functions. e.g. A01 ... A10, OPT001 ... OPT010, SEQ001 ... SEQ010, etc...

Display onMenuFunctionDescriptionprocess menuSeqHEADERInput order header detailsYESOPT002

Default next function : LINES Allowable next function : LINES

HOLDHold order for processing laterYESOPT001Default next function : MENUAllowable next function : MENU

4.19 Special Entries on the Process Menu

"Special entries" on the process menu are IBM i CL (control language) commands that have been defined to LANSA using this facility.

Any IBM i CL command that can be executed via IBM supplied programs QCMDEXC or QCAEXEC, can be included onto a process's main menu.

In addition, the command can be "prompted" at the time it is being used to allow you to specify or change the associated parameters.

CL commands that allow you to display or control your own jobs are most commonly included on the process's main menu. Some of these include:

5	-
WRKSBMJOB	Work with submitted jobs
WRKJOB	Work with current job
WRKOUTQ	Work with output queue
WRKJOBQ	Work with job queue
DSPMSG	Display message queue
SNDMSG	Send a message

For more information about the CL commands available on the IBM i and the programs QCMDEXC and QCAEXEC that are used to execute them, refer to the appropriate IBM supplied manual.

4.19.1 Steps Required to Rev/Chg Special Entries on the Process Menu

The steps involved in the definition, review or change of special entries on the process's main menu are:



4.19.2 Special Entries on the Process Menu Maintenance

When the option to define, review or change special entries on the process's main menu is chosen from the process definition menu a screen similar to the example following will result:

DC@P301601 Additional Menu Entries

Run Description to display on n	n Time nenu	Prompt	Comman	d to be executed

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

From the Additional Menu Entries screen it is possible to:

- Review the existing special menu entries that are defined for the process. Use the ROLL keys to roll backwards and forwards through the menu entries.
- Define, change or delete special menu entries. In all cases use the CHANGE function key. To change an existing entry type new details over the existing details. To define a new entry use the ROLL UP key to roll to the end of the

list of existing entries and add the new entry to the end of the list. To delete an entry blank out all associated details.

Input Options

The following input options apply to defining or changing a special entry on the process's main menu.

Description to Display on Menu

Specifies the description that is to be associated with the command. This description is displayed on the process's main menu and is used when choosing to execute the IBM i command. A description of the command is mandatory.

Run Time Prompt

Specifies whether or not the command should be "prompted" when it is used from the process's main menu. Allowable values are Y (prompt the command) or N (do not prompt the command).

When used, the prompt is the standard IBM CL command style prompting. Refer to the appropriate IBM supplied manual for more details.

Command to be Executed

Specifies the IBM i command that is to be executed when the entry is chosen from the process's main menu. The command should be entered exactly as it would be on the command entry display screen. See following examples for details.

Examples - Special Menu Entries

The following examples apply to special menu entries that are to appear on a process's main menu.

Example 1

Define 2 special menu entries to display jobs submitted by the current user and jobs submitted from the workstation:

Run TimeDesc to display on menu____Prompt___CommaWork jobs subd. by userNWRKSBMJOB *USERWork jobs subd. from wrkstnNWRKSBMJOB *WRKSTN

Example 2

Define 2 special menu entries. The first is to display a nominated message queue and the second is to display a nominated job:

Run Time	Desc	to display on menu_	Prompt	Commai
Display a nominated msgq	<u>Y</u>	<u>DSPMSG</u>		
Display a nominated job	Y	<u>DSPJOB</u>		

These special menu entries can also be used to show options on the main process's menu such as:

• submit a General ledger print to batch

SBMJOB JOB(GLPRINT) JOBD(QBATCH) CMD(CALL GLA0742) OUTQ(QPRINT2)

• display the General Ledger system main menu CALL GLA0001

4.20 Multilingual Special Entries

In a multilingual partition, multilingual special entries may be specified for each language that is defined within the partition. Multilingual special entries are the descriptions to display on the process menu.

4.20.1 Steps to Review or Change Multilingual Special Entries

4.20.2 Multilingual Special Entries Maintenance

4.20.1 Steps to Review or Change Multilingual Special Entries

The steps involved in reviewing or changing multilingual special entries are as follows:



4.20.2 Multilingual Special Entries Maintenance

When the option to review or change multilingual special entries is chosen from the process definition menu is chosen, a screen similar to this will result:

DC@P300701 Multilingual Options - LRTB

Process : PSLSYS

Sequence Description to display	on menu Language
1 Zend een boodschap	(DUT)
1 Send a message	(ENG)
1 Senden einer Mitteilung	(DEU)

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

From this screen it is possible to:

- Review the multilingual information associated with special entries.
- Change the information associated with multilingual special entries. Use the CHANGE key to place the screen in change mode and make the desired changes.

Input Options

The following input option is used to change a special entry's multilingual attributes:

Description to display on menu

Specifies the description that is to be associated with the special option command. This description will be displayed on the process's main menu and is used when choosing to execute the IBM i command. A description of the special option command for all languages is mandatory.

4.21 Other Processes on the Process Menu

A process is essentially a menu of functions.

A process named PROC01 with 3 functions named FUNC01 to FUNC03 could be visualized like this:



If another process named PROC02 existed and had 2 associated functions called FUNC04 and FUNC05, then it could be visualized like this:



|_____|

Both PROC01 and PROC02 can be independently invoked from within LANSA. However, it is possible to "attach" PROC02 to PROC01 using the facilities described in the following sections.

If PROC02 was "attached" to PROC01, then PROC01 would now be visualized like this:



It is also possible to "attach" FUNC04 and FUNC05 directly to PROC01 (even though they are part of process PROC02). If this was done PROC01 could be visualized like this:



In fact, process PROC02 may have a process(s) "attached" to it. For instance, if process PROC03 was attached to process PROC02 and process PROC02 was itself attached to process PROC01, then process PROC01 could be visualized like this:





It can be seen from these examples that by "attaching" a process to another process a "hierarchy" of process menus can be created.

Some other points to note about this facility are:

- Multiple processes may be attached to a process.
- Multiple functions (from other processes) may be directly attached to a process.
- A process (or any of its associated functions) may be attached to itself.
- If a process B is attached to process A, then it is possible to attach process A to process B.
- Processes can be built into a "hierarchy" using this facility. There is no limit to the "depth" of the hierarchy that can be defined, but when actually using a

process the "depth" being used must not exceed 9 processes. If it does an error message will be issued indicating that it is not possible to go any deeper in the process "hierarchy" and that the required process should be accessed via a different route.

- Processes and functions can be attached and detached dynamically. There is no need to recompile any associated functions after attaching or detaching them from another process.
- When a process or function is deleted all associated attachments are also automatically deleted.

4.21.1 Steps to Review/Change Other Processes on the Process Menu

The steps involved in the definition, review or change of other processes on the process's main menu are:



4.21.2 Other Processes on the Process Menu Maintenance

When the option to define, review or change other processes on the process's main menu is chosen from the process definition menu a screen similar to the example following will result:

DC@P302001	Processe	s on Process Menu	T:TT:TT
Process : XXXXXX	xxxx xx	****	xxxxxxxxxxxx
Display Attached Sequence Process	Attached Function	Description XXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXX	XXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXX

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

From the *Processes on Process Menu* screen it is possible to:

- Review the existing other process entries (i.e. "attached" processes) that are defined for this process. Use the ROLL keys to roll backwards and forwards through the displayed list.
- Define, change or delete other process entries. In all cases use the CHANGE function key. To change an existing entry type the new details over the existing details. To define a new entry use the ROLL UP key to roll to the end of the list of existing entries and add the new entry to the end of the list. To delete an entry blank out all associated details.

Input Options

The following input options apply to defining or changing other processes on a process's main menu.

Display Sequence

Is used to nominate the relative order in which the attached process or function should be displayed. Enter a number in the range 1 to 999 that indicates the required display order of the process or function relative to other attached processes or functions.

Attached Process

Indicates the name of the process that is to be attached to the current process (i.e. the one being worked on).

The name of the process to be attached must be nominated when attaching a process or when directly attaching a single function.

Note that the process nominated does not have to exist. If it does not currently exist a warning message will be issued.

Attached Function

Indicates either the name of the function that is to be directly attached to the current process (i.e. the one being worked on) or indicates that all functions in the process should be attached.

To directly attach a function to the process specify the name of the function.

To attach all functions (i.e. attach the entire process) specify *ALL as the attached function name.

Note that the function nominated does not have to exist. If it does not currently exist a warning message will be issued.

Description

This output field indicates the description of the attached process or function that will appear on the current process's main menu (i.e. the process being worked on).

If a single function from another process has been attached then the description associated with that function will be displayed.

If an entire process (attached function = *ALL) has been attached then the description associated with the process will be displayed.

If the attached process or function does not currently exist then a message will be displayed in the description area indicating that the description is not currently available.

Examples - Other Processes

The following examples apply to other processes that are to appear on (i.e. be attached to) a process's main menu.

Example 1

The introduction to this section described two processes called PROC01 and PROC02 and indicated that they could be "attached" to form a hierarchy that looked like this:



To do this, the definition of PROC01 would have to be altered using the "other

process" facility like this:

Display Attached Attached Sequence Process Function Description 10 PROC02 *ALL

Example 2

The introduction to this section also described how functions FUNC04 and FUNC05 from process PROC02 could be directly attached to process PROC01 in a hierarchy like this:



To do this, the definition of PROC01 would be have to be altered using the "other process" facility like this:

Display Attached Attached

Sequer	nce Process	Function	Description
10	PROC02	FUNC04	
20	PROC02	FUNC05	

Example 3

The introduction to this section described a more complex hierarchy that involved three processes like this:





To do this, the definition of PROC01 would be have to be altered using the "other process" facility like this:

DisplayAttachedAttachedSequenceProcessFunctionDescription10PROC02*ALL

Then the definition of process **PROC02** would have to be altered like this:

Display Attached Attached Sequence Process Function Description 10 PROC03 *ALL

4.22 Process Parameters

Note: Process Parameters should never be used. They exist only for backward compatibility.

Parameters are fields that are passed into a process from an "external" source.

The "external" source can be a data entry screen, which is presented by LANSA, into which the user keys values for the parameters. It can also be a user written application program that passes the parameter values into LANSA.

Once a parameter is defined for a process it is accessible to every function within the process. Each parameter defined in a process is assigned a "symbolic" name that allows it to be easily accessed by the RDML commands associated with a function. The symbolic names have the form *UPnn where "nn" is the parameter number in the range 01 -> 10. Generally a parameter's symbolic name can be used anywhere in an RDML command that a normal field name or literal value could be used.

Some specific instances where Process Parameters **cannot** be used are:

- When a function in the process uses FUNCTION OPTIONS(*DIRECT);
- When the process must execute in Visual LANSA, because Visual LANSA functions require FUNCTION OPTIONS(*DIRECT);
- When a function in the process is RDMLX, because RDMLX functions require FUNCTION OPTIONS(*DIRECT)

In previous versions of LANSA, parameters may have been used in the following situations:

- To pass information into a process that is submitted to run in batch. An alternative is to use a Function and use one of the inter-function methods described below.
- To pass information from a user application into an interactive process. An alternative is to use one of the methods for communicating between jobs described below.
- Some specific instances where process parameters should not be used are:
- As a quick way of obtaining information from the user. Instead use the RDML command REQUEST in the associated function to get the information. This approach is more aesthetic and more consistent.
- As a means of exchanging information between functions. A much more powerful and flexible means of exchanging information is to use the RDML

command EXCHANGE or the EXCHANGE parameter on the TRANSFER command.

• As a means of exchanging information between separately executing jobs. Instead use a data queue, data area or system variable. Visual LANSA also provides a user defined parameter on the command line - see a description of the UDEF parameter.

For more information about the RDML commands mentioned, refer to **REQUEST, EXCHANGE** and **TRANSFER** in the *Technical Reference Guide*.

4.22.1 Steps to Define, Review or Change Process Parameters

The steps involved in the definition, review or change of process parameters are as follows:



4.22.2 Process Parameters Maintenance

When the option to define, review or change a process's parameters is chosen from the process definition menu a screen similar to the following example will result:

DC@P3	01701	Paran	neters Pass	sed to Process
Process	: XXXXX	XXXXX	XXXXXX	*****
User		Leng	gth	
Parame	ter Symbo	olic	/Total I	Dec
Numbe	r Name	Type	Digits	Pos Description
01	*UP01			
02	*UP02			
03	*UP03			
04	*UP04			
05	*UP05			
06	*UP06			
07	*UP07			
80	*UP08			
09	*UP09			
10	*UP10			
Fnn=Hel	p Fnn=Ex	it Fnn=C	ancel Fnn	n=Messages

From the *Parameters Passed to Process* screen it is possible to:

- Review the existing parameters that are defined for the process. Note that although the symbolic names are shown for all 10 parameters, only the parameters that have an associated type and length are actually defined within the process.
- Define, change or delete process parameters. In all cases use the CHANGE key to place the screen in change mode. To change the definition of a parameter, enter the new details over the existing details. To delete the

definition of a parameter blank out the associated details. To define a new parameter, specify the new details beside the first unused symbolic name.

Process parameters must be used sequentially. To assign 3 process parameters you must use the symbolic names *UP01, *UP02 and *UP03. It is not possible to use symbolic names *UP01, *UP04 and *UP09.

Note that when a process parameter is changed or deleted no check is made for references to (or contextual usage of) the parameter in any associated RDML commands. Changes or deletions may cause errors to occur when compiling the associated RDML commands. In such cases, correct the associated RDML commands and submit the compile request again.

When changes are made to the parameters defined in a process all functions in the process are flagged as "requiring re-compilation". All functions in the process must be re-compiled before they can be used again.

Input Options

The following input options apply to defining or changing the parameters associated with a process:

Parameter Number

Output field. Specifies the number associated with the parameter. From 0 to 10 parameters can be defined for any process.

Symbolic Name

Output field. Indicates the symbolic name that will be associated with the parameter if it is used. This is the name that is used in associated RDML commands to reference the parameter.

Туре

Specifies the type of parameter that is to be defined. Allowable values are:

- ALPHA The parameter is to be alphanumeric.
- NUMBER The parameter is to be numeric. If this option is used the parameter is in fact defined as a packed variable as this format is easiest to pass.

Length / Total Digits

Specifies the length for type ALPHA parameters or the total number of digits (including decimals) for NUMBER parameters.

For type ALPHA parameters the length specified must be in the range 1 to 256.

For type NUMBER parameters the total number of digits must be in the range 1 to 15 and not less than the number of decimal positions specified.

Dec Pos

Specifies the number of decimals for type NUMBER parameters only. The number of decimals specified must be in the range 0 to 9 and less than or equal to the total number of digits specified.

Description

Specifies a short description that is to be associated with the parameter. If it is necessary for LANSA to display a data entry screen for specification of parameter values this description will be displayed to aid the user. A brief description of every process parameter that is defined must be supplied.

For more details on how and when LANSA will request that parameter values be specified refer to 4.10 Use/Execute a Process from a Workstation and 4.11 Submit a Process to Batch.

4.23 Miscellaneous Process Details

Miscellaneous process details are all the parts of the process definition that do not fit into any of the other facilities available from the process definition menu.

All of the details associated with a process that are classified as "miscellaneous" were originally specified when the process was first created. The "Review or change miscellaneous process details" facility is provided to allow these details to be changed. They include:

- The process description
- The anticipated usage option
- The menu selection style
- The optimise remote communications flag
- The enable for the web flag
- The function key assignments (in non-SAA/CUA processes only)
4.23.1 Steps to Review or Chang Misc. Process Details

The steps involved in the review or change of miscellaneous process details are as follows:



4.23.2 Miscellaneous Process Details Maintenance

When the option to define review or change miscellaneous process details is chosen from the process definition menu a screen similar to the following example will result:

DC@P301801	Basic Process Options
Process : ORDERS	INQUIRE AND/OR MAINTAIN COMPANY OF
Process description	INQUIRE AND/OR MAINTAIN COMPANY OF
Anticipated usage	<u>HEAVY</u>
Process/menu style	<u>CURSOR</u>
Optimise for remote con	nm's N Y=Yes, N=No
Enabled for the Web	N Y=Yes, N=No
Enabled XML generatio	nN Y=Yes, N=No
Function keys: Exit M <u>1</u> 27	Menu Messages Add Change Delete <u>9 10 11</u>
Fnn=Help Fnn=Exit Fr	nn=Cancel Fnn=Messages

From the Basic Process Options screen it is possible to:

- Review the miscellaneous process details.
- Change the miscellaneous process details. Use the CHANGE key to place the screen in change mode and make the desired changes.

Input Options

The following input options apply to changing miscellaneous process details.

Process Description

Specifies the description to be associated with the process. A brief description must be specified for every process created The description aids other users of this process in identifying what it can be used for. See selecting required processes earlier in this chapter.

Anticipated Usage

Specifies what amount of usage of the process in anticipated.

- LIGHT Anticipated usage is light. The process will not be used repeatedly and continuously. Most processes are considered to be LIGHT usage.
- HEAVY Anticipated usage is heavy. The process will be used repeatedly and continuously. This option is normally only used in repetitive data entry applications.

In technical terms this value indicates whether or not the RPG programs created for the functions in this process should set on the LR (last record) indicator and close all files when terminating.

This value can be changed dynamically (without having to recompile any programs) so it may be worthwhile experimenting with it to modify system performance/throughput.

Process/Menu Style

Specifies the "style" of the process that is being reviewed. Allowable values are:

- SAA/CUA All menus and screen formats used by this process and any of its associated functions are to conform to the SAA (Systems Application Architecture) and/or CUA (Common User Access) standards defined for the partition in which the process is being defined. Refer to SAA/CUA Partitions in IBM i SAA/CUA Implementation in the LANSA Application Design Guide for more details of what the SAA/CUA standards are for a partition and how they apply.
- ACT/BAR The process is to act as an "Action bar" as defined by the CUA (Common User Access) standards defined by IBM and for this partition. To use this option the following prerequisites must be satisfied:

- The current partition must be SAA/CUA enabled.

- You must read all relevant information in the *Technical Reference Guide* and the LANSA Application Design Guide and the IBM supplied CUA 1989 Basic Interface Design Guide.

- You must be totally committed to the CUA 1989 standard for the "look" and "feel" of your application software.

Note: The following list of menu styles are now considered to be obsolete and their use is not recommended for new systems. They are supplied purely to allow compatibility with existing systems.

- CURSOR Entries are to be selected from the process menu by positioning the screen cursor on the same line as the entry.
- NUMBER Each entry on the menu is to be assigned a number. An entry is selected by entering the number associated with the entry into a field at the bottom of the screen. This is the "traditional" style of menu and is the most commonly used in other IBM i application systems.
- FUNCTION Entries on the menu are to be displayed with their associated function name. An entry is selected by entering the name of the associated function into a field at the bottom of the screen. This style of menu processing is called "next function" processing and allows the users to go from function to function without having to return to the process menu. Refer to 4.18 Function Control Table for more details.

Optimize for Remote Communications?

Specifies whether remote communications are optimized for all functions within this process.

Allowable values are:

- N Remote communications will not be optimized for all functions within this process. This is the default value used for all new processes created.
- Y Remote communications will be optimized for all functions within this process.

When Y is used, and a process menu and its associated functions are (re)compiled, you should be aware of the following things that may affect your application:

- Compiled process menu display files are created with RSTDSP(*NO) to stop the IBM i restore display "flash" from occurring when a process menu is (re)presented after executing a function. Additionally the compiled process itself uses slightly modified logic to cater for this change.
- Compiled RDML function display files are created with RSTDSP(*NO) to stop the IBM i restore display "flash" from occurring when returning from a call to another RDML function or 3GL program.
- The generated DDS statements make use of the special keywords PUTOVR (put overrides), OVRATR (override attributes) and OVRDTA (override data) to significantly reduce the amount of information (re)sent to the display device on (re)displays of the same screen panel. Generally only fields and their display attributes are (re)sent on a (re)display of the same screen panel. Textual information such as panel titles, field identification details, etc, are not re-sent.
- The screen panel handling code generated for an RDML function is changed to make use of the PUTOVR, OVRATR and OVRDTA key- words. This logic involves a special field called OA@LSQ that is used to track the last screen panel that was presented. Whenever a screen panel is to be presented this value is examined. If it matches the sequence number of the current screen panel command, the PUTOVR keywords are used to reduce the amount of information being (re)sent to the device.
- The fact that textual information is not re-sent can have a detrimental effect on screen panels that use the DEF_COND command to alter the fields and identification details that are visible between subsequent (re)displays of the same screen panel. This effect is immediately apparent, and can be easily corrected by altering the value in field OA@LSQ. See the following points for more details.
- The use of this option can also adversely effect the ":" (full colon) or "." (period) that are used on the end of automatically generated field descriptions (identification method *DESC) in some situations where the screen mode is automatically changed. This is a minor cosmetic problem that may occur in some applications, but it will not effect the application in any other way.
- The use of this option can also affect the automatic display of column headings on a browse list. When a display of an empty list is (re)displayed with a list containing data, the column headings may not appear. The reverse of this is also true, when a display of a list containing data is (re)displayed

with an empty list the column headings may still be displayed. Again, this minor cosmetic effect can be corrected by making reference to the OA@LSQ field.

• The use of this option with POP_UP commands that then invoke other external routines such as help, messages, prompt key processing or calls to other functions or 3GL programs may be cosmetically effected by this option.

Although the (re)display of the original POP_UP window is fully completed, the "background" may disappear because the RSTDSP(*YES) option is required to put back any part of the screen panel that was not created by the POP_UP command.

• Field OA@LSQ is accessible at the RDML program level. Simply define field OA@LSQ in your data dictionary as a packed decimal field of length 7, with 0 decimals, and you alter its value at the RDML level. The most common change to this field is to set its value to zero to "trigger" a complete resend of an entire screen panel that is being (re)displayed on the display device.

Some actions automatically reset this field to cause a complete resend of all screen data. These include:

- A CALL to another RDML function or 3GL program.
- Executing a BROWSE command.
- Executing a MESSAGE TYPE(*WINDOW) command
- Executing a different DISPLAY, REQUEST or POP_UP command within the same function.
- Using the help function key (usually F1). Note also that using this option prevents the use of the actual engraved help key, because screen data could be lost.
- Using the Messages function key.
- Using the Prompt function key.

Access to this "trigger" field allows you to specifically handle situations that require a complete resend of all information to the display device.

For instance, because the generated code is unaware that the IBM i command DSPWTR has overwritten its current screen panel, the following RDML logic will not work correctly when the remote communications option is used:

```
BEGIN_LOOP
REQUEST FIELDS(#PRINTER) IDENTIFY(*DESC)
EXEC_OS400 CMD('DSPWTR #PRINTER')
END_LOOP
```

When the REQUEST command is (re)executed it will only send the field #PRINTER, and not send any identifying text such as the panel title or the field description.

To correct this problem, and to make the (re)execution of the REQUEST command send the complete screen panel to the display device, simply add the reference to #OA@LSQ like this:

BEGIN_LOOP REQUEST FIELDS(#PRINTER) IDENTIFY(*DESC) EXEC_OS400 CMD('DSPWTR #PRINTER') CHANGE FIELD(#OA@LSQ) TO(*NULL) END_LOOP

- Note that this change affects compiled process menus only. It does not affect the way interpretive mode process menus are (re)presented on a display device.
- This option only aids when the same screen panel is being redisplayed. It cannot aid in any way when the screen panel is not already present on the display device.
- The preceding point indicates that when an application is being designed for frequent and heavy usage on remotely attached display devices, there is no better performance aid than the minimisation of the amount of information shown on, and therefore sent to and/or received from, the remote display device.

Enabled for the Web?

Specifies whether LANSA for the WEB is enabled for all functions within this process. Allowable values are:

N The Web will not be enabled for all functions within this process.

Y LANSA for the Web will be enabled for all functions within this process. When functions within this process are compiled, graphical HTML pages will be generated for the screens in the functions. These functions may then be deployed on the IBM i (5250 emulation) or on the Internet. Refer to LANSA for the Web Guide for more information.

Enable XML Generation?

Specifies whether XML Generation is enabled for all functions within this process. Allowable values are:

- N XML generation will not be enabled for all functions within this process.
- Y XML generation will be enabled for all functions within this process. When functions within this process are compiled, XML will be generated for the screens in the functions. These functions may then be deployed on the IBM i (5250 emulation) or over HTTP to XML enabled devices. Refer to *LANSA for the Web - XML Extensions* for more information

Function Keys

Specifies the assignment of function keys to functions within this process.

This information only appears on the screen when reviewing a process in a non-SAA/CUA partition, or when specifically reviewing a non-SAA/CUA process. In an SAA/CUA style process all function key assignments exactly follow those defined for the associated partition.

If desired, change the function key number assigned to the function. Function key numbers specified for any of the functions must be in the range 1 - 24. The same function key cannot be assigned to more than one function.

Function key assignments can be changed dynamically (without having to recompile any programs).

4.24 Process Multilingual Attributes

In a multilingual partition, process multilingual attributes may be specified for each language that is defined within the partition. Process multilingual attributes include the process description and all associated function descriptions.

4.24.1 The Steps to Review or Change Process Multilingual Attributes

4.24.2 Process Multilingual Attributes Maintenance

4.24.1 The Steps to Review or Change Process Multilingual Attributes

The steps involved in reviewing or changing the process multilingual attributes are as follows:



4.24.2 Process Multilingual Attributes Maintenance

When the option to review or change a process's multilingual attributes is chosen from the process definition menu is chosen, a screen similar to the following example will result:

DC@P300601 Multilingual Process - LRTB

Process : PSLSYS

Type	Name P	rocess or Function Description	Language
Process	PSLSYS	Personeels Systeem Hoofdmenu	(DUT)
Process	PSLSYS	Personnel System Main Menu	(ENG)
Process	PSLSYS	Gestione del Personale - Menu P	rincipale (ITL)
Function	ENROL	Enrol a New Employee	(DUT)
Function	ENROL	Enrol a New Employee	(ENG)
Function	ENROL	Registra Nuovi Impiegato	(ITL)

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

From the *Multilingual Process* - *LRTB* screen it is possible to:

- Review the process multilingual attributes.
- Change the process multilingual attributes. Use the CHANGE key to place the screen in change mode and make the desired changes.

This panel will be repeated for each language group i.e. all LRTB (Latin languages), DBCS (Ideographic or Double Byte Character Set languages) and RLTB (Bi-directional languages).

Input Options

The following input options apply to changing a process's multilingual attributes:

Process Description

Specifies the description to be associated with the process. A brief description

must be specified for every language. The description aids other users of this process in identifying what it can be used for.

Function Description

Specifies the description to be associated with each function. A brief description must be specified for every language. These descriptions will appear on the process's main menu.

4.25 The Action Bar Control Table

The *CUA 1989 Basic Interface Design Guide* defines an "action bar" like this: "The action bar is the panel element at the top of the panel that consists of a list of choices that represent a group of related actions that the users can request. A group of actions appear in a pull down when users request an action bar choice. Pull downs are located immediately below the action bar. The actions typically affect information displayed in the work area or in some way control the user's dialogue with the application."

The action bar implementation within LANSA is at the process level.

A process, which is typically used as a menu in an SAA/CUA partition, can be made to act as action bar instead.

This facility is controlled by the "Process Menu/Style" field that is specified when "Creating a new process".

Additionally it can be changed via the "Miscellaneous Process Details" facility. A single line action bar process might appear like this:

File Edit Options Help

If a user selected the first action bar choice (file), then a pull down might appear like this:

 File
 Edit
 Options
 Help

 | _1. Open
 |

 | 2. Close
 |

 | 3. DTX to H/O
 |

 | 4. FAX to H/O
 |

An action bar can use from 1 to 3 lines across the top of the panel. Each line can contain at most 6 action bar choices, thus totalling 18 choices within one process.

Each action bar choice has an associated pull down. Each pull down can support at most 9 choices. Thus $18 \ge 9 = 162$ choices can be controlled from one action

bar.

When a process is used as a normal menu, no extra details are required.

However, when a process is used as an "action bar", additional details about the layout and options in the action bar and it's associated pull downs must be specified in a table called the "Action Bar Control Table".

This is what described in this section: the mechanics of how to define an action bar via the action bar control table.

This section does not describe:

- The mechanics of designing and creating action bar applications.
- The level of commitment to the CUA 1989 standard that is required to design and implement action bar applications.
- Tips and techniques for the design and construction of action bar applications.
- Object/Action design techniques that are essential to the effective implementation of an action bar application.
- The applicability and non-applicability of action bars to different types of applications.
- The non-procedural component of programming action bar processing RDML programs.

It is absolutely essential that you are aware of all these things before attempting to create an action bar application.

The relevant sections from the following guides/manuals must be read in full before attempting to design or implement an action bar application:

- The Technical Reference Guide
- The LANSA Application Design Guide
- The IBM manual CUA 1989 Basic Interface Design Guide, (SC26-4583)

4.25.1 Steps to Review or Change the Action Bar Table

The steps involved in the review or change of the action bar control table are as follows:



4.25.2 Action Bar Control Table Maintenance

If you are not working in a multilingual environment, ignore this language selection panel and proceed to details of the next screen panel.

If you are working in a multilingual environment, then the initial screen that will appear after choosing to review or change the action bar control table will look like this:

DC@P302101Action Bar Definition TableChoose from the list below the language to be worked withSelect Language DescriptionXXAction Bar Definition TableSelect Language DescriptionXX</tr

Use the "/" symbol to select the language that the action bar control table is to be displayed/changed in, then press enter.

Textual changes made to the action bar control table will only effect the selected language.

Structural changes made to the action bar control table (like removing options, changing accelerator keys, etc) will effect all languages.

After the desired language has been selected (if required), a screen similar to the following example will result:

DC@P302101 Action Bar Definition Table

A/Bar Keyword: File	Sequence: 1 AB\$OPT Code : FIL		
P/D : No Description	AK PD\$OPT IA Associated with		
1. Open 2. Close 3 New 4. Select Employee 5 6 7 8 9	OPN Y 1 Function/Employee CLS N 1 Function/Employee CRT N 1 Function/Employee 16 SEL N 1 Function/Employee 17 IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII		
Allowable associations a	are		
2 - Function/Action B	ar Tester		
3 - Function/Department Change			
5 - Special Entry/Oper 6 - Special Entry/Pers	rator Messages onal Messages		
Fnn=Help Fnn=Exit Fn	n=Cancel Fnn=Messages Fnn=Change		

From the *Action Bar Definition Table* screen it is possible to:

- Review the action bar control table.
- Change the action bar control table. Use the CHANGE key to place the screen in change mode and make the desired changes.

Input Options

The following input options apply to changing the action bar control table.

Upper Half of the Screen Panel

The upper half of the screen panel defines each of the 18 possible action bar choices and their associated pull down options.

Use the scroll keys, with the cursor positioned in the upper half of the screen panel, to scroll backwards and forwards through the 18 possible action bar choices.

Lower Half of the Screen Panel

The lower half of the screen panel lists all the things that are associated with the current process, and that may be eligible for invocation by the selection of a pull down choice.

Things that may appear in this list include:

- RDML functions that belong to the current process.
- Other processes/functions that have been "attached" to the current process by the process attachment facility.
- Special entries that have been associated with the current process such as CL commands, 3GL program calls, etc.

Use the scroll keys, with the cursor positioned in the lower half of the screen panel, to scroll backwards and forwards through this list of allowable associations.

A/Bar Keyword

Specifies the text that is to appear in the action bar to identify this action bar choice:

- Try to use just one word.
- Use upper and lower case characters.
- Support for bidirectional and DBCS languages is provided.
- Conform to the CUA 1989 guidelines.
- Help option is automatic. You do not have to define it.

Sequence

Specifies the sequence that this keyword is to appear in the action bar:

- Generally only used to reorder action bar choices.
- Must be in the range 1 to 18 and unique within the action bar.

AB\$OPT Code

AB\$OPT is a field defined in the LANSA data dictionary:

- If it isn't, define it as alphanumeric (length 3).
- The value you specify here is placed into field AB\$OPT when this action bar choice is used.

- Do not use values CUR or ALL, as they are reserved to mean "current" and "all" to Built-In Function SET_ACTION_BAR
- The field is accessible to RDML functions.
- Allows them to decide exactly which action bar choice was used to cause them to be invoked.
- One RDML function handling multiple action bar choices can have good performance implications. Refer to the Sample Program: All 3 Functions in One Program in the *LANSA Application Design Guide*.
- Value specified should be unique within this action bar.
- Standards for AB\$OPT values should be established.

P/D

The list of 9 options (numbered 1 through 9) define the 9 possible pull down choices that can be associated with the action bar choice that is on display:

- To add a new choice, simply enter its details into the next available free space.
- To remove a choice, simply blank out/field exit over all its details.
- To change a choice, just type over the details that are to be changed.
- Avoid using the 9th choice in the first pull down. By the CUA 1989 standards this is reserved for the "Exit" choice. The inclusion of the exit choice into the first pull down is automatic. You do not have to specify it.

No

Specifies the number associated with the pull down choice. It cannot be typed over. To reorder choices in a pull down you must type over the details. Deleted choices cause following choices to move up so that all choices are always numbered consecutively.

Description

Specifies the text that is to appear in the pull down to identify this pull down choice:

- Use upper and lower case characters.
- Support for bidirectional and DBCS languages is provided.
- Conform to the CUA 1989 guidelines.
- Include "Fnn" to identify accelerator keys (where required).
- Include "..." ellipses for resulting pop-ups (where required).

• Help pull downs are automatic. You do not have to define them.

AK(Accelerator Key)

Specifies the accelerator key that is to be associated with this pull down choice:

- Use values in range 01 to 24 to identify.
- Accelerator keys are optional.
- Avoid conflicts with other key assignments. This is not checked.
- Avoid overuse. They will confuse users and complicate the system.
- Conform to the CUA 1989 guidelines.
- Activated on any panel showing this action bar.
- Can be changed dynamically without requiring recompilation.

Pd\$Opt

PD\$OPT is a field defined in the LANSA data dictionary:

- If it isn't, define it as alphanumeric (length 3).
- The value you specify here is placed into field PD\$OPT when this pull down choice is used.
- Do not use values CUR or ALL, as they are reserved to mean "current" and "all" to Built-In Function SET_ACTION_BAR
- The field is accessible to RDML functions.
- Allows them to decide exactly which pull down choice was used to cause them to be invoked.
- One RDML function handling multiple pull down choices can have good performance implications. Refer to the Sample Program: All 3 Functions in One Program in the *LANSA Application Design Guide*.
- Value specified should be unique within this pull down, and preferably, within the entire action bar.
- Standards for PD\$OPT values should be established.

IA (Initial Availability)

Specifies whether or not this pull down choice is to be made available on the initial invocation of the action bar:

- Leave as blank or enter Y to specify initial availability.
- Specify as N to cause initial non-availability.
- Unavailable pull down choices are shown in blue and have their associated selection numbers replaced by an "*".

• RDML program access to make pull down choices available/ unavailable is provide by the SET_ACTION_BAR Built-In Function.

Associated With

Specifies which function/process/special entry from the list on the lower half of the screen is to be invoked when this pull down choice is selected:

- Enter number corresponding to entry in list at bottom.
- Change number as desired.
- Repeated use of the same number is acceptable.
- What happens when the selected option is used depends upon the function/process/special entry.
- Being invoked from an action bar has panel design ramifications for RDML functions. Refer to the Pull Down Definition (for Each Action Bar Option Defined) in the LANSA Application Design Guide.

Allowable Associations Are

Displays the list of functions/processes/special entries that may be invoked by selection of a pull down choice:

- The number associates top of screen details with the bottom.
- The number is not fixed and may change from time to time.
- Text is in format xxxxxxx / ddddddddddddd.
- xxxxxxx may be "Function" indicating an RDML function that belongs to the current process.
- xxxxxxx may be "Attached Process" indicating a process that has been attached to the current process.
- xxxxxxx may be "Attached Function" indicating a function that has been attached to the current process from another process.
- xxxxxxx may be "Special Entry" indicating a special entry that has been defined within the current process.
- ddddddddddd is the description associated with the object.
- Attached processes may cause a menu to appear if they are menu style (i.e. SAA/CUA) and thus do not have an action bar.
- Attached functions may cause the action bar to disappear if they are menu style (i.e. SAA/CUA) and thus do not have an action bar.
- Attached processes may cause another action bar to appear if they are action bar style. This facility can be used to build up a "hierarchy" of action bars in

an acceptable manner.

• Attached functions may cause another action bar to appear. (i.e. the action bar associated with the process to which they belong, not the current process's action bar). This can be confusing to end users in some situations and should be carefully controlled or avoided.

5. Housekeeping

You will find these topics described in this chapter:

5.1 The Housekeeping Menu

- 5.2 Review and Change Access to LANSA Objects
- 5.3 User Access to the LANSA System
- 5.4 Task Tracking
- 5.5 How to Invoke the Work With Tasks Facility

5.6 Reorganizing the LANSA Internal Database

- 5.7 Changing Partition Being Worked With
- 5.8 System Partition Definitions
- 5.9 Frameworks and Groups
- 5.10 Partition Multilingual Attributes
- 5.11 The File Maintenance Utility
- 5.12 Exporting and Importing
- 5.13 Application Templates
- 5.14 Impact Analysis
- 5.15 The Multilingual Development Textual Data Facility

5.1 The Housekeeping Menu

Access to the housekeeping components of LANSA is from the Housekeeping menu. The Housekeeping menu is accessed from the main system menu.



pppHKPMENU

Housekeeping Menu

Enter number of function or place cursor on same line and press Enter.

- _ 1. Review access to objects defined within LANSA
- 2. Review user access to LANSA system
- 3. Submit job to reorg LANSA internal database
- 4. Change partition being worked with
- 5. Create or change system partition definitions
- 6. Work with partition multilingual attributes
- 7. Use the file maintenance utility
- 8. Work with list of objects to be exported
- 9. Import objects into this partition

- 10. Work with Application Template definitions
- 11. Work with Impact Lists
- 12. Create, review or run an online presentation
- 13. Work with multilingual development textual data
- 14. Work with tasks

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Note: 13. Work with multilingual development textual data is for LANSA Internal use only.

5.2 Review and Change Access to LANSA Objects

- 5.2.1 LANSA Objects and Security
- 5.2.2 Initial User Access to Objects
- 5.2.3 The Special *PUBLIC User
- 5.2.4 Initial Public Access to Objects
- 5.2.5 Steps to Review or Change Object Security
- 5.2.6 Select the Objects to Review or Change
- 5.2.7 Review or Change the Object Security
- 5.2.8 External Security Matching

5.2.1 LANSA Objects and Security

The LANSA security system normally deals with:

- Fields
- Files
- Processes (and optionally their associated functions)
- Weblets

Security at the function level is optional. Using function level security involves more run time security checking, and therefore uses more system resources.

A flag in the system definition data area DC@A01 must be set to indicate that function level security is required. Refer to Execution and Security Settings for details.

There are four additional special objects that LANSA deals with. These are:

- System Partitions
- Application Templates
- System Variables
- Multilingual Variables.

Access to all of these objects is controlled by the LANSA security system.

Refer to 5.8 System Partition Definitions and 5.13 Application Templates for further information. Application Template Program Examples are found in the *Visual* LANSA Developer Guide.

Within the LANSA security system there are 2 'classes' of access associated with any object type. These are:

- Access to the DEFINITION of the object. This class of access is applicable to fields, files, processes, functions, partitions, templates, system variables and multilingual variables. This access controls a users right the USE, MODIFY and DELETE the definition of an object.
- Access to the DATA contained in an object. This class of access is only applicable to files. This access controls a users right to READ, ADD, CHANGE or DELETE information (records) contained in the file.

The 2 object classes (DEFINITION and DATA) and the way they affect the 7 object types can be summarized in the following table.

Obj Type

Access Description Of Access Allowable Class

FIELD	DEF	USE: User can use the field definition. MODIFY: User can modify the field definition. DELETE: User can delete the field definition.
	DATA	Data rights are not applicable.
FILE	DEF	USE: User can use the file definition. MODIFY: User can modify the file definition. DELETE: User can delete the file definition.
	DATA	READ: User can read records from the file. ADD: User can add records to the file. CHANGE: User can change records in the file DELETE: User can delete records from the file.
PROCESS or FUNCTION	DEF	USE: User can use (run) the process/function. MODIFY: User can change the definition. DELETE: User can delete the definition.
	DATA	Data rights are not applicable.
PARTITION	DEF	USE: User can access the partition. MODIFY: User can change the partition definition. DELETE: User can delete the partition definition.
	DATA	Data rights are not applicable.
TEMPLATE	DEF	USE: User can use the template.

		MODIFY: User can change the template definition. DELETE: User can delete the template definition.
	DATA	Data rights are not applicable.
SYSTEM VARIABLE	DEF	USE: User can use the system variable. MODIFY: User can change the system variable. DELETE: User can delete the system variable.
	DATA	Data rights are not applicable.
MULTILINGUAL VARIABLE	DEF	USE: User can use the multilingual variable. MODIFY: User can change the multilingual variable. DELETE: User can delete the multilingual variable.
	DATA	Data rights are not applicable.
WEBLET	DEF	USE: User can use the field definition. MODIFY: User can modify the field definition. DELETE: User can delete the field definition.
	DATA	Data rights are not applicable.

5.2.2 Initial User Access to Objects

When a LANSA object is created a default set of users are given authority to the object.

- On **IBM i** it's the System Owner, Partition Security Officer, the creating user's group **or** the creating user themselves, and *PUBLIC.
 - If the creating user's user profile has the Owner setting of *GRPPRF, the creating user's group is used.
 - If the creating user's user profile has the Owner setting of *USRPRF, the creating user is used.
- On **Visual LANSA** it's the System Owner, Partition Security Officer, creating user's group **and** creating user themselves, and *PUBLIC.

5.2.3 The Special *PUBLIC User

When allowing or disallowing access to an object it is very difficult to nominate specific access rights for each and every IBM i user profile. To avoid having to do this a 'special' user profile of *PUBLIC can be used.

The *PUBLIC user profile means 'any other IBM i user' not specifically mentioned in the list of authorized users. Thus if a file had the following security information associated with it:

User	Def: Use	Def: Mod	Def: Dlt	Data: R	Data: A	Data: C	Data: D
QPGMR	Х	Х	Х	Х	Х	Х	Х
QSECOFR	Х	Х	Х	Х	Х	Х	Х
QSYSOPR	Х			Х	Х	Х	Х
QUSER							
*PUBLIC	Х			Х			

It can be seen that:

- Users QPGMR and QSECOFR have full rights to the file.
- User QSYSOPR can read, change, update and delete records in the file but cannot modify or delete the file definition.
- User QUSER has no rights at all to the file
- Any other user (*PUBLIC) can read information from the file.

5.2.4 Initial Public Access to Objects

When a field, file, process, function, partition, template, system variable or multilingual variable is created a field called 'Initial public access' is specified. It has allowable values ALL, NORMAL or NONE. The following table summarizes exactly what access each of the 'initial public access' values grants to each different object type for both access classes (DEFINITION and DATA).

Object Type	Access Class	Action	ALL	NORMAL NONE
FIELD	DEF	USE MODIFY DELETE	Y Y Y	Y
FILE	DEF	USE MODIFY DELETE	Y Y Y	Y
	DATA	READ ADD CHANGE DELETE	Y Y Y Y	Y Y Y Y
PROCESS or FUNCTION	DEF	USE MODIFY DELETE	Y Y Y	Y
PARTITION	DEF	USE MODIFY DELETE	Y Y Y	Y
TEMPLATE	DEF	USE MODIFY DELETE	Y Y Y	Y
SYSTEM VARIABLE	DEF	USE MODIFY	Y Y	Y

		DELETE	Y	
MULTILINGUAL	DEF	USE	Y	Y
VARIABLE		MODIFY	Y	
		DELETE	Y	
WEBLET	DEF	USE	Y	Y
		MODIFY	Y	
		DELETE	Y	

5.2.5 Steps to Review or Change Object Security

The steps involved in reviewing or changing object security are:



5.2.6 Select the Objects to Review or Change

When the option to review access to objects defined within LANSA is chosen from the housekeeping menu a screen like the example following will be presented:

DC@P400102	Select Object(s) to Review
Enter full or pa or leave blank t	tial name of the object(s) to be worked with o select from a list of all objects : Filter type :
Sel Object L	brary Type Description
_ ORDNUM	FIELD Purchase order number
_ ORDLIN	FIELD Purchase order line number
_ ORDMAST	QGPL FILE Purchase order header file
_ ORDLINE	QGPL FILE Purchase order lines file
Fnn=Help Fnn=E	xit Fnn=Cancel Fnn=Messages F15=Bulk Chg
F16=SelectAll F2	2=ShowLong

Using *Select Object(s) to Review* screen, there are three different way to nominate the field(s), file(s), process(s), partition(s), template(s), system variable(s), multilingual variable(s) or weblets required:

- **In full.** Enter the full name of the object required.
- **Partially.** Enter as much as desired of the partial name of the object. A search is made for all objects which generically match the name specified and the resulting list is displayed for subsequent selection.
- Leave blank. In this case a list of all objects is displayed from which the desired object(s) can be selected. Using this method is not recommended as it can take a considerable length of time to build a list of all fields, files, processes, partitions, templates, system variables, multilingual variables and weblets in the LANSA system.

- **Select all** objects in the list by pressing F16
- **Show Long Name** for an object in the list by placing the cursor on it and pressing **F22**. Refer to Long Names for more details.

Filter type

A filter value may be entered to restrict the list of objects to a particual object type. Filter values may be FIELD, FILE, PROCESS, SYS PART, TEMPLATE, SYS VAR, ML TEXT, WEBLET or various abbreviations for these types.

When a list of objects is displayed the screen will look like the following example which resulted from entering partial name ORD (and found 2 fields and 2 files which matched):

DC@P400102 Select Object(s) to Review
Enter full or partial name of the object(s) to be worked with or leave blank to select from a list of all objects : Filter type :
SelObjectLibraryTypeDescription_ORDNUMFIELDPurchase order number_ORDLINFIELDPurchase order line number_ORDMASTQGPLFILEPurchase order header file_ORDLINEQGPLFILEPurchase order lines file
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages F15=Bulk Chg

Using the resulting list of objects it is possible to:

- Alter the object name and/or filter at the top right of the screen. The change can be to specify a full object name, a partial object name, a blank name or a different filter type. In all cases another search is made for the object(s) required.
- **Select** one or more objects from the list by entering any non-blank character beside the object in the column labeled 'Sel' and pressing *Enter*. As described in 5.2.7 Review or Change the Object Security, the full object
security display is then presented for each object chosen.

• **Select** multiple objects from the list and press F15 to change the security details for all the selected objects.

5.2.7 Review or Change the Object Security

When an object (field, file, process, partition, template, system variable, multilingual variable or weblet) has been chosen from the display detailed in the preceding section a screen similar to the following example will be presented:

DC@P400101 Sele	ct Object(s) to Review
Enter full or partial name	e of the object(s) to be worked with
or leave blank to display	a list of all objects : le
	Filter by Type : FIELD
Sel Object Library	Туре
X LEABACCEL	FIELD Action Bar Accelerator key
X LEABPDDSC	FIELD Pull Down Description
X LEACACCN	FIELD Action Bar Accelerator key
X LEACBARDS	FIELD Action Bar Item Description
X LEACBAROP	FIELD Action Bar Options Code
X LEACCDESC	FIELD Description
LEACCDFTA	FIELD Default action
LEACCFILE	FIELD File accessed
LEACCKEEP	FIELD Keep last
LEACCKEY	FIELD Key Field/Value
LEACCMAXR	FIELD Number of records
LEACCNAME	FIELD Name
LEACCPJFD	FIELD PJF derivation
LEACCREL	FIELD Relationship Type +
Fn=Help Fn=Exit Fnn=	Cancel Fnn=Messages Fnn=BulkChg Fnn=SelectAl
F22=ShowLong	

If the *Bulk Change* option is used, a list of all the selected objects that you have authority to change will be shown along with the user authorities.

From this screen it is possible to:

ĪF

• **Show Long Name** for the object by pressing **F22**. Refer to Long Names for more details.

Note: The change function key is only enabled if the current user is the LANSA security officer or has change rights to the definition of the object being displayed.

DC@P400102 Review / Change Object Access Object : ORDLINE Library : QGPL Type : FILE : Purchase order lines file Description DEFINITION DATA DEFINITION DATA User Use-Mod-Dlt R-A-C-D User Use-Mod-Dlt R-A-C-D *PUBLIC Х XXXX QSECOFR X X X X X X X X QUSER QPGMR X X X X XXXX Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages F22=ShowLong

Enter the users and authorities you wish to apply to all the listed objects. Press *Enter* to apply the change to all the objects.

To show the long name for an object, select it (by placing your cursor on it) and press **F22**.

5.2.8 External Security Matching

A flag field in the system definition data area DC@A01 must be set to indicate that External Security Matching is required. Refer to External security matching for LANSA files in *Work with Execution and Security Settings* for *details*.

External Security Matching is only applicable to LANSA defined files. Any other objects or 'OTHER' files will not be affected.

When External Security Matching is enabled, the internally defined object rights for a LANSA defined file will be reflected by the operating system.

Security matching is performed when:

- LANSA defined file security details are changed from the Housekeeping Menu option 'Review access to objects defined within LANSA'.
- During an Import of a LANSA defined file.
- When a LANSA defined file is made operational.

Working from this display it is possible to:

- Review the list of authorized users and their associated access rights.
- Change the list of authorized users and their associated access rights. To do this use the CHANGE function key to place the screen in change mode and make any desired changes. The changes made can include changing a user already in the list, removing a user from the list or adding a new user to the list.

Note 1: Changes to a user's access rights to LANSA objects may not take effect until the next time the user starts to use LANSA. If the user is currently using LANSA they should exit from LANSA and then re-invoke LANSA to ensure that the changed object access rights take effect.

Note 2: Function level security is optional. A flag field in the system definition data area DC@A01 must be set to indicate that function level security is required. Refer to System Definition Data Area DC@A01 for details of its layout and how to change flags within it.

5.3 User Access to the LANSA System

Access to the LANSA system is automatic. Any IBM i user can type in the command LANSA or call program LANSA and the main system menu will be displayed.

However, unless the user has been specifically allowed access to some component(s) within the LANSA system (or is part of a 'group' profile that has) there will be no entries on the main system menu and the user will not be allowed to do anything within LANSA. You set the components to which a user has access using the *Review user access to LANSA system* on the Housekeeping menu.

5.3.1 Access to the Components within LANSA

5.3.2 Steps to Review, Change or Allow Access to LANSA

You can specify a user's name as described in 5.3.3 Specify the User Profile Name or in Authorize Visual LANSA Developer via the *Administration* menu.

You can remove a user's name via the Remove a LANSA User via the *Administration* menu.

5.3.1 Access to the Components within LANSA

Access to the various components of the LANSA system is controlled by which entries appear on the user's LANSA menus. If the component is on the user's menu it can be used, otherwise it cannot.

To allow a user access to LANSA an option in the housekeeping component of LANSA is used. The user name is specified and then a list of all the LANSA menus and all the entries on the menus is displayed.

To authorize the user to have (and thus be able to use) certain entries on his/her menu all that has to be done is go through the list and 'tick' the ones required.

To 'unauthorize' the user to remove (and thus not be able to use) certain entries on his/her menu all that is required is to go through the list and blank out (or 'untick'?) the ones no longer required.

The steps involved in these procedures are described in more detail in the following sections.

5.3.2 Steps to Review, Change or Allow Access to LANSA

The steps involved in reviewing, changing or allowing access to the various components of LANSA are as follows:



5.3.3 Specify the User Profile Name

When the option to review, change or allow access to the LANSA system has been chosen from the housekeeping menu a screen like the example following will be presented initially:

USRACCESS Review / Change User Access Enter user name whose access rights to LANSA are to be displayed Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Enter the name of the IBM i user who is to be allowed access to LANSA, or whose existing access rights to LANSA are to be reviewed or changed and press the ENTER key.

The resulting display will look like this:

USRACCES	2SS Review / Change User Access						
Enter user name whose access rights to LANSA are to be displayed QPGMR							
	Allow						
Menu name	Description of menu entry Acce	SS					
MAIN SYST	ΓΕΜ ΜΕΝU						
	Use a process _						
	Submit a process to batch _						
	Field control menu						
	File control menu						
	Process control menu _						
	Housekeeping menu						
	Administration Tasks						
	Display submitted jobs - workstation						

Display submitted jobs - user
Display workstation message queue _
Exit from LANSA
PROCESS CONTROL MENU
Create a new process definition
Review or change a process definition
Delete a process definition
Compile a process from new or amended def.
Review, change or create process HELP text _
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The resulting display shows all menus in the LANSA system and all entries on each of the menus. When the screen is first presented the user's current right to use a menu entry is indicated by a 'Y' (for YES) in the right hand column headed 'Allow access'.

Additional entries are shown under the heading *Visual LANSA Authorities*. These are not IBM i menu entries. They are used to control user access within a slave Visual LANSA system.

Working from this display it is possible to:

- Review the user's current access rights. Use the ROLL keys to roll backwards and forwards through the list of menu and menu entries.
- Change the user's current access rights. Use the CHANGE function key to place the screen in change mode. Enter a 'Y' beside the menu entry if the user is allowed to use it. Remove the 'Y' from beside the menu entry if the user is no longer allowed to use it.

Note: Menu 'hierarchy' must be observed when allowing access to the various menu entries. For instance authorizing the user to 'Create a new process definition' (on the process control menu) will not allow access unless the user is also authorized to 'Display process control menu' (on the main system menu).

Note: Changes to a user's access rights to LANSA only take effect after they exit from LANSA, and any connected jobs, and then re-invoke LANSA.

5.4 Task Tracking

Task Tracking will record all development work performed in LANSA at partition level on the following LANSA objects:

- Fields
- Visual LANSA components
- Files
- Functions
- Processes
- Application templates
- System variables
- Multilingual variables.

Whenever the term 'object' is used in relation to task tracking, it is intended to describe one or all of these objects.

Task Tracking is used to keep a history of all work performed on objects for a particular task and then the task identifier can be used to export all objects worked on for a task.

To do this, task tracking enforces locks when changing or deleting objects. An object which is locked to one task cannot be used by another task.

When setting up task tracking you need to consider how much control you want to maintain over the development effort, how much administration overhead you are willing to bear, how big your team is and whether development needs to be controlled by individual developer or by product or by both.

For example:

• **Task-oriented tracking** is used to keep track of units of work. A unit of work can be, for example, a new program feature or a program fix. Developers are assigned a new task identifier for each new unit of work to be performed on any product. When a task is completed, the task identifier is removed from use.

The style of task tracking ensures that your development work is carefully controlled. Its use is recommended for large, professional development teams. It should be noted that it involves a relatively high level of planning and administration.

• **Product task tracking** requires less adminstration and is configured by assigning unique task identifiers to products (groups of objects). In product-

oriented tracking all developers use the same task identifier when working on a particular product.

- Another simplified way of using task tracking is to assign task identifiers to **individual developers**. The developers use their own task identifier regardless of the product they are working on.
- Task tracking should not be disabled, but in small teams with highly trusted developers a minimalist task tracking setup can be used. In this setup there is only one task per system.

You set the user's Task Tracking options when you create (or amend) a partition, as described in 5.8.3 Partition Definitions - Create, Change or Delete.

5.4.1 What Happens When Task Tracking Is Active?

When Task Tracking has been activated for a partition, all development work within LANSA will be monitored. When an object is selected (to create, compile, change, delete or review), Task Tracking will execute the following procedures every time:

- 1. After the user has selected an object to work with and the normal LANSA object security checks have allowed work to commence, it makes sure that:
 - A valid task identifier has been specified if the user requires a task identifier to work in LANSA.
 - The status of the task is either 'OPN' or 'WRK'.
 - The user for the job is in the list of authorized users/groups for the Task Identifier, or is the LANSA partition security officer user/group profile or QSECOFR user/group profile.
 - The user is allowed to change Task Identifiers and is authorized to work with the task that is allocated to the object if the object that has been selected to work with is allocated to another task identifier.
- 2. After the user has completed work on the selected object, task tracking makes sure that:
 - If no task identifier has been specified from the time LANSA was invoked OR the user is required to confirm the task identifier he/she is working with, a POPUP window will be displayed. This is to prompt/confirm the user for the task identifier that should be allocated to the selected object.
 - All the checks carried out in Step 1 will be performed again when the Task Identifier has been allocated in the POPUP window.
 - If any of the checks performed in Step 1 have been found to be false OR the user has decided to exit from the POPUP window, a message will be issued stating that 'Work has not been committed' and LANSA database changes will not be performed.
- 3. If work has completed successfully for the object, LANSA database files are updated with the object changes and the Task Tracking database files are updated with the details of the events that have taken place.

Note: The Task Id locked to an object can be displayed/hidden using a function key.

5.4.2 Choose Your Task Tracking Approach

When you set up task tracking you need to choose the approach that best suits your needs. In many cases full task-oriented tracking is not required. You can choose from these four basic task tracking approaches:

Full task- oriented tracking	Developers are assigned a new task identifier for each new unit of work they perform on any product. Tasks may involve multiple developers and multiple products.				
	Suitable for large sites, for sites with task-oriented methodologies and sites with product and developer cross-over.				
	- Administrative overhead.				
Tracking	Task identifiers are assigned to products.				
by product	Suitable for small teams which have little cross-over between products				
	+ Little administrative overhead				
	- No product completion points per developer				
Tracking	Task identifiers are assigned to individual developers.				
by developer	Small teams where there is little cross-over between developers.+ Little administrative overhead				
	- No product completion points				
Minimum tracking	One task for the entire system. Only system internal task tracking activities are performed. Suitable for small teams with trusted developers				
	+ Minimal administrative overhead.				
	- There is no locking out of objects and you cannot export by task or inquire on task history.				
	- No definable units of work				

Configuring Task Tracking

Full task-oriented tracking is the default and requires no additional configuration. Minimum tracking as well as product and developer oriented tracking are configured by inserting a task prefix for each in the Work with Task Settings which you will find via the *Review System Settings* available from the *Administration Menu*.

User options are set for each partition as described in 5.8.3 Partition Definitions - Create, Change or Delete.

Tracking by product	Insert the prefix for product-oriented tasks in <i>Product level task tracking prefix</i> in the Work with Task Settings. It is recommended that you use '*T' as the first two characters for product-oriented tasks. If you follow this recommendation, you could, for example, create task identifiers *T00000A and *T00000B for a Product A and Product B.
	Users need to be authorized to product-oriented tasks as usual.
Tracking by developer	Insert the prefix for developer-oriented tasks in <i>Developer task tracking prefix</i> in the Work with Task Settings. It is recommended that you use '*U' as the first two characters for developer-oriented tasks.
	It is strongly suggested that an appropriate naming convention be used to identify tasks with users. If you use the recommended prefix, you could, for example, create task identifiers such as *U000BOB or *U00MARY.
	Only one user can be authorized to a developer-oriented task.
Minimum tracking	Insert the prefix for a single task for the system in the <i>Minimum task tracking prefix</i> in the Work with Task Settings. The recommended prefix for the single permanent task is '*N'. A suggested name for this task is '*NONE'. All users can use this task and they do not need to be authorized to it.

In addition to the prefixes, for each task tracking level, you can set a *Check in unlocking* option in the Work with Task settings. When this option is set to 'Y', an object is unlocked from the task both on the IBM i and on the workstation when it is checked in. At the same time, it is changed to read-only on the workstation. To change an object again on the workstation, you must check it out for update.

Your choice of task tracking approach should also be reflected in the task tracking configuration settings of partition definitions. Most notably in the minimum, product-oriented and developer-oriented approaches there is usually no reason to allow users to change tasks while working. In these approaches there is also no need retain task history, so it can be purged.

5.4.3 Purge Task History

The purging of task history is controlled using:

• The *Purge Task History* date in a LANSA REORG job.

and

• The *Retain task history on REORG* option, which is set using the Work with Task settings.

The default value for the retain task history flag is Y.

When the Retain Task History flag is set to N, task history will be purged during the next REORG job.

When the flag is set to N, you will be able to purge up to today's data or change to an earlier date when the reorg job is submitted. If the date is set to zeroes, the history is not purged. Similarly, if the flag is set to Y the history is not purged.

For information on the LANSA Reorg job, refer to 5.6.4 Submit Job to Reorganize the Database.

5.4.4 Prompt/Confirm Task Identifier (POPUP Window)

When a task identifier is required for input or if you are required to confirm the task identifier you are working with, a popup window is used for you to prompt or confirm the task identifier that should be allocated to a selected object:

```
Prompt/Confirm Task Identifier :
Task Identifier ... <u>*TSKAABC</u> :
Task Identifier ... <u>*TSKAABC</u> :
Fnn=Help Fnn=Exit Fnn=Prompt :
Fnn=Cancel Fnn=Messages :
```

Task Identifier

Mandatory. Enter the **Task Identifier** to be allocated to the object you are currently working with. A number of checks will be performed to ensure that the task identifier is valid and can be allocated to the object.

Alternatively, if you press the Prompt key, a popup window displaying a list of the task identifiers so that you can select one.

When you have confirmed the task identifier, the work that has been performed on the object will be completed and the task identifier will be allocated to this job until you exit from LANSA or the task identifier for this job is changed.

5.4.5 Select Task Identifier (POPUP Window)

When the Prompt key has been selected from the 'Prompt/Confirm Task Identifier' window, a popup window is used to display a list of existing task identifiers:

: :		
: Select Task Identifier :		
: Task ID Description :		
: *TSK0327 Design A/C module database - Ref. 0327		:
: *TSK0331 User change request - Ref. 0331	:	
: *TSK1205 Implement new A/C module - Ref. 1205		:
: *TSK1336 User change request - Ref. 1336	:	
: :		
: :		
: :		
: :		
: :		
: Bottom :		
: Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages		:
: :		
·		

Position the cursor under the required task identifier and press Enter.

When a task identifier has been selected, you will return to the 'Prompt/Confirm Task Identifier' popup window.

5.5 How to Invoke the Work With Tasks Facility

The steps involved in invoking the 'Work with Tasks' facility can be visualized like this:



5.5.1 Working with Tasks

When the *Work with Tasks* option is selected from the Housekeeping Menu the first screen presented looks something like this:

DC@P810001	Work with Tasks	
DC@P010001	WORK WILL LASKS	

Position to _____ Filter by Status . . ____ Type options (and Task Identifier) and press Enter.

1=Create 2=Review/Change definition 4=Delete 5=Review Task ID History 6=Print Task ID History 17=Last Action Details

Opt Task ID Task Identifier Description Status

*TSK0327 Design A/C module database - Ref. 0327 WRK
 *TSK0331 User change request - Ref. 0331 OPN
 *TSK1205 New A/C type module - Ref. 1205 OPN
 *TSK4071 User change request - Ref. 4071 CLS

Bottom

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Option

Enter the code representing the action you wish to take beside a Task in the list. The actions available are:

1=Create: to create a new task identifier. Enter this code on the first Task line where the Task ID is blank. You can enter your new Task ID here or on the Create Task Identifier screen that is displayed for you to enter the new task's details.

2=Review/Change definition to review and possibly alter details of the selected task. This option is not valid on the first input line displayed unless an existing task identifier is also specified.

When this option is used, the Review Task Identifier screen is displayed showing the details currently recorded for the task.

4=Delete to delete an existing task identifier. Note that a task identifier can only be deleted if its status has been set to 'CLS' or 'FIN'. This option is not valid on the first input line unless an existing task identifier is also specified.

You will be asked to confirm the deletion. When you have accepted, the Work Tasks screen will be redisplayed with the deleted task identifier omitted.

5=Review task history To review the selected task's history. If you enter this option on the first input line, you will also need to enter a task identifier.

When this option is used, the *Review Task Identifier History* screen is displayed with a list of objects that have been allocated to the selected task.

6=Print task history to print the history of the selected task. If you enter this option on the first input line, you will also need to enter a task identifier. When you use this option, a report is produced listing all the objects and their events worked on under this task identifier.

17=Last Action Details: indicates that you wish to review the last action details recorded for the selected task identifier. A screen showing the last action details will be displayed. This screen is described in Online Access to Last Action Details.

F16=Settings: displays the system settings relevant to Task Tracking. For details, refer to Export and Import settings.

The list of tasks shown may be filtered by entering a value in the *Filter by Status*.

If *Filter by Status* is left blank, all tasks will be shown.

5.5.2 Create a Task Identifier

When option 1=Create is chosen from the Work with Tasks menu, the Create Task Identifier screen is displayed.

DC@P810002	Create Task Identifier				
Type new Task Identifier details and press Enter.					
Task Identifier	. <u>*TSK4075</u>				
Description Authorized users .	•				
Task status	<u>OPN</u>	-			
Fnn=Help Fnn=Ex	it Fnn=Cancel Fnn=Messages				

All the entries on the Create Task Identifier screen are mandatory.

Task Identifier

Mandatory. The unique identifier of this new Task. This is a free format field of up to 8 characters long. The value entered is 'right blank' adjusted.

The first two (left most) characters of the Task Identifier should conform to the codes set in positions 640-645 of data area DC@A07. These codes indicate the type of task tracking that you wish to use with this Task. If you're not familiar with the different types of task tracking you can use, refer to 5.4.2 Choose Your Task Tracking Approach

Note that task identifiers are shared across all partitions, they are not unique in each partition.

You can set the codes using the Product level task tracking option in *Work with Task settings*.

Description

Mandatory. A description of the task identifier. This will appear on all lists and

reports of task identifiers.

Authorized Users

Mandatory. Authorized user/group profiles that can work under the task. At least one and up to ten valid system user/group profiles can be entered. All user/group profiles not defined cannot use this task identifier except for QSECOFR, the QSECOFR group, or the Partition security officer user/group profile.

Task Status

Mandatory. The current status of the task. Task default is initially OPN (Task is Open). Other valid task status values are WRK (Work is being/has been performed), CLS (Task is Closed) and FIN (Task is Finished).

When you have entered all the new task identifier's details and pressed Enter, you will be prompted to enter another new task identifier.

Note: Task Identifiers can be used in any partition within the LANSA system but it is recommended that a task identifier should be allocated for use in one partition only.

5.5.3 Review/Change Task Identifier

When option 2=Review/Change definition is chosen from the 'Work with Tasks' menu, the Review Task Identifier screen is displayed.

DC@P810003	Review Task Identifier			
Task ID : *TSK1205 Ne	w A/C type module - Ref. 1205			
Use Change key to modi	fy task identifier definition.			
Description New A/C type module - Ref. 1205 Authorized users QPGMR				
Task status OPN				
Fnn=Help Fnn=Exit Fn	n=Cancel Fnn=Change Fnn=Messages			

Working from the 'Review/Change Definition' screen you can:

- Review the task identifier's details
- Change the task identifier's details. To do this, use the CHANGE function key to place the screen in change mode and make any desired changes. The Change Task Identifier screen is displayed.

DC@P810004Change Task IdentifierTask ID : *TSK1205 New A/C type module - Ref. 1205Type changes to task identifier details and press EnterDescription <u>User change request - Ref. 1205</u>
Authorized users . . . QPGMR

Task status <u>OPN</u>

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

You can change these details on the Change Task Identifier screen:

Description

A description of the task identifier. This will appear on all Task lists and reports.

Authorized Users

Authorized user/group profiles that can work under the task. Up to ten valid system user/group profiles can be entered. User/group profiles not defined are restricted from using this task identifier except for QSECOFR, the QSECOFR group or the Partition security officer user/group profile.

Task Status

The current status of the task. Valid task status are OPN (Task is Open), WRK (Work is/has been performed), CLS (Task is Closed) and FIN (Task is Finished). Note that you must set a task's status to FIN or CLS before you can delete it.

Note: Task status cannot be changed from FIN (Finished) to any other status.

When you change the task's status to FIN, you will be asked if you want to retain the Task History for this task.

If you enter NO, the task history will be deleted immediately.

If you enter YES, the task history will be retained. This history is subject to the normal purging rules. Refer to 5.4.3 Purge Task History for details.

In both cases, the objects locked to this task will be released and may then be used by other tasks.

5.5.4 Delete a Task Identifier

When option 4=Delete is chosen from the 'Work with Tasks' menu, the Delete Task Identifier screen is displayed for you to confirm the deletion.

DC@P810005 Delete Task Identifier Task identifier . . . : *TSK4071

Description : User change request - Ref. 4071 Status : CLS

Confirm this task identifier to be deleted

Delete this task identifier . . . ____ YES, NO

WARNING : Deletion of this task identifier will remove its history

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Delete This Task Identifier?

A YES or NO entry is required.

YES will delete the task identifier and its history, NO will cancel this request and the task identifier and its history will not be deleted.

Note: A task identifier can only be deleted if its status is set to CLS (Task is closed) or FIN (Task is finished). You can change the status using the 5.5.3 Review/Change Task Identifier screen.

When you have answered the 'Delete this task identifier' question, you are returned to the 'Work with Tasks' menu.

5.5.5 Review Task Identifier History

When you choose option 5=Review Task ID History on the *Work with Tasks* menu, the *Review Task Identifier History* screen is displayed so that you can select objects worked on by a specific task identifier for review or for printing:

DC@	P810006	6 Revi	ew Task Ide	entifier	History	
Task	ID: *TSŀ	K0327 Desigr Position to 1	1 A/C modu Partition/Ob	le datab oject <u>-</u>	oase - Ref.	0327 Status: WRK
Type	options,	press Enter.	The by Iy	Je		
5= R e	view Ob	ject History	6=Print O	bject H	istory 9=	Change Task
Opt	Partitio	n Object	Extension	Туре	Curre	nt PC
_	DEM	TTSDF1	FIE	LD Z	XXXXXX	
_	DEM	TTSDF2	FIE	ELD		
_	DEM	TTSFD1	DC@DEM	ÍOLIB	FILE	
_	DEM	TTSPD1	DC@DEM	ÍOLIB	PROCES	S
_	DEM	TTSPF1	TTSPD1	FUNC	CTION	
_	DEM	TTSPF2	TTSPD1	FUNC	CTION	
_	DEM	*NEXTINV	OICENUM	BER	SYS VA	R
_	DEM	*MTXTORI	DERENTRY	/HDR	MULT	TEXT
				Botton	า	

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages F22=ShowLong

Position to Partition /Object

You can control the partition/object display using the 'Position to Partition /Object' to position the start of the objects displayed. The partition and object can be full or partial names. **This value is case sensitive**.

Some of the objects may have lower case names. For example:

- for the Position to Weblets enter a lower case value.
- for other object types, enter an uppercase value.

Filter by Type

You can control what object types are displayed. The filter may be any of the possible types shown or various abbreviations of the type. If the filter is left blank, all types are shown.

Option

Working from the Review Task ID History screen, enter the code representing the action you wish to take beside an object in the list. The actions you can take are:

5=Review Object History, to review the history of the object selected under this task identifier. When this option is used, a screen will be displayed in the form of a selection list to allow all the events that have taken place for an object (while worked on under a specific task identifier) to be reviewed or printed.

6=Print Object History to print the history of the object worked on under a specific task identifier. When this option is used, a report is produced listing all the events that have taken place for an object while it was worked on under a specific task identifier.

9=Change Task to change the task that the object is currently locked to. When this option is used, a screen is displayed to allow this. This screen is described in 5.5.8 Change Task for an Object.

Show Long Name for an object in the list by placing the cursor on it and pressing **F22**. Refer to Long Names for more details.

Partition

Indicates in which partition an object was worked on under a specific task identifier.

Object

Is the name of the object worked on under a specific task identifier, object name can be a LANSA FIELD, FILE, FUNCTION, PROCESS, APPLICATION TEMPLATE, SYSTEM VARIABLE or MULTILINGUAL VARIABLE.

Extension

Is the extension to the object name, the extension for a FILE is the library in which the file was created, the extension for a FUNCTION is the PROCESS in which the function was created, the extension for a PROCESS is the library in which the process was created. FIELD, APPLICATION TEMPLATE, SYSTEM VARIABLE and MULTILINGUAL VARIABLE have no extension.

Туре

The kind of object that has been worked on under a specific task identifier.

These may be:

- Field
- File
- Function
- Process
- Application template
- System variable
- Multilingual variable
- Web component
- Weblet
- Technological service.

When you have finished reviewing the history of a task identifier, you will be returned to the 'Work with Tasks' menu.

Current PC

Is the name of the PC that currently has the object locked, as it is known to LANSA. This will be blank if Visual LANSA is not used for development, or if the object is not currently checked out to a PC for update.

5.5.6 Review Object History

When 5=Review Object History is chosen from the 'Review Task ID History' menu, a screen similar to the following example is used to review Object history and select events that have taken place for an object (while worked on under a specific task identifier) for review and printing:

DC@P810007 **Review Object History** Task ID: *TSK0327 Design A/C module database - Ref. 0327 Status: WRK Partition: DEM Object: TTSDF1 Type: FIELD Extension: Type options, press Enter. **5=Review Event Details** 6=Print Event Details Time **Opt Event Date** User Message ID PC Name CRT 9/11/90 12:19:39 QPGMR DCM0015 XXXXXX CHG 11/11/90 15:37:53 QPGMR DCM0014 **Bottom** Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages F22=ShowLong

Option

Working from the Review Object History screen, enter the code representing the action you wish to take beside an event in the list. The actions you can take are:

5=Review Event Details to display the details of the events which have taken place for the selected object.

6=Print Event Details to print the details of the events which have taken place for the selected object while working under the selected Task identifier.

Show Long Name for the object by pressing **F22**. Refer to Long Names for more details.

Event

Is the action that has taken place on the object (while worked on under a specific task identifier). The event types are:

CHG - Object has been Changed

CRT - Object has been Created

DLT - Object has been Deleted

IMP - Object has been Imported.

Date & Time

Is the date and time the event took place on the object while worked on under a specific task identifier.

User

Is the name of the user who performed the event on the object while worked on under a specific task identifier.

Message ID

Is the message identifier of the message which was issued when the event took place on the object while worked on under a specific task identifier.

When you have finished reviewing the history of an object, you will be returned to the 'Review Task ID History' menu.

PC Name

Is the PC name associated with the event, as it is known to LANSA. This will be blank if Visual LANSA is not used for development, or if a PC was not part of the event.

5.5.7 Review Event Details

When option 5=Review Event Details is chosen from the 'Review Object History' menu, a screen similar to the following example is used to review the event details of an object:

DC@P810008 Review Event History

Task ID: *TSK0327 Design A/C module database - Ref. 0327 Status: WRK Partition: DEM Object: TTSDF1 Extension: Type: FIELD Event: CRT Date: 9/11/90 Time: 12:19:39 User: QPGMR

Message ID . . . : DCM0015

Message file . . : DC@M01

Library ...: DC@PGMLIB

Message text . . : Definition of field TTSDF1 successfully added to data dictionary.

The definition of field TTSDF1 has been added to the Data dictionary. If this field has been Automatically added to the dictionary you should review its definition and make changes to the associated description, column headings and label as required.

Message text . . : Object XXXXXX type XX check in to IBM i repository completed.

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages F22=ShowLong

Show Long Name for the object by pressing **F22**. Refer to Long Names for more details.

When you have finished reviewing the event details of an object, you will be returned to the 'Review Object History' menu.

5.5.8 Change Task for an Object

When option 9=Change task is chosen from the 'Review Task Identifier History' menu, a screen similar to the following example is displayed for you to change the task the object is locked to:

DC@P811201 Change Task for an Object

> New Copy Rel Default: Task Hist PC

			Curren	t New	Сору	Rel	
Part C)bject	Extensio	n Type	PC	Task	Hist	PC
DEM	*AQJ	PNSYSV	ARI	SYS VAR	XX	XXXX	QATEST
DEM	*AQJ	PNSYSV	ARI	SYS VAR	XX	XXXX	QATEST
DEM	*AQJ	PNSYSV	ARI	SYS VAR	XX	XXXX	QATEST
DEM	*AQJ	PNSYSV	ARI	SYS VAR	XX	XXXX	QATEST
DEM	*AQJ	PNSYSV	ARI	SYS VAR	XX	XXXX	QATEST
DEM	*AQJ	PNSYSV	ARI	SYS VAR	XX	XXXX	QATEST
DEM	*AQJ	PNSYSV	ARI	SYS VAR	XX	XXXX	QATEST

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Working from the 'Change Task for an Object' screen it is possible to:

- Change the task ID that an object is locked to.
- Copy the task history for the object to the new task ID.
- Unlock an object from the task ID to which it is locked.
- Release an object from the PC to which it is locked.

If repository propagation is not on, then you must manually change all PC Repositories to reflect the changes made on the Master Repository. If this is not done it is possible for a PC to overwrite changes on the IBM i Master Repository.

These input options are required when changing the task identifier for an object:

New Task ID

To change the Task ID that the object is locked to, enter the new Task ID for the object, leaving Release from PC Only blank. This Task ID must be already exist.

To unlock the object from the current task ID, leave blank, leaving Release from PC Only blank also.

In both cases the PC lock will be released.

Note that task identifiers are shared across all partitions, they are not unique in each partition.

Default Values

You may enter a New Task and Copy History OR Release from PC as a default value or you may enter the value for individual objects. The Default values will be used when values are not provided for an individual object.

Copy Event History

Indicates whether or not the event history for the object is to be copied to the new task identifier.

Enter YES if the event history for the object is to be copied to the new Task ID. This is not applicable if the object is simply being unlocked from the task, or if the object is being released from the PC only.

Enter NO if the event history for the object is not to be copied to the new Task ID.

Release from PC

Indicates whether or not to release the object from the PC it is locked to only.

Enter YES to release the object from the PC it is locked to while it remains locked to the task. This is not applicable if the Task ID that the object is locked to is being changed.

Note:

Up to 100 objects may be listed. If more than 100 objects were selected, the list

will be presented multiple times.

When you have finished changing the task for all the objects, you will be returned to the *Review Task Identifier History* menu.
5.6 Reorganizing the LANSA Internal Database

LANSA stores a great deal of information about fields, files and processes. This information is kept in a set of approximately 80 IBM i database files that reside in the LANSA 'data' library. This is normally called DC@DTALIB, but may be different at your installation. These files begin with the prefix 'DC@F', 'DC@W'or 'DC@X' which identify them as 'internal' LANSA files.

Also see

5.6.1 Why Does it Have to Be Reorganized?

- 5.6.2 When Should It Be Reorganized?
- 5.6.3 Steps to Reorganize the Database
- 5.6.4 Submit Job to Reorganize the Database
- 5.6.5 Remotely Invoke the Reorganization Facility

5.6.1 Why Does it Have to Be Reorganized?

Like all IBM i database files, those in the LANSA 'internal' database should be regularly reorganized. There are two good reasons for doing this:

- To free up disk space by removing deleted record space from the files. When records are deleted from IBM i database files the disk space they occupy is not freed or re-used until the file is Reorganized.
- To improve the performance of both LANSA and client applications by re-arranging the data in the file so that it is in the same order as the most commonly used access path. Performing this function speeds up applications that access the file.

5.6.2 When Should It Be Reorganized?

The need to reorganize the LANSA database depends on the frequency and volume of changes made to the LANSA field and file definitions.

It is recommended that the LANSA database be reorganized weekly during a development phase. In general, a stable production system only needs to be reorganized after importing a large number of LANSA objects or if major changes are made to the Help text.

Scheduling of the Reorganization Job

The Reorganize job clears all data more than 3 months (90 days) old from a LANSA internal file named DC@F16. This file is used to exchange information between LANSA online tasks and batch jobs that they have submitted.

For example, it exchanges the list of fields to print between the online 'Print Field Definitions' facility and the actual batch job that produces the report. Likewise, it exchanges details between online RDML programs and batch RDML programs invoked by using the SUBMIT command (most notably the exchange list contents).

If there is a delay of more than 90 days between a LANSA batch job being submitted and being executed, and if a Reorganize job is run to remove the 90 day old information from DC@F16, unpredictable results may occur. This happens because the stored information that was exchanged between the online job and the batch job, and that is older than 90 days, has been erased by the Reorganize job.

Deleted record space in file DC@F16 is also reorganized and removed.

The Reorganize job also clears all temporarily saved working lists that have reached the end of their retention period from a LANSA internal file named DC@F80. This file is used to store saved working lists. Refer to List Handling Built-In Functions in the list of LANSA Built-In Functions by Category in the *Technical Reference Guide* for more information on saving, restoring, and deleting working lists.

Deleted record space in file DC@F80 is also Reorganized and removed.

The Reorganize job may optionally reorganize the shared data files which exist in library QGPL. These files are shared by all LANSA systems which exist on this machine. These files are typically used in communication activity and include some LANSA for the WEB files.

5.6.3 Steps to Reorganize the Database

The steps involved in Reorganizing the LANSA internal database are as follows:



5.6.4 Submit Job to Reorganize the Database

Select the option to *Submit job to reorg LANSA internal database* from the Housekeeping menu. A screen similar to the example following will be presented:

DC@P400201	Reorganize Internal database
Submit this job	<u>YES</u> YES, NO
Using Job name Job description Job queue Output queue	
Change work spaces Total FFC	

Remove Multilingual support <u>NO</u> YES, NO Purge Developer Messages starting from date ... <u>00000000</u> (CCYYMMDD) Delete Deleted Partition Language Extensions. .. <u>NO</u> YES, NO Reorganize Shared LANSA system files NO YES, NO Purge Task History starting from date <u>00000000</u> (CCYYMMDD) Remove deleted users from LANSA files <u>NO</u> YES, NO

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=LastActDtl

The resulting screen requests that the submit request be confirmed and allows the job name, job description, job queue and output queue to be changed from their default values.

If you wish to attach this job to another process or submit it out of hours, then use the 5.6.5 Remotely Invoke the Reorganization Facility instead of using this option.

Note: The batch job submitted requires exclusive use of the entire LANSA database. As such it should be scheduled so that it executes at a time when no other user is using LANSA or has used LANSA since they signed on. If the file which are shared by LANSA systems are to be Reorganized, the batch job will required exclusive use of those files and therefore, no LANSA system on this machine should be in use when the job is scheduled to run. No harm will result from submitting and executing the batch job while other users are accessing LANSA, but it will not Reorganize the database or only partly Reorganize the database.

Input Options

The following input options apply when specifying details of the job to be submitted to Reorganize the LANSA internal database

Submit This Job Using Job Name Job Description Job Queue Output Queue

Change Work Spaces (PARM01)

Allows you to control the number of concurrent Full Function Checks and Compiles that can be run at any one time. When the LANSA system is first installed there is a sufficient number of members available for 99 concurrent Full Function Checks and 20 concurrent Compiles.

This field is defaulted to NO but can be changed to YES. If YES is entered, the number of concurrent Full Function Checks and Compiles that can be run at any one time will be changed to the number specified in the 'Total FFC' and 'Total Compiles' parameters by the Reorganize job.

If directly invoke from the LANSA command or the program the valid values are Y, N and blanks. The LANSA command PARM01 is mapped to this.

If blanks are passed, the default value of 'N' will be used.

Total FFC (PARM02)

Requests the maximum number of concurrent Full Function Checks that can be run at any one time.

This field is defaulted to 99, but can be changed to any number between 1 to 99. If directly invoke from the LANSA command or the program the valid values are 1 to 99 and blanks. The LANSA command PARM02 is mapped to this.

Total Compiles (PARM03)

Requests the maximum number of concurrent Compiles that can be run at any one time.

This field is defaulted to 20, but can be changed to any number between 1 to 99. If directly invoke from the LANSA command or the program the valid values are 1 to 99and ''. The LANSA command PARM03 is mapped to this.

If blanks are passed, the default value of '20' will be used.

Remove Multilingual Support (PARM04)

Removes unwanted multilingual objects for this system. USE THIS OPTION WITH EXTREME CAUTION.

This field is defaulted to NO but can be changed to YES.

If directly invoked from the LANSA command or the program, the valid values are Y, N or blanks. The LANSA command PARM04 is mapped to this.

If blanks are passed, the default value of 'N' will be used.

Purge Developer Messages Starting from Date (PARM05)

Allows you to specify a start date from which temporary developer messages can be purged from the system.

All Temporary developer messages that have a creation date equal to or less than the start date will be deleted from the system.

This field is defaulted to 00000000, but can be changed to any date. There is no date validation performed for the specified start date so please ensure the date entered is valid and is in CCYYMMDD (Century, Year, Month and Day) format: That is, a start date of 19911231 would be the 31st of December 1991.

If directly invoke from the LANSA command or the program the valid values are date in CCYYMMDD format blanks. The LANSA command PARM05 is mapped to this.

If blanks are passed, the default value '00000000' will be used.

Delete Deleted Partition Language Extensions (PARM06)

When a language has been deleted from a partition the records in the multilingual extension files are not deleted. If this option is set to 'YES' the Reorg will check the languages for each partition and will delete any extensions for languages which have been deleted. The extensions are descriptions and help created using a particular language, for fields, files, processes and functions. These extensions belong to a particular partition/language combination.

For example Language 'AAAA' may be used in partitions 'XXX' and 'ZZZ'. Language 'AAAA' is deleted from partition 'XXX' only the extensions for partition 'XXX', language 'AAAA' are deleted by this option.

This option is defaulted to 'NO', but can be changed to 'YES'.

If directly invoke from the LANSA command or the program the valid values are N, Y and blanks. The LANSA command PARM06 is mapped to this. If blanks are passed the default value 'N' will be used.

Warning: Back up of the LANSA database should be reviewed before using this option in case you may wish to reinstate the language to the partition.

Reorganize Shared LANSA system files (PARM07)

Various files are shared by LANSA systems that exist on this machine. Typically these files are used for communication and include various LANSA for the Web files. These files exist in a common library sometimes called QGPL, sometimes called DC@COMMLIB, but can be another name. These files will be Reorganized if this option is set to 'YES'. This will require exclusive use of these files and therefore no LANSA system that shares the files in this library on this machine should be in use when this job is scheduled to run.

If directly invoke from the LANSA command or the program the valid values are N, Y or blanks. The LANSA command PARM07 is mapped to this.

If blanks are passed, the default value of 'N' will be used.

Purge Task History Starting from Date (PARM08)

Allows you to specify a start date from which task tracking history can be purged from the system.

The purging of task history is controlled using:

• The Retain Task History flag in position 639 of the extended information data area DC@A07

and

• The Purge Task History Starting from Date

When the Retain Task History flag is set to N, all task tracking history that has a creation date equal to or less than the Starting from Date will be deleted from the system. If the date is set to zeroes, the history is not purged. Similarly, if the

flag is set to Y the history is not purged.

This field is defaulted to 00000000 if Retain Task History flag is Y and defaulted to today's date if Retain Task History flag is N. The field can be changed to any date. There is no date validation performed for the specified start date so please ensure the date entered is valid and is in CCYYMMDD (Century, Year, Month and Day) format. That is, a start date of 19911231 would be the 31st of December 1991.'

If directly invoke from the LANSA command or the program the valid date in CCYYMMDDY and blanks. The LANSA command PARM08 is mapped to this.

If blanks are passed, the default value of '00000000' will be used.

You can set the Retain Task History flag via Retain task history on REORG in *Work with Task settings*.

Remove deleted users from LANSA files (PARM09)

Allows you to remove those users from the LANSA files who no longer have an IBM i user profile.

If this option is set to 'YES' or 'Y', each user on the LANSA security file is checked. If an IBM i user profile does not exist, the user is removed from the LANSA files. For details of the information removed, refer to Remove LANSA User.

If directly invoked from the LANSA command or program, the valid values are N, Y and blank. The LANSA command PARM09 is mapped to this.

If blanks are passed, the default value of 'N' will be used.

5.6.5 Remotely Invoke the Reorganization Facility

To invoke the Reorganize job remotely, or to include it in existing end of day/week jobs or with one of the automatic job scheduling facilities, you can do so using variations of the LANSA command or by calling the LANSA entry point program like this:

LANSA REQUEST(REORG)

or

LANSA REQUEST(REORG) PARM01(Y) PARM02(99) PARM03(20) or

LANSA REQUEST(REORG) PARM01(Y) PARM02(99) PARM03(20) PARM04(N) PARM05(CCYYMMDD)

Note that two conditions apply when you run this job: You will need the profile of a security officer/partition security officer of QOTHPRDOWN or a security officer in that group. and

No job should be accessing LANSA.

If you prefer manually submit the Reorganize job, refer to 5.6.4 Submit Job to Reorganize the Database.

For details of the parameters used with this command, refer to 5.6.4 Submit Job to Reorganize the Database:

PARM01 Change Work Spaces (PARM01)

PARM02 Total FFC (PARM02)

- PARM03 Total Compiles (PARM03)
- PARM04 Remove Multilingual Support (PARM04)
- PARM05 Purge Developer Messages Starting from Date (PARM05)
- PARM06 Delete Deleted Partition Language Extensions (PARM06)
- PARM07 Reorganize Shared LANSA system files (PARM07)

PARM08 Purge Task History Starting from Date (PARM08)

PARM09 Remove deleted users from LANSA files (PARM09)

5.7 Changing Partition Being Worked With

5.7.1 Steps to Change System Partitions

5.7.2 Change the System Partition Being Worked With

5.7.1 Steps to Change System Partitions

The steps involved in changing the system partition being worked with are as follows:



The concept of a system partition and what they are used for is described in detail in 5.8 System Partition Definitions.

5.7.2 Change the System Partition Being Worked With

When the option to change the partition being worked with is selected from the housekeeping menu a screen similar to the example following will be presented:

DC@P400401 Change System Partition being used

Sel	Partition	Description
_	XXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The upper right area of the screen displays the name and description of the partition that is currently being used.

Displayed in the lower part of the screen is a list of all other partitions that you are authorized to use.

• Either select the partition that you wish to work with/in from the list displayed by entering any non-blank character beside it in the column headed 'Select'. Once this is done the housekeeping menu will be re-displayed and you will now being working with/in the selected partition.

or

• Press enter without selecting any partition. In this case the housekeeping menu will be re-displayed and you will still being working in the same partition (since no new partition was selected).

Warning: If you change partitions and return to the housekeeping menu to find it totally blank (i.e. no entries), it is probably because

you are not authorized to use any of the menu entries in the new partition. Remember that each partition has its own independent security system. Arrange for the security officer to grant you the required access to the partition.

5.8 System Partition Definitions

A system partition is a means of 'dividing up' or 'partitioning' one LANSA system. Each partition is completely separate from other partitions.

Points to note about partitions:

- There is very little 'inter-partition' communication. Files, fields and processes defined in one partition are not directly accessible from another partition.
- Each partition has a separate dictionary. Having too many partitions violates the concept of a corporate dictionary and increase maintenance.
- Each partition has a separate security system. If you have too many partitions or too many users accessing multiple partitions this can become a maintenance problem.

Generally partitions should only be used to:

- Separate large classes of users from one another (e.g.: production from development).
- Create 'standalone' application systems or software packages.

Generally partitions should NOT be used to:

- Divide up applications or software packages unless they are (and will remain) completely independent from one another.
- Satisfy a developer's urge to start new applications with a 'blank page'. The downstream effect of satisfying such an urge may be poor reuse of existing objects and unnecessary and avoidable maintenance and deployment issues.

Also see

- 5.8.1 Steps to Create or Change a Partition
- 5.8.2 Partition Definition Maintenance
- 5.8.3 Partition Definitions Create, Change or Delete
- 5.8.4 If you Enable a Partition for Full RDMLX
- 5.8.5 If you Comply with SAA/CUA Standards and Guidelines.

5.8.1 Steps to Create or Change a Partition

The steps involved in creation, review, change or deletion of a system partition definition are:



5.8.2 Partition Definition Maintenance

When the option to create or change a partition definition is selected from the housekeeping menu a screen similar to the example following will be presented:

DC@P400402 System Partitions

Select partition to be reviewed, changed or deleted

Or use the ADD command key to add a new partition

Sel	Partition	Description
_	XXX	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=LastActDtl

A list of all existing system partitions is displayed in the lower part of the screen.

• Select the partition that you wish to review, change or delete from the list displayed by entering any non-blank character beside it in the column headed 'Select'. Once this is done, full details of the partition will be displayed.

or

• Use the ADD function key to indicate that a new partition is to be defined. Once this has been done a screen will be presented that allows specification of the details of the new partition.

5.8.3 Partition Definitions - Create, Change or Delete

When an existing partition has been selected for display (and possible change or delete) or the add function key has been used to indicate that a new partition is to be defined, the System Partition screen is displayed.

DC@P400403 XXXXXXX System Partition Partition identifier : XXX Module library : XXXXXXXXXXX Unique object prefix : XSecurity officer Default file library : <u>XXXXXXXXX</u> Include in library list ? <u>NO</u> Initial public access : <u>NORMAL</u> Lib for Help Panel Groups: <u>XXXXXX</u> Copy system fields from: SYS SAA/CUA standards apply? YES Multilingual support ? YES Help option on menus : <u>Display process or function HELP text</u> Return prefix on menus : <u>Return to</u> Exit option on menus : <u>Exit from system</u> Keep translated RPG ? <u>NO</u> in source file _____ in library _____ Keep translated DDS ? <u>NO</u> in source file _____ in library _____ Configure Task Tracking: <u>NO</u> / <u>NO</u> Enable for full RDMLX : NO Enable Documentor : NO

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=LastActDtl Fnn=Chan

Enforce User Access in VL: NO		
Ignore Propagated Deletes: NO		
Check Before Propagating : NO		
Force *ENDWHERESQL : NO		
Enable Short Char : : 0		
Enable Long Names : YES		
Target Database Products		

DB2 IBM i	: YES
Sybase	: YES
SQL Server	: YES
ORACLE	: YES

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=LastActDtl Fnn=Chan

Note: The change and delete command keys are only enabled when reviewing the definition of an existing partition. The *Initial public access* and *Copy system fields* options only appear when you are defining a new system partition.

Working from the System Partition it is possible to:

- Input details of a new partition.
- Review details of an existing partition.
- Use the change function key to indicate that the partition definition is to be changed. If this option is used the screen will be re-presented with some fields input capable. Make the desired changes and press enter to proceed.
- Use the delete function key to indicate that the partition is to be deleted. If this option is used, a screen will be displayed to submit the deletion job to batch. You must confirm that the partition is to be deleted by entering YES. No other job should be using the partition when the deletion job executes. A user profile which is the LANSA system owner or part of that group should be used to execute the deletion job. It is recommended that a REORG of the LANSA system be run after the deletion has completed. For REORG details, refer to 5.6 Reorganizing the LANSA Internal Database. This deletion will

not remove objects from the partition's module library or the partition's file library. These libraries or the LANSA objects should be deleted using IBM i commands.

Note: The system partition SYS cannot be deleted. Neither can the partition that is currently being used/worked with.

Input Options

The following input options apply when specifying details of a new partition definition or changing details of an existing partition definition:

Partition Identifier

Specifies identifier / mnemonic assigned to the partition. Must be 3 characters long and consist of 'A' -> 'Z', '@', '#', '\$', '0' -> '9' only. It must be unique. No two partitions can have the same identifier.

Partition Description

Specifies the description of the partition. Must not be blank.

Module Library

Specifies the name of the library in which compiled RDML programs associated with this partition are kept. The library must currently exist and must not be the same as the module library used by any other partition. Once specified, the name of the module library cannot be changed. Do not use the LANSA system program and data libraries.

Unique Object Prefix

Specifies a unique object naming prefix that is to be used by LANSA for this partition. This prefix is used internally by LANSA within its database and externally in the name of the compiled RPGIII programs it produces from RDML source statements.

The prefix must be alphabetic in the range 'D' - 'Z', '@', '\$' or '#', or 0 - 9 and unique. No two partitions can have the same prefix. Once established the prefix for a partition cannot be changed.

Security Officer

Specifies the name of the IBM i user profile who is to be the security officer for the partition. The profile must exist and does not have to be QSECOFR.

A user profile that is nominated as a partition security officer does not have any special rights in other partitions or outside of the LANSA system.

Default File Library

Specifies the name of the library that is to be the default library for new files created in this partition. Note that this is a default value only and does not restrict users of the partition from creating files in other libraries. Do not use the LANSA system program and data libraries.

Warnings: Avoid using the library specified here as the IBM i 'current library' in any interactive or batch job.

Include in Library List

Requests that you specify whether or not the default file library for this partition is to be automatically included in the job's library list when accessing this partition.

Allowable values are:

- NO The default file library is not to be included into the job's library list.
- YES The default file library is to be included into the job's library list.

The default value is NO.

Warning

If the default file library is changed, all Visual LANSA slaves will require Files that have a library equal to the previous default file library to have JUST THE OAM re-built. This will retain existing data. If the table is re-built, existing data will be lost.

Initial Public Access

Appears only when defining a new partition. Specifies what access to the partition is to be allowed to other system users initially. Allowable values are:

NORMAL	Other system users can use this partition but cannot change its
	definition or delete it.

- ALL Other system users can use this partition and can change its definition or delete it.
- NONE Other system users cannot access this partition.

After a partition has been created individual or public access to it can be displayed or modified by the option available on the Housekeeping menu.

Library for Help Panel Groups

This item is shown only if Panel Groups are used in this LANSA system for the presentation of HELP text (see section on 'using Panel Groups for HELP text'). The specification of a library name is optional. If no library is specified, any Panel Groups created in monolingual systems, or the Panel Groups created for the default language in multilingual systems, will be placed into the partition's module library. If a library is specified, these Panel Groups created will be placed into this library. The library does not need to exist when the name is specified here. It will be created, if required, when the first Panel Group is compiled.

This library will be added into the library list at the start of a LANSA session. It should be specified, if required, before panel groups are created for this partition, and should not be changed later. If the library specified here is changed after panel groups have been created, these Panel Groups should be moved to the new library, or deleted and re-created, so that the HELP text is found at execution time. This is the user's responsibility.

Copy System Field From

Appears only when defining a new partition. Optionally allows all 'system' field definitions to be copied from an existing partition into the new partition.

Certain fields defined within LANSA are required in any partition for it to be used effectively. Some examples include IO\$STS and @@UPID. In addition a number of user defined fields may be considered as vital and be flagged as 'system' fields. Refer to the field definition section of this guide for *details* of how system fields are defined.

The default value for this option is 'SYS', which indicates that all system fields from partition 'SYS' should be copied to this new partition.

Either accept the default of 'SYS' or specify the name of another partition from which the definition of system fields should be copied. Leave blank to avoid copying any system fields.

System frameworks and groups will also be copied from the nominated partition. For more information, refer to the Visual LANSA documentation.

SAA/CUA Standards Apply

Appears when defining a new partition or changing an existing partition. Requests that you specify whether or not SAA/CUA (System Application Architecture / Common User Access) standards apply to objects created within this partition. Allowable values are:

YES SAA/CUA standards apply to this partition.

NO SAA/CUA standards do not apply to this partition.

The default value is YES.

Before attempting to change or create a partition that uses SAA/CUA standards read in full 5.8.5 If you Comply with SAA/CUA Standards and Guidelines and IBM i SAA/CUA Implementation in the LANSA Application Design Guide.

Multilingual Support

Appears when defining a new partition or changing an existing partition. Requests that you specify whether or not this partition requires multilingual support.

It is recommended that all partitions are defined as multilingual even if they will be used with only one language.

Allowable values are:

YES Multilingual support is required for this partition.

NO Multilingual support is not required for this partition.

The default value is YES.

If you specify YES then you must also specify YES for the SAA/CUA standards apply option.

Before changing or creating a partition that uses multilingual support, read Setting Up LANSA to Run Multilingual Applications in the LANSA Multilingual Application Design Guide as well as the Note following.

Note:

There are some important things that you should know about multilingual support before you attempt to turn it on (or off):

- Providing (or not providing) multilingual support in your applications is an important application design decision that you should make decisively, even before LANSA installation. Changing your mind later (or changing your mind multiple times) may lead to unnecessary and avoidable maintenance and deployment issues. Refer to Imports with the LANSA IBM i Software in the *Installing LANSA on IBM i Guide* for a list of LANSA software that requires a multilingual partition.
- There are important procedures, considerations and guidelines that you

should understand and then follow when converting a partition from monolingual form to multilingual form (or vice versa).

- Changing a partition to multilingual or monolingual form impacts the repositories in attached Visual LANSA development workstations. You must take steps to ensure that your central repository and all attached Visual LANSA workstation repositories are changed in a synchronized fashion. In outline, the simplest way to ensure this synchronization is as follows:
 - Check in all new or modified objects from all attached Visual LANSA workstations into the central repository.
 - Delete the partition from all attached Visual LANSA workstations.
 - Change the central repository to/from multilingual form.
 - Make the changes to object definitions in the central repository as specified in the *LANSA Multilingual Application Design Guide*.
 - Set up the partition again on all attached Visual LANSA workstations as if it was a brand new partition, and then verify that it now has the correct multilingual characteristics (or not).
 - Check out all required objects to each attached Visual LANSA workstation.

Help Option on Menus

Appears when defining a new partition or changing an existing partition. Requests that you specify whether or not a 'help' option is to appear on process menus, and if it is to appear, what the text that is to be displayed should be.

The default is 'Display process or function HELP text'.

Change the text as required, particularly if you are running a system in a language other than English.

Otherwise specify *NONE (in uppercase characters) to indicate that the help option is not required on process menus. Note that the non-appearance of this option as a menu option does not prevent the user from using the help function key(s).

Recompile any compiled process menus after changing this option.

Return Prefix on Menus

Appears when defining a new partition or changing an existing partition. Requests that you specify whether or not a 'return' option is to appear on process menus, and if it is to appear, what the prefix of what is displayed should be. The default is 'Return to'.

Change the prefix as required, particularly if you are running a system in a language other than English.

Otherwise specify *NONE (in uppercase characters) to indicate that the return option is not required on process menus. Note that the non-appearance of this option as a menu option does not prevent the user from using the cancel function key.

Recompile any compiled process menus after changing this option.

Exit Option on Menus

Appears when defining a new partition or changing an existing partition. Requests that you specify whether or not an 'exit' option is to appear on process menus, and if it is to appear, what the text that is to be displayed should be.

The default is 'Exit from system'.

Change the text as required, particularly if you are running a system in a language other than English.

Otherwise specify *NONE (in uppercase characters) to indicate that the exit option is not required on process menus. Note that the non-appearance of this option as a menu option does prevent the user from using the exit function key.

Recompile any compiled process menus after changing this option.

Keep Translated RPG / Keep Translated DDS

Appears when defining a new partition or changing an existing partition. Requests that you specify whether or not:

- DDS (Data Description Specifications) translated from your LANSA requests should be permanently kept.
- RPG (Report Program Generator) source statements translated from your LANSA requests should be permanently kept.

The default for both these options is NO (source statements are not to be kept). Keeping source statements in most situations is just a waste of disk space.

However, if you specify YES to request that either translated DDS or translated RPG be kept, please take note of the following very important points:

• If you specify YES, you must specify both a source file name and a library name. *LIBL is not acceptable as the library name. The file and library name you specify are not validated in any way, so check what you specify carefully.

- The source files nominated should be created before they are made known to LANSA. Use the CRTSRCPF (Create Source Physical File) command to do this. The existence of the nominated files is not checked.
- The same source file should not be specified for the storage of both DDS and RPG. This is not checked.
- No two partitions should share the same source files for the storage of translated RPG or DDS. This is not checked.
- Stored translated DDS or RPG is not automatically imported or exported by the LANSA export/import facilities. If this facility is required it must be defined to the export/import routines as if 'non-LANSA' objects are being shipped. The setup and correct execution of such facilities is a user responsibility.
- When LANSA file, process or function definitions are deleted, any associated translated DDS or RPG is not removed from the source files specified.
- When LANSA file, process or function definitions are recreated or recompiled, any translated DDS or RPG is replaced by the newer version resulting from the recreate or recompile.
- Translated RPG may be moved to another CPU, but it cannot ever be successfully executed by moving it this way, even if LANSA is resident on the target machine.

The only way to move field, file, process or function definitions between machines is via the LANSA import/export facilities.

• Translated RPG code is not intended for maintenance by 'human beings'. The translated code is very cryptic and would be very difficult to effectively maintain at the RPG level.

Remember that LANSA is based on a fully procedural RDML language. All development and maintenance work should be done at the RDML level. RPG is used as a 'vehicle' to produce executable program objects. Its existence should be invisible and immaterial to RDML application programmers.

LANSA is not designed to be an RPG generator. This is why the word 'translated' is used throughout this section.

• If stored DDS or RPG are manually modified and then used to recreate or recompile LANSA objects, all maintenance and support for the object(s) involved will be suspended. If you create problems by doing this, your product vendor may help you to correct them, but is in no way obliged to do

so.

This table identifies the types of source members that may be placed into source files:

Type of Source	Type Of Output	Member Name/Comments
File definition	PF/DDS LF/DDS	Same as physical file and logical file(s) involved.
Field Reference File	PF/DDS	Same as field reference file name specified. Not really a LANSA object, but DDS may be kept.
Process definition	DSPF/DDS RPG	Both have the same name as the process involved.
Function definition for an old style function that does NOT use *DIRECT.	DSPF/DDS PF/DDS RPG RPG	Display file: @innnnn External data: E@innnnn RDML function: F@innnnn RDML multilingual setup module: F@innnnML RDML GUI control module: F@innnnGI Extra member for *MLOPTIMISE: F@innnnUM
Extra members when *DBOPTIMIZE used in function to store 'inline' I/O logic.	RPG RPG RPG RPG RPG RPG RPG	/COPY F specs: F@innnnFM /COPY E specs: F@innnnEM /COPY I specs: F@innnnRM /COPY I specs: F@innnnIM /COPY C specs: F@innnnCM

		/COPY O specs: F@innnnOM /COPY arrays: F@innnnAM
Function definition for a new style function that uses the *DIRECT option.	DSPF/DDS PRTF/DDS PF/DDS RPG RPG	Display file: @fffffff Printer file: \$fffffff External data: E@fffffff RDML function: @ffffffff RDML multilingual setup module: @fffffffML RDML GUI control module: @fffffffGI Extra member for *MLOPTIMIZE: @fffffffUM
Extra members when *DBOPTIMIZE used in function to store 'inline' I/O logic.	RPG RPG RPG RPG RPG RPG RPG	/COPY F specs: @fffffffFM /COPY E specs: @fffffffEM /COPY I specs: @fffffffRM /COPY I specs: @ffffffffM /COPY C specs: @fffffffCM /COPY O specs: @fffffffOM /COPY arrays: @fffffffAM

where:

innnnn is the internal function identifier derived from field F23FID in file DC@F23.

fffffff is the actual 1 to 7 character function name derived from field F23FMT in file DC@F23.

Note: The external data member is used to create an externally described data structure in library QTEMP during compiles. After the compile completes it ceases to exist. It is used to ensure that even-length packed fields are handled correctly.

Configure Task Tracking

Displayed when defining a new partition or changing an existing partition. Configure task tracking is split into the following six questions, each requiring a YES or NO answer. These answers will define how task tracking will operate within this partition.

1. Is Task Tracking active within this partition?

Allowable values are:

- YES Task Tracking is active in this partition, object and task authority checks will be performed and all events that have taken place for work performed on objects will be recorded.
- NO Task Tracking is not active in this partition.

The default value is NO.

2. Does the user require a task identifier to do work?

Allowable values are:

- YES User does require a task identifier before any work can be performed on a selected object.
- NO User does not require a task identifier to do work, until work has been completed on a selected object, where on he/she will be prompted by a POPUP window to allocate a task identifier against the selected object.

The default value is NO.

Note: This option will be ignored if Task Tracking is not active in this partition.

3. Prompt/Confirm task identifier?

Allowable values are:

- YES Prompt/confirm task identifier is required when work has completed on a selected object. The user will be prompted with the 'Prompt/Confirm POPUP Window' to confirm or change (if CHANGE function key is enabled) the task identifier to be allocated for work performed on the selected object.
- NO Prompt/confirm task identifier is not required. The task identifier specified for this job (on entry into LANSA or by a previous request to specify a task identifier for the job via the 'Prompt/Confirm POPUP Window'), will be automatically confirmed as the task identifier for

work performed on the selected object.

The default value is NO.

Note: This option will be ignored if Task Tracking not active in this partition.

4. Allow user to change tasks while working?

Allowable values are:

- YES User is allowed to change the task identifier that is allocated to the selected object on which work was performed but only if the user is authorized to the task identifier, OR, if the 'Prompt/Confirm POPUP Window' is displayed, the CHANGE function key will be enabled to allow the user to change task identifiers manually.
- NO User not allowed to change tasks.

The default value is NO.

Note: This option will be ignored if Task Tracking is not active in this partition.

5. Disable Special 'Work with Tasks' Security?

Allowable values are:

YES Disable the special security checks within the 'Work with Tasks' option.

NO The special security checks within the 'Work with Tasks' option are active.

The default value is NO.

This option will be ignored if Task Tracking is not active in this partition.

6. Activate Task Tracking for Import jobs?

Allowable values are:

YES Task tracking is active for import jobs

NO Task tracking is not used for import jobs.

The default value is NO.

This is only used if Task Tracking is active in a partition.

Note: Special security checks ensure that only QSECOFR, the Partition Security Officer or a user belonging to one of these groups has access to the Work with Tasks option. Whether special Work with Tasks security is disabled or not, normal menu security still applies to Work with Tasks.

Task Identifiers are allocated using the 5.5.2 Create a Task Identifier option on

the Work with Tasks menu.

When Task Tracking has been activated for a partition, invoke LANSA by using the LANSA command and passing the partition parameter (PARTITION) and the task identifier parameter (TASK_ID). The following example demonstrates how to invoke LANSA when Task Tracking is active:

LANSA PARTITION(DEM) TASK_ID(0000001)

Before attempting to configure Task Tracking please read the section that details the requirements and considerations of 5.4 Task Tracking in LANSA.

Enable for full RDMLX

This option appears when defining or changing an existing partition so that you can specify whether or not this partition is to allow full RDMLX objects.

Before changing or creating a partition that is enabled for full RDMLX, read the RDML and RDMLX Partitions Concepts for details of the requirements and conditions for RDMLX development.

Allowable values are:

YES full RDMLX is required for this partition.

NO full RDMLX is not required for this partition.

The default value is NO.

If you specify YES, you will be asked to confirm your selection. Once a partition is enabled for full RDMLX:

- it cannot be returned to a non-RDMLX state
- all development must be done in Visual LANSA
- development in LANSA for i is NOT permitted.

If you select YES to enable the partition for full RDMLX, you will need to specify further options, as described in 5.8.4 If you Enable a Partition for Full RDMLX.

Enable Documentor

Specifies that LANSA/DOCUMENTOR is to be enabled in this partition. Before enabling Documentor, please read the section that explains the requirements and considerations of using the Documentor.

Enforce User Access in Visual LANSA

This option can be used to control User Access in slave Visual LANSA systems. Enter YES if you wish to control User Access in the slave Visual LANSA systems in accordance with defined User Access in this IBM i master system partition. The default is NO.

Ignore Propagated Deletes

This option can be used to control the processing of propagated deletes in slave Visual LANSA systems. Enter YES if you wish propagated deletes to be ignored in the slave Visual LANSA systems.

The default is YES.

Check Before Propagating

This option can be used to control the processing of propagated deletes in slave Visual LANSA systems. Enter YES if you wish referential checks to be performed before actioning propagated deletes on the slave Visual LANSA systems. This will prevent, for example, the deletion of a field that has been used in a local file.

Force *ENDWHERESQL

For use by Visual LANSA with the SELECT command to differentiate between the use of SQL or native I/O access. Refer to the OPTIONS parameter in the SELECT Parameters for details.

The default is NO.

Enable Short Char

Select *Enable Level One* through to *Nine* as required.

Default is zero (which is Disable).

For details, refer to Enable Short Char in the *Visual LANSA Administrator's Guide*.

Changing a partition's *Short Char Level* can alter the structure of LANSA working lists that contain one or more fields of type string and/or char. This altered structure is not compatible between LANSA objects that share the working list unless all these objects are rebuilt. If you do change it, you must rebuild all the files, functions, forms and reusable parts in the partition in order to avoid unpredictable behaviors at runtime.

Enable Long Names

Apart from setting this option, Yes or No, long names, if used, are entered via the Visual LANSA system.

For details, refer to *Enable Long Names* in the *Visual LANSA Administrator's Guide*.

Target Database Products

Enter YES for Database Products you wish to use with your IBM i or Visual LANSA application.

Different database products have different requirements for field types, lengths, etc. Setting these flags will help ensure that files created in your application can be supported in the database.

5.8.4 If you Enable a Partition for Full RDMLX

This section only applies if you are modifying an existing partition or creating a new partition to be enabled for full RDMLX.

Before enabling a partition for full RDMLX, review RDML and RDMLX Partitions Concepts to understand the implications of this option.

If you choose to enable a partition for RDMLX, then you will be presented with additional screens to specify defaults for object creation in the RDMLX-enabled partition.

DC@P400410 Add RDMLX Field Defaults **Create as full RDMLX** Components : YES Function : YES Fields : YES Fields : YES Allow in Keyboard Edit Fields Default Shift Length Dec Cde A Alphanumeric YES *BLANK 10 0 Input Attributes : FE **Output** Attributes : **H** Character YES 10 0 *NULL Input Attributes : FE **Output Attributes :** P Packed *****ZERO 15 0 3 YES Input Attributes : FE RB **Output Attributes : S** Signed YES *ZERO 15 0 3 Input Attributes : FE RB **Output Attributes : D Datetime** YES *SQLNULL 26 0 Input Attributes : ASQN FE ISO SUTC **Output Attributes :** +Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=LastActDtl

Create full RDMLX Components, Functions, Fields, Files

The options enable you to set the defaults for full RDMLX status for these object types in this partition. This default value will be used when creating the objects in the Visual LANSA development environment.

The default is YES.

A value of YES means that this object type will be enabled for full RDMLX when created. You may change the default value for the particular object being created.

Defaults by Field Type

Various defaults and enabling flags may be specified for each field type. These defaults will be used when creating fields in this partition. Only those field types which are flagged as enabled will be able to be created in this partition.

Allow in Fields

Allowable values are:

YES Fields of this type may be created in this partition.

NO Fields of this type may not be created in this partition.

If your partition has existing field types, such as A (alphanumeric), P (packed) or S (signed), it is recommended that these types remain enabled to ensure maintenance of existing code is not restricted.

Default values may be specified for:

- Field default value
- Keyboard shift
- Length
- Number of decimal places
- Edit code
- Input attributes
- Output attributes

These defaults will be used when fields of this type are created in this partition. Input and output attributes for field types A (alphanumeric), P (packed) or S (signed) will be set initially to the system level values. They may be changed. The values set will be used in this partition.
5.8.5 If you Comply with SAA/CUA Standards and Guidelines

The following section only applies if you are modifying an existing partition or creating a new partition to comply with SAA/CUA standards and guidelines.

Before attempting to create or use an SAA/CUA compliant partition read the chapter in the *LANSA Application Design Guide* that describes in full the IBM i SAA/CUA Implementation used by LANSA.

When you are changing an existing partition to implement SAA/CUA standards or creating a new partition that will use SAA/CUA standards a series of additional screen formats will be presented which will request details of your SAA/CUA implementation.

SAA/CUA Defaults Recommended Value And Comments N - Select by number or cursor Process menu style Enter number of function required or place Process menu prompt line cursor on same line: Process title in upper N (upper and lower allowed) case N (upper and lower allowed) Function title in upper case Display panel Y (in CUA 1989 non-action bar systems) identifiers N (in CUA 1989 action bar systems) Message line number 22 (above function keys) Function key line 23/24 (below messages) number 1/2Show process date and N (not required) time N (not required) Show function date and time Process auto-record Y (yes)

The following information tables describe the recommended answers to all questions asked on these additional screen formats:

advance	
Function auto-record advance	Y (yes)
Pull down auto record advance	Y (yes)
Window border fill characters	'(Sides, Top/Bottom, Top
OR	:: corner and Bottom corner)

Note 1: If you plan to use reverse video borders, the use of the blank (i.e. X'40') character for all four of these window fill characters is strongly recommended.

Note 2: Compiled pop-up window applications are locked into these fill characters at compile time, so a change to any value may mean that applications need to be recompiled before the change will be completely visible to end users.

SAA/CUA Color/Attributes Defaults

The following color and attribute table describes the default colors and attributes adopted by LANSA itself, and for applications created within SAA/CUA partitions.

Panel Element Category	Color and Additional Color Screen Attributes	Additional Mono Screen Attributes
ABCH Action bar and pull-down	White	High Intensity
PBPT Panel title	Blue	
PBPI Panel identifier	Blue	
PBIN Instructions to user	Green	
PBFP Field label/description	Green	

You are not forced to follow these default values.

PBBR Brackets	Green	
PBCM Field column headings	Blue	High Intensity
PBGH Group headings	Blue	High Intensity
PBNT Normal text	Green	
PBET Emphasized text	White	High Intensity
PBEN Input capable field (normal)	Green, Underline	Underline
PBEE Input capable field (emph)	White, Underline	Underline, High Intensity
PBCH Choices shown on menu	Green	
PBSC Choice last selected on menu	White	High Intensity
PBUC Choices not available	Blue	
PBCN Protected field (normal)	Green	
PBCE Protected field (emphasized)	White	High Intensity
PBSI Scrolling information	White	High Intensity
PBSL Separator line	Blue	
PBWB Pop-up window border	Blue	
FKCH Function key information	Blue	

Color codes you should use are	White- WHT
	Green- GRN
	Blue- BLU
Attribute codes you should use are	High Intensity - HI
	Underline- UL

For other color and attribute codes that are available, refer to the field definition section of this guide.

SAA/CUA Function Key Defaults

The table below indicates the standard function key assignments used by the LANSA implementation under SAA/CUA, the suggested short form descriptions and the suggested function key that should be assigned.

You are not forced to follow the recommended values and can change them to your site standards if desired.

Notes	Description Of Function Key	Short Form Description	Function Key No
Re	Cancel current action	Cancel	12
Re	Enter any entry fields	Enter	RA
Re	Request help for the current function	Help	01
Ор	Display choices from a list of entries	Prompt	04
Op *	Restore or refresh panel	Refresh	05
***	LANSA defined messages key	Messages	14
Op *	Scroll panel backward	Bkwd	07
Op *	Allow entry of a command	Command	09
Op *	Display cycle of function key	Keys	13

	area forms		
Re	Exit to the highest level	Exit	03
Re	Exit to the next level above	Exit	03
Op *	Move cursor to first field on the screen	Home	НМ
Op *	Scroll panel forward	Forward	08
Ор	Display table of contents for help	Contents	23
Ор	Provide information about entire panel	Ex Help	02
Op *	Provide index of help information *	Index	11
Op *	Provide names and functions of keys *	Keys help	09
Op *	Scroll panel left *	Left	19
Op *	Re-display last command that was issued *	Retrieve	09
Op *	Scroll panel right *	Right	20
Op *	Move cursor backward *	Switchbkwd	18
Op *	Move cursor forward *	Switch fwd	06
Ор	Move cursor to action bar	Actions	10
***	LANSA defined add key	Add/Create	06
***	LANSA defined change key	Change	21
***	LANSA defined delete key	Delete	22

Notes about this Table

Re indicates an SAA/CUA 'reserved' function key. In such cases the

function key cannot be reassigned to other functions, even if the specified functions don't apply to a particular panel.

- Op indicates a 'non-reserved' SAA/CUA function key. In such cases the function key may be reassigned to other functions, but only if the application does not support the functions proscribed for the function key by this table.
- * indicates an SAA/CUA function key assignment that is proscribed for complete SAA/CUA compatibility, but is not currently implemented directly within LANSA. The use of such keys is controlled by the USER_KEYS parameter of DISPLAY and REQUEST commands. It is the responsibility of the user to implement such key assignments in RDML programs as per the SAA/CUA guidelines described by the table above.
- *** indicates a non-SAA/CUA proscribed function key that is required in this table because it is implemented by LANSA.

5.9 Frameworks and Groups

Frameworks and groups are used in versions of Visual LANSA which support the Visual LANSA feature, to establish a link between related components. Groups are a development-oriented means of grouping similar items together and frameworks are a business-oriented grouping of items.

When components are exported, frameworks and groups are automatically exported with them. See the section in this chapter on exporting and importing. Both are maintained from a similar maintenance screen.

DC@P400	407		Partition
Partition	:]	DEM -	Demonstration partition
Identity 99999 -	10	Sys Y Den	Framework Short & Long Names DEMONSTRATION nonstration System Framework

F1=Help F3=Exit F12=Cancel F14=Messages F17=LastActDtl F21=Change

Input Options

To add or change frameworks or groups use the F21 function key. In the case of system frameworks and groups, only the names can be changed.

Identity

The identity number is made up of two parts. The first is the serial number of the server on which the framework or group was created, with any letters removed. The second is a user assigned number to make the identity unique within the partition.

Sys

Use this field to indicate that this is a system framework or group.

System frameworks and groups cannot be deleted. If you are creating a new partition and copying system fields from a nominated partition, the system

frameworks and groups in that partition will also be copied.

Short and Long Names

The short and long names must be entered. The short name must be in uppercase but the long name can be in mixed case.

Framework and Group Deletion

To delete a non-system framework or Group, blank out the identifier, system flag, short name and long name on the maintenance screen and press the Enter key. System frameworks and groups cannot be deleted.

5.10 Partition Multilingual Attributes

In a multilingual partition, partition multilingual attributes may be specified that define any number of languages (from any of the language groups) that are to be supported in this partition. Partition multilingual attributes include, for each language, the language code and description, the type of language support required (if DBCS or RLTB) and translations in the required language for menu options and functions key descriptions.

5.10.1 Steps to Review or Change Partition Multilingual Attributes 5.10.2 Review or Change a Partition's Multilingual Attributes

5.10.1 Steps to Review or Change Partition Multilingual Attributes

The steps involved in reviewing or changing the partition multilingual attributes are as follows:



5.10.2 Review or Change a Partition's Multilingual Attributes

When you choose the option to review or change a partition's multilingual attributes on the housekeeping menu, a screen similar to the following will result:

DC@P40	0501	Multilingual Partit	ion Languages		
Multiling	Multilingual Partition : XXX				
Select la	nguage to re	view/change or use	e Add key to add a new language		
Sel _ _ _ _	Language ENG DEU TCHI HEB	Description English German Traditional Chine Hebrew	Default YES se +		
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages					

From the Multilingual Partition Languages screen it is possible to:

- Review the multilingual information associated with a partition.
- Change the multilingual information associated with a partition.
- Delete the multilingual information associated with a partition.

Select the language(s) that you wish to review or change or use the Add function key to add a new language that is to be supported in this partition.

On the following screens, you may use the CHANGE key to place the screen in change mode and make the desired changes.

DC@P400502 Display Multilingual Partition Language

Multilingual Partition : XXX

Language ENG Language description English

DBCS support required NO YES, NO IGCCNV required NO YES, NO Function key / description / Line number

RLTB support required NO YES, NO

PC Locale uppercasing required . . . NO YES, NO ISO language code en

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change Fnn=Delet

DC@P400503 Display Multilingual SAA / CUA Defaults

Multilingual Partition : XXX

Language ENG Language description English

Partition description Demonstration and Training Partition

Process menu prompt line Enter number of function required or place cursor on same line. Help option on menus Display process/function HELP text Return prefix on menus Return to Exit option on menus Exit from system Library for Panel Groups User message files

IBM i EBCDIC CCSID 00037 Windows ANSI code page 00819 Linux Code Page

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

Input Options

The following input options apply to changing a partition's multilingual attributes:

Language

A 1 to 4 character code that uniquely identifies the language within the partition. It should be the same code used for the same language in other partitions, allowing language matching on import and export runs. If it is a base language then it should be the same code as detailed in the *LANSA Multilingual Application Design Guide*. Recommended codes are also in the *LANSA Multilingual Application Design Guide*.

Language Description

A 1 to 20 character full description of the language. This is used on some reports as an unambiguous description of the language being used.

DBCS Support Required

Whether or not the language requires DBCS (double byte character set or ideographic) support to successfully implement applications using this language.

IGCCNV Support Required

Used to indicate whether IGC conversion (via the DDS keyword IGCCNV) is required for applications generated under this language in this partition. YES indicates IGC conversion is required. Any other value will indicate that IGC conversion is not required for this language.

Function Key / Description

If IGC conversion is required, specify the function key that is to be assigned to the IGC conversion task and a short form description of the function key. Choose a function key number in the range of 01 to 24 so that it does not conflict with any other CUA assigned function key in this partition. You must also provide a short form description of the function key.

Important Note: If you are using a DBCS language you may find that you cannot specify the function key description in the DBCS language initially. To solve this problem:

- Define the DBCS language in full, but specify NO for the IGC conversion option initially.
- Exit from LANSA and sign off. Sign on again and then invoke LANSA. Proceed to this function again.
- Review the multilingual attributes of the DBCS language again. Use the change function key. This time you should be able to specify the function key description, and other details on this screen panel, correctly.

Line Number

If IGC conversion is required, specify the line on the screen panel that the conversion area is to be placed.

It is recommended that the conversion area be placed on the same line as the message area. If your system is configured to show messages on line 22, it is advisable to also place the IGC conversion area on line 22.

RLTB Support Required

Whether or not the language requires right to left support (or bi-directional) to successfully implement applications using this language.

Default Language

Specifies if this language is the default language for this partition.

One of the languages defined using these options must be nominated as the default language. The default language is an important language and has the following implications:

- When an application is invoked without nominating the required language, the default language will be used.
- Initial data dictionary, database and application development must be done in the default language. For instance, if you specify a field description it is assumed that the description is in the default language.

Development Language

Specifies if this language is a development language for this partition. That is, applications can be developed in LANSA with all development panels being displayed in the nominated language.

Warning: The ENG (English) language is a development language by default and must not be specified as a development language.

You should contact your product vendor before indicating if any language is to be used as a development language to confirm its availability for use in development.

PC Locale uppercasing required

In a client/server configuration, this option is set on each PC during system initialisation on the PC.

This option is only used with fields created **without** the attribute LC (*Lowercase entry allowed*) specified. When LC (*Lowercase entry allowed*) is not specified, that field's input values in the range **a** to **z** will be converted to upper case (that is **A** to **Z**).

Set this option to:

YES if you wish to have characters outside the **a** to **z** range also converted to upper case, for example, from **à** to **À**

NO, if you want the special characters to remain as lower case while those in the **a** to **z** range are converted to upper case. This could be the result: RESUMé.

Default is NO.

ISO language code

The ISO 639-1 two-letter language code. It can be optionally qualified by an ISO 3166-1 two-letter country code. The language code or qualified language-country code must be a valid registered code, for example: **en** (English), **en-US** (English–United States) and **en-GB** (English–United Kingdom). This field is case sensitive. Values must be entered in the correct case (for example: **en-US**).

The ISO language code is commonly used in Web applications and when interfacing with external libraries that support localization.

For more information, refer to Input Attributes in the *Technical Reference Guide*.

Partition Description

Refer to Creating or Changing System Partition Definitions for *details*.

Process Menu Prompt Line

Refer to Creating or Changing System Partition Definitions for details.

Help Option On Menus

Refer to Creating or Changing System Partition Definitions for details.

Return Prefix On Menus

Refer to Creating or Changing System Partition Definitions for details.

Exit Option On Menus

Refer to Creating or Changing System Partition Definitions for *details*.

Library For Help Panel Groups

This item is shown only if Panel Groups are used in this LANSA system for the presentation of HELP text (see section on 'using Panel Groups for HELP text'). The specification of a library name is optional. If no library is specified, any Panel Groups created will be placed into a library with the generated name 'PN@ppllll' where ppp is the partition identifier, and llll is the language. If a library is specified here, any Panel Groups created for this language will be placed into this library. The library does not need to exist when the name is specified here. It will be created, if required, when the first Panel Group is compiled for this language. The name will then be put on file automatically. This library will be added into the library list at the start of a LANSA session. It should be specified, if required, before panel groups are created for this partition, and should not be changed later. If the library specified here is changed after panel groups have been created, so that the HELP text for this language is found at execution time. This is the user's responsibility.

User Message Files

Up to 10 user defined message files may be nominated for the language. They are used to execute OVRMSGF commands when entering applications defined within this partition.

Refer to the LANSA Multilingual Application Design Guide for detailed information on specifying message files.

IBM i EBCDIC CCSID

When text is moved from the client to the server and vice versa it must be translated from one system's code page to the other. Specify the CCSID to be used to translate text to the correct IBM i format for the selected language. If the code page is not entered then the default CCSID for the language will be used

by the LANSA system.

Windows ANSI code page

When text is moved from the client to the server and vice versa it must be translated from one system's code page to the other. Specify the code page to be used to translate text to the correct Windows format for the selected language. If the code page is not entered then the default code page for the language will be used by the LANSA system.

Linux code page

When text is moved from the client to the server and vice versa it must be translated from one system's code page to the other. Specify the code page to be used to translate text to the correct Linux format for the selected language. If the code page is not entered then the default code page for the language will be used by the LANSA system

Function Key Short Form Descriptions

Allows you to specify the short form description for all SAA/CUA function keys in the appropriate language. For details, refer to If you Comply with SAA/CUA Standards and Guidelines.

Delete Partition Language

The delete key will only delete the record in the LANSA internal file associating the language with the current partition. The language is then not available for use in the current partition. There may be many other records in the internal LANSA multilingual extension files, i.e.process and function descriptions in the now deleted language. Refer to the parameter *Delete Deleted Partition Language Extensions* in Submit Job to Reorganize the Database for further details.

5.11 The File Maintenance Utility

The file maintenance utility is a facility provided with the LANSA system that allows direct access to any file defined under the LANSA system.

The file maintenance utility can be used to:

- Add new records to a file.
- Inquire about individual file records by key. The selected records can be updated or deleted.
- Browse the file by full, partial or generic key. Individual records selected can be displayed in full, changed or deleted.

In addition, the file maintenance utility has the following features:

- LANSA object security checking is performed. Users cannot use the file maintenance facility to perform operations on a file they are not authorized to use.
- LANSA dictionary and file level validation checking is performed. Thus when the file maintenance utility is used to add, update or delete records in a file the action must satisfy all validation checks associated with fields in the file at the dictionary and file level.

5.11.1 When to Use or Not Use the File Maintenance Utility?

The file maintenance utility is provided as an aid to software developers for the entry of test data and the testing of file validation rules. As such, it is primarily designed to handle 'mainstream' processing requirements in database files set up and maintained by LANSA.

If it is used on files that contain 'non-mainstream' features, some minor problems may occur that require the construction (by application template) of purpose built (or tailor made) RDML functions to fully support your testing requirements.

Files that are under commitment control should be treated with extreme caution. If a file under commitment control is maintained, the utility will issue commit operations after each insert, update and delete. It is the responsibility of the user to ensure that the file is journalled and that the job has had commitment control started.

Some of the 'non-mainstream' situations that the file maintenance utility will not handle, requiring the construction of purpose built RDML functions, include the following:

- Files that have no primary key. The FMU insists that the physical file underlying the file being processed has a primary/unique key. This is not true of RDML functions.
- Multi-member files. The FMU will only work with the first member in any file. This is not true of RDML functions.
- Alteration of file being used by library list. The FMU will only work on the file selected, in the library selected, as it is invoked. This is not true of RDML functions which will work on the first occurrence of the file found in the library list.
- Internally described files. Again, RDML functions can be constructed to handle internally described files.
- Files that do not have an I/O module. Since version 4.0 it has been possible to use files without having an I/O module. Such files cannot be manipulated by the FMU.
- Files in production systems. The FMU is provided as a tool to aid software developers. As such, it is not a strategic part of LANSA, and should not be used in production systems in any way.
- Files with fields that have keyboard shifts J or E and are longer than 60

bytes. This is because FMU cannot guarantee the integrity of such fields. This impacts users of DBCS languages only.

5.11.2 Steps to Invoke the File Maintenance Utility

The steps involved in invoking the file maintenance utility for a particular file are as follows:



5.11.3 Select the Physical File

When the option to use the file maintenance utility has been selected from the housekeeping menu a screen similar to the example following will be presented:

SELFILE Select File to Work with Enter full or partial name of the file definition(s) to be worked with or leave blank to select from all file definitions . _____ Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen requests the name of the PHYSICAL FILE to be maintained. This can be done in 3 different ways:

- **In full.** Enter the full name of the file required.
- **Partially.** Enter as much as desired of the partial name of the file. A search is made for all files which generically match the name specified and the resulting list is displayed.
- **Leave blank.** In this case a list of all files in the LANSA system is displayed from which the desired file(s) can be selected.

When a list of files is displayed a screen similar to the following example will be presented (which resulted from entering partial file name DEM):

SELFILE Select File to Work with

Enter full or partial name of the file definition(s) to be worked with or leave blank to select from all file definitions . . <u>DEM</u>

Sel File Library Description

DEMDIVNDC@DEMOLIBTable of valid company divisionsDEMHEADDC@DEMOLIBDemo Order Header FileDEMLINEDC@DEMOLIBDemo Order Lines FileDEMNAMEDC@DEMOLIBDemo Names FileDEMPRODDC@DEMOLIBProduct master file

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Using the Select File to Work with display it is possible to:

- Alter the file name at the top right of the screen. The change can be to specify a full file name, a partial file name or a change to blanks. In all cases another search is made for the file required.
- **Select** the desired file from the list by entering any non-blank character beside the file in the column labeled 'Sel'.

5.11.4 Select the Maintenance Access Path

Once the physical file that is maintained has been selected, a screen similar to the following example will be displayed:

DC@P201101 Select Maintenance File to use
File : DEMNAME DC@DEMOLIB Demo Names File
From list choose view of this file that is to be used for maintenance
Sel File Description / Access key information
DEMNAME Customer master file
Access by : DEMNAC Name code

DEMNAMEN View of customer master by customer name Access by : DEMNAM Name

DEMNAMEP View of customer master by postcode Access by : DEMPCD Postcode

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen displays the details of the physical file selected and all available logical views. For each file the name, library, description and key information is shown.

In the example screen shown the physical file is called DEMNAME and is keyed by 'Name code'. Additional logical views DEMNAMEN (keyed by 'Name') and DEMNAMEP (keyed by 'Post code') are also available and either could be selected as an alternative access path to DEMNAME.

Select either the physical file or one of the associated logical views. This step is described as 'selecting the maintenance access path' because you are effectively choosing the 'access path' through which maintenance is to be performed.

Note that the selection of the most suitable logical view (or access path) is only important when you wish to do direct record inquires or browse the file because the key(s) of the logical file influences:

- The key(s) to be specified for retrieval of record(s) from the file.
- The sequence in which browse information is displayed.

5.11.5 Steps to Use the File Maintenance Utility

Once the physical file to be maintained has been nominated and the desired maintenance access path has been selected it is possible to use the file maintenance facility.

The steps involved in using the file maintenance utility can be visualized like this:



|DISPLAY/| | | SELECT |---| |BROWSE | | |RECORDS | | | DISPLAY/ | | | MAINTAIN |---A RECORD |

5.11.6 Select the Type of File Maintenance Required

When the file maintenance facility is being used and the physical file and required access path have been nominated as described in 5.11.4 Select the Maintenance Access Path a screen similar to the following example will be displayed:

DC@P201102 Select Type of Maintenance

File : DEMNAMEN DC@DEMOLIB View of customer master by customer

Use cursor to select type of file access to be used

Display, change or delete individual file record(s) Browse the file to choose required record(s) Add new record(s) to the file

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

From this screen you can select the type of file access (maintenance) that you wish to use. The 3 available types are:

- Display and maintain individual file records. If this option is chosen a full key must be provided to select the record that is to be displayed. Once the record has been displayed it can be changed or deleted by using the appropriate function key.
- Browse the file and select record(s) to be maintained. If this option is chosen a partial or generic key must be specified to nominate the file record(s) to be displayed. Once the browse records are displayed it is then possible to choose individual records that are to be changed or deleted.
- Add new records to the file. Using this option allows new records to be added (inserted) into the file. Full validation checking is performed.

To select the type of file maintenance required, position the screen cursor on the same line as the description and press the enter key. This type of menu selection method is identical to that used on all LANSA menus.

Note: Only the first 100 fields of a file are displayed to perform operations such as add, update, and browse. This is a limitation of the File Maintenance Utility.

5.11.7 Add New Records to the File

When you have elected to add new record(s) to a file a screen similar in layout (but not content) to the following example will be presented.

This example is for a name and address file:

DC@P201103 Add File Record

File : DEMNAMEN DC@DEMOLIB View of customer master by customer

*	. 1 2 3 4 5
Name code	
Name	
Address line 1	
Address line 2	
Address line 3	
Post code _	

Enter Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The left hand column of the screen indicates the names of the fields in the file. You enter the value that the field is to have in the new record into the associated entry in the right hand column.

Values can be entered in totally free format. LANSA automatically checks that the value you enter does not exceed the length or decimal precision of the associated field. In addition all associated file and dictionary validation checks will be performed.

A grid is provided across the top of the input column to aid in the entry of long alphanumeric fields.

If the file record contains more fields than will fit on one screen use the ROLL keys to roll the screen forward to input subsequent fields. Input all fields before

pressing the enter key.

Once the record has been validated and added to the file the screen will be cleared and re-presented for the next record to be added. If there are no more records to add use the MENU or EXIT function keys to terminate the addition process.

5.11.8 Browse a File - Select the Browse Fields to be Displayed

If you have elected to browse the file to choose the required records a screen similar in layout (but not content) to the following example will be displayed. This example is for a name and address file as can be seen by the fields that are in the file.

DC@P201104 Select Fields to be Displayed

File : DEMNAMEN DC@DEMOLIB View of customer master by customer

Select / order fields to be displayed while browsing the file

		1 0		•
Sel	Field De	escription	Type Le	eng Dec
1	DEMNAC	Name Code	А	6
2	DEMNAM	Name	А	25
	DEMAD1	Address Line 1	А	25
	DEMAD2	Address Line 2	А	25
	DEMAD3	Address Line 3	А	25
3	DEMPCD	Post Code	А	6

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen is requesting that you select the fields that are to be displayed while browsing the file. Select and order the fields by entering a number beside them. The presence of the number selects the field. The relative order of the number sequences the fields (from left to right) across the browse display.

If you select more fields than can fit on one browse line an error message will result. In this situation reduce the number of fields chosen until they will fit on one browse line. Note that it is not necessary for the entire field to fit onto the browse line, only that at least part of the field will fit onto the browse line.

Note that the key(s) of the file being browsed do not have to be included into the browse display.

In this example it can be seen that the user has elected to browse on fields

DEMNAC, DEMNAM and DEMPCD and to display them in the same order.

5.11.9 Specify the Key(s) to Be Used

When using the file maintenance utility for individual record maintenance or browse/select file maintenance you will always be asked to specify the key(s) of the record(s) you wish to display or browse.

In such cases a screen similar in layout (but not content) to the following example will be presented.

This example is for a name and address that is being accessed via a view keyed by a field called 'Name'.

DC@P201105 Specify First Record to Review

File : DEMNAMEN DC@DEMOLIB View of customer master by customer

Specify all or part of the key of the first file record to be reviewed

Name

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This screen is requesting that you specify the key(s) of the record(s) you wish to review. The key values you enter can be input in a totally free format manner. LANSA will automatically adjust them to the required format.

When you are inquiring against a specific file record specify the full key of the record required.

When you are browsing a file specify at least one of the keys requested. In addition, if the first key is alphanumeric it can be entered as a 'generic' value. This will cause a 'generic' search to occur. For instance entering customer name 'ACME' (as in this example) will result in a list of all customers whose names start with ACME being displayed.

5.11.10 Browse a File - Select a Specific Record to Maintain

Once a full, partial or generic key has been supplied for a browse request a search is made for all matching records in the file. If one or more records are found a screen similar in layout (but not content) to the following example will be displayed.

This example is the result of a generic search on a name and address file using name 'ACME'. The user also indicated that only the name code, the name and post code were to be displayed on the browse screen:

DC@P201106 Browse File / Select Record(s)

File : DEMNAMEN DC@DEMOLIB View of customer master by customer

NamePostSel CodeNameCode_A00010 ACME AND FRIENDS2356_A00008 ACME ENGINEERING2234XA00001 ACME ENGINEERING COMPANY2014_A00009 ACME FRIENDS3003_A01000 ACME SHIPPERS & ENGINEERS2016

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

If more records were found that can be displayed on one screen use the ROLL keys to roll backwards and forwards through the list.

To select a record for full display, change or delete, enter any non-blank character beside it in the column headed 'Sel'. In this example it can be seen that the user has selected the 'ACME ENGINEERING COMPANY' record for full display.

5.11.11 Display, Change or Delete a Specific File Record

Whenever the file maintenance utility is required to display a file record in full a screen similar in layout (but not content) to the following example will result.

A full record display may occur as the result of direct record inquiry where the user enters the key(s) of the record to be displayed or as the result of the record being displayed and then selected from a browse list. In either case the type of display is identical.

This example is the full record of a record from a name and address file:

DC@P201107 Display File Record File : DEMNAME DC@DEMOLIB Demo Names File *.....1.....2.....3.....4.....5... Name code A00001 Name ACME ENGINEERING COMPANY Address line 1 101 AUSTRALIA STREET Address line 2 NEWTOWN Address line 3 NSW Post code 2042

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

If the file record contains more fields than can be displayed on one screen use the ROLL keys to roll backwards or forwards through the fields.

To change (update) a record use the CHANGE function key to place the screen in change mode. Make the desired changes and press enter. Changes made can be input in a totally free format manner. LANSA will automatically check that the changed value matches the length and decimal precision requirements of the associated field. In addition any dictionary or file level validation rules that are to be performed during update operations must be satisfied.
To delete a record use the DELETE function key. The delete is not immediate. You must confirm the record is to be deleted by entering 'YES' into a confirmation field that appears at the top of the screen. In addition any dictionary or file level validation rules that are to be performed during delete operations must be satisfied.

5.12 Exporting and Importing

The export and import facilities in LANSA are provided to enable the definition of fields, files, processes, variables, lists or components to be moved from one LANSA partition to another LANSA partition. To do this, they are 'exported' from one LANSA partition and then 'imported' into another LANSA partition.

Objects that are not part of the LANSA system may also be exported and imported via these facilities.

The LANSA partitions involved may be resident in the same physical computer (within the same LANSA system or different LANSA systems) or be resident on different computers.

An example of exporting and importing on the same physical computer would be the transfer of fields, files and processes from a development partition (or system) into a production partition (or system).

An example of exporting and importing on different computers would be copying a complete application system written using LANSA that currently runs on an IBM i in one city into a new IBM i in another city.

The important thing about the 'import' and 'export' facilities is that they allow LANSA applications to operate effectively in multiple environments on the same computer or on multiple computers in the same environment.

The export facility uses a LANSA object called an 'export list'.

To export information, you first prepare a 'list of the things to be exported and then request that LANSA exports everything on the list onto a magnetic tape or into a save file on disk.

Every 'export list' you create has a name and remains in the LANSA system until you delete it. They can be altered at any time.

As an example of how the export and import facilities are used, consider a user who has to ship a small order entry system from a head office IBM i into a new IBM i in a branch office. The steps involved would be something like this:

• On the head office IBM i prepare a list of all the 'things' that the order entry system will require to run successfully on the branch office IBM i. This is done using the list manipulation facilities provided by LANSA:

List: DIST245 |



- Request that LANSA exports all the 'things' on the list out onto a magnetic tape or into a save file. Note that this process duplicates the objects, leaving the originals unchanged.
- Send the tape or disk to the branch office, or send the save file to the branch office via a communications link.
- Sign on to the IBM i system at the branch office and then use the LANSA import facility to import the order entry system from the tape or save file.
- Use the order entry system on the branch office IBM i.

5.12.1 Control the Export and Import of Visual LANSA Components

You can control the exporting and importing of component information using the system flags *NOLADCOMP and *NOL4WCOMP.

Use of these flags should be considered carefully to avoid loss of component details.

The ***NOLADCOMP** flag allows you to controls the exporting and importing of component information between LANSA for the IBM i systems, giving you the option of exporting component information and then only importing it to selected systems.

- If you insert the flag in the system from which you are doing the export, no component information is exported.
- If you insert the flag in the system to which you are importing data, no component information is imported.

The ***NOL4WCOMP** flag allows you to stop the exporting or checking out of component information from a LANSA for i to a Visual LANSA system. This flag also stops component information from being checked into a LANSA for i Series system from a Visual LANSA system.

Frameworks and groups are always exported and imported with the component details. Messages will be printed in the export and import logs when component details exist but are not included.

If no component details are imported, the existing component definitions in the target system are not changed.

The *NOLADCOMP and *NOL4WCOMP flags can be set using the *Include Visual LANSA components in IBM i export/import* and *Include Web components in IBM i export/import* options in Work with Export and Import Settings in the *Review System Settings* facility.

5.12.2 Control the Export and Import of LANSA Web Components

To control the export and import of LANSA for the web components, two system flags *NOWEBEXP and *NOWEBIMP can be set.

- If *NOWEBEXP is set, no web details will be exported.
- If *NOWEBIMP is set no web details will be imported.

The *NOWEBEXP and *NOWEBIMP flags can be set using the *Include Web details in export* and *Include Web details in export* options in Work with Export and Import Settings in the *Review System Settings* facility.

5.12.3 Work with Export Lists

The steps involved in working with export lists can be visualized like this:



Work with Lists of Objects to Be Exported

When the option to 'Work with lists of objects to be exported' is selected from the Housekeeping Menu, the first screen presented looks something like this:

```
DC@P620001
                    Work with Export Lists
Type options and press enter .....
1=Create
            3=Copy
                                   5=Review 6=Print
                       4=Delete
            8=Merge To 9=Merge From 17=Last Action Details
7=Export
                                 Target
            Description
                                       System Type
Opt List
                                                           AS/400
    DIS006
              Distribute basic order entry details
                                                 AS/400
    AMD387
                Install order entry amendments
                                                  AS/400
                Fix for problem number 3145 12/12/88
    AMD388
                                                      AS/400
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages F16=Settings
```

Working from this panel you can, by entering one of the following options beside a list entry, elect to:

1=Create: creates a new list. This option is only valid on the first input line displayed. The following is a valid example of creating a new list:



The list name specified must be a valid IBM i name as per the rules described in Valid Names on the IBM i in *Common Input Options*. A brief description of the list must also be specified.

The target system type is important and should be correctly specified. To export to a shared folder, specify PC as the Target System Type.

Note: When objects are exported into a shared folder, the shared folder is created with the same name as the Export List.

When the Create option is used, if the new list details are all valid, the list will be created. A menu headed 'WORK WITH EXPORT LIST' is then displayed. By working from this menu it is possible to define what is to be included in the new list. This menu is described in more detail later.

3=Copy: use this option only in conjunction with the create option. To create a new list by copying an existing list, enter the create details on the first input line, then before pressing enter, specify option 3 (to copy) beside the existing list, finally press the enter key. Only entries that are valid on the target system type of the list being created will be copied. For example if the target system type of the new list is PC and the copied list contains web components, the web components will not be copied to the new list because web components can not be exported to PC.

4=Delete: deletes an existing list. Use this option against any existing list. A screen requesting confirmation of the request will be displayed before the delete is performed.

5=Review: indicates that a list is to be reviewed and possibly altered. Use this option only against an existing list. When the REVIEW option is used a menu headed 'WORK WITH EXPORT LIST' will be displayed.

6=Print: prints details of the export list. A batch job will be submitted to print the details of the list.

7=Export: indicates that you wish to export all objects nominated in the list onto a magnetic tape or into a save file. Enter this option against any existing list shown on the display. An additional screen prompt will appear asking for details of where you wish to export to. This prompt is described in detail later.

8=Merge To: use this option only in conjunction with the Merge From option.

9=Merge From: use this option only in conjunction with the Merge To option. To merge export lists, enter 8 against one list, 9 against another list and press

enter. Enter option 8 against the target list and enter option 9 against the list whose contents will be added to the target list.

Only entries that are valid on the system type of the target list will be copied. For example, if the system type of the Merge To list is PC and the Merge From list contains AS/400 other objects, the AS/400 other objects will not be added to the target list.

Matching of objects between the lists is done by type and name of object. If an object exists in both lists the entry in the Merge To list will remain unchanged. For example, if a file exists in both the Merge To and Merge From lists but the attributes (target library, compile form, copy data) differ, the values for these attributes will be left unchanged in the Merge To list.

Matching of substitution variables between the lists is done by Substitution Variable Name. If a variable exists in both lists, the value in the Merge To list is left unchanged.

All IBM i commands to be executed before/after import will be added to the target list. If a duplicate Before/After and sequence number is found, the command from the Merge From list will be added with a new sequence number. A warning message will be issued.

17=Last Action Details: indicates that you wish to review the last action details recorded for the export list. Enter this option against any existing list shown on the display. A screen will be displayed showing the last action details. This screen is described in Online Access to Last Action Details.

F16=Settings: displays the system setting relevant to the export and import. For details, refer to Review System Settings. You will only be able to change these settings if you have access to the Administration Menu - Review System Settings.

By working from this panel it is possible to review and alter the things that are included on the export list. This menu is described in further detail later.

Note that no security exists over export lists. Any user authorized to use this facility can review and alter any export list.

Submit a PC Export Run to Batch

When option 7 to Export has been entered beside an existing list that has a Target System Type of PC, this screen is displayed.

Check Objects Out to a PC Platform DC@P620015 List Specify WIN Usage as R or M, also for M, the Check Out Task & PC Name Default: Usage/Task PC Name Usage/Task PC Name Object Description Type AUTHOR FIELD AUTHOR FIELD **##GENERIC NAME##** DEM ____ Department code table _ ____ DEPTAB FILE PSL FILE **##GENERIC NAME##** ORDERS PROCESS Order Processes _ ____ PSL PROCESS ##GENERIC NAME## **RECEIVE FUNCTION Receive Orders** *MTXTDEM MLTVAR ##GENERIC NAME## ____ FRPRT05 TEMPLATE Drill Down Summary TEMPLATE ##GENERIC NAME## LD Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The Check Objects Out to a PC Platform screen requests use, task and PC name information about the objects to be exported.

Default: Usage/Task/PC Name

The Usage, task and PC Name that will apply to objects that are unspecified.

Usage/Task/PC Name

The Usage may be R = Read-Only or M = Modify for each object to be exported. A valid Task ID and PC Name must be specified for Usage Modify, but is not required for Usage Read-Only.

Press Enter when complete and the Submit Export to Batch screen is displayed.

DC@P620002	Submit Export to Batch
List : XXXXXX	x xxxxxxxxxxxxxxxxxxxxxxxxxxx
Submit this job to ba Using Job name Job description . Job queue Output queue	ttchYES

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This export run confirmation screen requests details of the actual export job which will run in batch. The details required to successfully run the export job are found in:

Submit This Job

Using Job Name

Job Description

Job Queue

Output Queue

If you are transfering objects between multilingual machines, please refer to Handling Multilingual Text in the LANSA Multilingual Application Design *Guide*.

Submit a IBM i Export Run to Batch

If a Target System Type of IBM i is specified on the Work with Export Lists screen, this Submit Export to Batch screen is displayed.

DC@P620002 Submit Export to Batch List Submit this job to batch <u>YES</u> Using Job name LANSAEXP Job description *LIBL/QBATCH Job queue *JOBD Output queue*LIBL/QPRINT Export into tape or diskette device named using file sequence number Or Export into save file named Omit RDML code from exported data NO Omit Frameworks and Groups from exported date . / NO Include Documentor Details NO Select languages to be exported or *ALL *ALL Sel Language Description DEU German ENG English +

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This export run confirmation screen requests details of the actual export job which will run in batch. The details required to successfully run the export job are:

Submit This Job to Batch Using Job Name Job Description Job Queue Output Queue

Export into Tape or Disk Device Named

When you want to export objects defined in the list out to magnetic tape or disk, specify the name of the actual disk or tape device. For instance, names like DKT01 and TAP01 are commonly used.

Using File Sequence Number

Enter the tape or disk device name to specify the file sequence number to be used for the export of objects to tape or disk. The default sequence number is *END.

Or Export Into Save File Named

Use this field if you want to export the objects defined in the list into a save file. A save file is an object created and maintained by the operating system and it can be treated as if it were a magnetic tape that is actually stored on disk. Refer to the operating system command CRTSAVF (create save file) in the IBM supplied manual for *details* about creating and manipulating save files. Use a save file in the following situations:

- When you are exporting/importing between LANSA systems that are resident on the same CPU. This will save on the logistics of using tapes and disks, and will actually perform better than using those devices.
- When you wish to transfer the exported objects via a communications link to another machine. The operating system contains a command called SNDNETF (send network file) that will transfer a save file down a communications link to another machine. Refer to this command in the IBM supplied manuals for *details* of how save files can be transferred via a communications link.

Omit RDML Code From Exported Data

Note: Use this option with extreme caution.

Use this field to indicate if the RDML or RDMLX source code for all functions in this export list is to be included or omitted from exported data. The default is NO.

If NO or N is entered, the RDML & RDMLX source code will be included in

the exported data. In this case, the source code will therefore be available after this export list is later imported into a LANSA partition.

If YES or Y is entered, the source code will be omitted from the exported data. In this case, the RDML & RDMLX source code for all functions in the export list will not be included in the exported data, and will therefore not be available after this export list is later imported into a LANSA partition.

If you export a set of functions with their source code omitted, and then (re)import them again, you will have effectively and completely deleted their source code from the partition.

Omit Frameworks and Groups from exported data

Use this field to indicate if Frameworks and Groups for all objects in this import list are to be included or omitted from imported data.

The default is NO.

If NO is entered, the Frameworks and Groups will be included in the exported data. In this case, the Frameworks and Groups will be available after this export list is imported into a LANSA partition.

If YES is entered, the Frameworks and Groups will be omitted from the exported data. In this case, the Frameworks and Groups for all objects in the export list will not be included in the exported data, and will therefore not be available after this export list is imported into a LANSA partition.

If you export a set of objects with their Frameworks and Groups omitted, it will leave any old Frameworks or Groups in the importing partition.

Include LANSA/Document Details

Use this field to indicate if LANSA/Document details for functions in this export list are to be included in the exported data. The default is NO.

If YES is not entered, or anything other than YES is entered, the LANSA/Document details will be omitted from the exported data.

If YES is entered, the LANSA/Document details in the export list will be included in the exported data, and will therefore be available after this export list is later imported into a LANSA partition.

Select Languages to Be Exported or *ALL

This option will only appear if the current partition is a multilingual partition. Use this field to indicate which of the partition languages should be included in the exported data. The default is *ALL.

If *ALL is entered, then all of the partition languages (which appear in the list

below this option) will be included in the exported data.

If this field is left blank, you must select the required partition languages to be included in the exported data from the list of partition languages.

Work With Objects in a List

When a new list is created or an existing list is reviewed a screen like the following is presented:

DC@P620003 Work with Export List List Enter number of function or place cursor on same line and press Enter. 1. Add fields to list 2. Add files to list 3. Add processes/functions to list 4. Add system variables to list 5. Add multilingual variables to list 6. Add application templates to list 7. Add all partition objects to list 8. Add other/non-LANSA objects to list 9. Add all objects worked on under a task identifier(s) to list 10. Add export lists to list 11. Review/delete objects already in list 12. Define substitution variables used in list 13. Define IBM i commands to execute before/after import 14. Add Web HTML components to list More...

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Save

Page down to see the additional menu options:

DC@P620003 Work with Export List

Enter number of function or place cursor on same line and press Enter.

____ 15. Add Web XML components to list
16. Add Technology services to list
17. Add Weblets to list
18. Add Visual LANSA Components to list

Bottom

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Msgs Fnn=Save

By working from this screen the objects that are defined in the new or existing export list can be manipulated. In summary, the facilities provided to manipulate the list are as follows:

- Add new LANSA fields to the list.
- Add new LANSA files to the list.
- Add new LANSA process and/or functions to the list.
- Add all objects (LANSA fields, files and processes/functions) defined in the current partition to the list.
- Add LANSA system variables to the list.
- Add LANSA multilingual variables to the list.
- Add non-LANSA objects to the list. This is available for lists with a target system type of AS/400. This means any IBM i object that is not directly controlled by LANSA.
- Add message files to the PC Export list. This is available for lists with a target system of PC.

- Add all objects (LANSA fields, files, processes/functions and application templates) worked on under a task identifier(s) for the current partition to the list.
- Add LANSA export lists to the list.
- Review all objects currently in the list and delete those no longer required.
- Define substitution variables to be used during an import run.
- Define IBM i commands to be executed before or after an import run.
- Add Web HTML components to the list.
- Add Web XML components to the list.
- Add Technology services to the list.
- Add Weblets to the list.
- Add Visual LANSA Components to list

When reviewing an old formatted export list, this message will be displayed

• "Please check. This list has been converted to show fields and components separately."

For such lists, the contents should be checked and then saved in the new format.

Save New or Changed List Details

All alterations to an export list are performed in a work area and do not become permanent until the revised export list is saved.

An export list can be saved by either:

- using the function key 'Save' available on the Work with Export List display. When this function key is used, the list is saved and the Work with Export List menu re-displayed with a completion message. Further changes to the list may then be made.
- exiting from the Work with Export List Menu. Using the Exit/System/Resume or Cancel function key will cause an exit from the menu. In any of these cases the export list is saved.

Generic Object Names

The ability to add generic object names to the export list is accessible from all the 'Add ... to list' facilities except for; 'Add system variables to list' and 'Add other/non-LANSA objects to list'.

The Generic function key is used to add the Partial Name entered at the topright corner of the screen, into the export list. If the Partial Name entered is blank the Generic function is ignored - to add all objects for the selected type, enter '*ALL', then press Generic or enter.

Any manual or generic object selections already made that are included by this new generic selection are removed. If an attempt is made to add a generic name that is included by a generic name previously added (i.e. add generic PSL when generic P has already been added) an error message is displayed.

Object names that are selected as the result of a generic name are shown with an 'G' beside them on their display list.

The 'G' cannot be removed. Thus a generic selection cannot be removed from the export list by using this facility. Use the option 'Review/delete objects already in list' to do this. Generic selections are identified by their description which is '##GENERIC NAME##'. This option is described in detail in a later section.

Generic name selection has the advantage that the actual objects exported is determined at the time of export, not when the list is created. This means that any objects that were in a generic selection, but since deleted are ignored, and any new objects are considered when exporting.

Add Fields to the List

When the option to *Add fields to list* is chosen from the *Work with Export List Menu*, the Add *Field(s) to List* screen displayed to allow you to manipulate the export list.

DC@P620004 Add Field(s) to List

Enter full or partial name of the data dictionary field(s) to be worked with or leave blank to select from all fields in dictionary or enter *ALL to select all fields in the data dictionary . . R

Field	ł	Export Export Exp	ort Export
Sel Name	Description	Files Fun	WebCmp XMLCmp
REP1PA0	GE Report numbe	r 1 page number	
X RESCO	DE Response coo	le (Y or N) _	
G RETCA	L Return Code f	rom Calender	
G RETUR	N Return Code		

Fnn=Help Fnn=Exit Fnn=Generic Fnn=Cancel Fnn=Messages F22=ShowL

This facility allows you to build temporary lists of fields from the LANSA data dictionary and display them on the screen. The list will show standard fields from the repository, including fields with Visual LANSA visualization. It will not show Visual LANSA components. Use the Add Visual LANSA Components to the list to view and select Visual LANSA components. From the displayed lists fields can be chosen for inclusion in your export list.

These lists can automatically be built from

- all fields
- only fields that have a certain generic name, or

• all fields in the dictionary.

A displayed list may not fit on one screen. This is indicated by a '+' sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

Long Names

You can show the long name for a field in the list by placing the cursor on it and pressing **F22**. Refer to Long Names for more details.

The Sel column

When you want to choose a field from the displayed list for inclusion into the export list enter a 'Y' beside it in the column headed 'Sel'.

Alternatively you may use the Generic function key to add the generic name into the export list. Generic names are described in detail in Generic Object Names.

X or G

Fields shown in the displayed list which are already included in the export list are shown with an 'X' or 'G' beside them. The 'X' or 'G' cannot be removed. Thus a field cannot be removed from the export list by using this facility. Use the Review/delete objects already in list to do this.

Export Files

Beside a chosen field, enter a Y in this column beside a field to automatically include in the export list all file definitions that contain a chosen field. To do this in the column headed 'Export Files'.

This column is ignored when using the Generic function.

Export Fun

Beside a chosen field, enter Y to automatically include in the export list all functions that use a chosen field in some way. Note that only functions that are currently compiled will be correctly selected when this option is used.

This column is ignored when using the Generic function.

Export WebCmp

Beside a chosen field, enter a 'Y' to automatically include Web HTML visual components that have been defined for a chosen field in the export list. Note that only visual Web HTML components with the same name as the field will be selected and added to the list.

If system flag ***NOWEBEXP** is set in the system data area DC@OSVEROP or

the target system is not an IBM i, the *Export WebCmp* column is not available. When you have completed using this facility, use the *Cancel* function key to return to the *Work with Export List Menu*.

The *Export WebCmp* column is ignored when using the Generic function. You can set the ***NOWEBEXP** flag via *Include Web details in export* option in *Export and Import settings*.

Export XMLCmp

Beside a chosen field, enter 'Y' in this column to automatically include Web XML visual components which have been defined for a chosen field in the export list. Note that only visual Web XML components with the same name as the field will be selected and added to the list.

If system flag ***NOXMLEXP** is set in the system data area DC@OSVEROP, or the target system is not an IBM i, the *Export XMLCmp* column is not available.

When you have completed using this facility, use the *Cancel* function key to return to the *Work with Export List Menu*.

Add Files to the List

When the option to 'Add files to list' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620005 Add H	File(s) to List
List : XXXXXXXXX	xxxxxxxxxxxxxxxxxxxxxxx
Enter full or partial name o With or leave blank to selec	f the file definition(s) to be worked ct from all file definitions <u>P</u>
File Sel Name Library D _ PSLMST DC@DEM _ PSLSKL DC@DEM	Comp Data Export Export escription Form Also Fields Function 4OLIB Personnel master 1OLIB Personnel skills
Fnn=Help Fnn=Exit Fnn=	Generic Fnn=Cancel Fnn=Messages F22=Show

This facility allows you to build temporary lists of file definitions known to LANSA and display them on the screen. From these displayed lists file definitions can be chosen for inclusion into your export list.

Note the use of the words file definition. Only the name of the file definition, which is the same as the associated physical file name, is ever included into the export list. When a file definition is exported, all associated details such as logical views, virtual fields, etc are automatically included.

These lists can be built from all files or only files that have a certain generic name.

A displayed list may not fit onto one screen. This is indicated by a '+' sign in the lower right of the screen. In such cases use the roll up and roll down keys to

scroll backwards and forwards through the displayed list.

Files shown in the displayed list which are already included in the export list are shown with an 'X' or 'G' beside them. The 'X' or 'G' cannot be removed. Thus a file cannot be removed from the export list by using this facility. Use the option 'Review/delete objects already in list' to do this. This option is described in detail in a later section.

Long Names

You can show the long name for a file in the list by placing the cursor on it and pressing **F22**. Refer to Long Names for more details.

The Sel column

Enter 'Y' beside a file displayed in the list to include it in the export list.

Enter 'L' if you want specific logical views to be included for the file. This will display a list of logical views that you can select from.

Enter 'O' if you want to include the I/O or OAM Modules for the file.

For further information on the 'L' and 'O' selectors, refer to Add Logical Views to the list or Add I/O or OAM Modules to the list.

Alternatively you may use the Generic function key to add the generic name into the export list. Generic names are described in detail in Generic Object Names.

Note: The 'Comp Form' and 'Data Also' columns are used when performing an IBM i export from LANSA. They are ignored when performing a PC export.

The 'Comp Form' column

Beside a chosen file, enter a 'Y' to indicate that the file definition is to be exported in 'compiled' or 'ready to use' form. If this option is used, the file will be usable as soon as it has been imported into the target system. If this option is not used, the file definition will have to be 'made operational' on the target system after it has been imported.

If L or O is already displayed, you can't change them.

'L' indicates that individual logical views have been selected for the file.

'O' indicates that an OAM or/and I/O module has been selected.

The 'Data Also' column

Beside a chosen file, enter 'Y' in this column to indicate that the data contained

in the file should be exported as well. If this option is used, the data in the file will replace any data already in the file on a target machine. If this option is not used, data already in the file on a target machine will be left intact (after undergoing any necessary conversion to match the new file layout).

When 'Data Also' = Y then 'Comp Form' should also = 'Y' to make the file definition operational.

When using the Generic function, the export options 'Compiled Form' and 'Data Also' are selected through a pop-up. The decisions made are applied to all files that are selected by this generic name.

DC@P620005 Add File(s) to List List : XXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX
: :
File : Use cursor to select export options :ort Export
Sel Name Li: Compiled Form :s Function
G TEMP L2 : Data Also :
G TEST L2: :
G TESTBAT L2 : Then use cursor to initiate action :
G TESTBC L2 : Continue with export options :
X TSTBC3 L2: Cancel generic add :
_ TSTFL2M L2: : N
: F1=Help F14=Messages Fnn=Cancel :
:
Fnn=Help Fnn=Exit Fnn=Generic Fnn=Cancel Fnn=Messages

An option is provided to automatically include in the export list all fields that are defined in the chosen file. To do this enter a 'Y' in the column headed 'Export Fields' beside the chosen file.

The 'Export Fields' column is ignored when using the Generic function.

Additionally, an option is provided to automatically include in the export list all functions that use a chosen file in some way. To do this enter a 'Y' in the column headed 'Export Functions' beside the chosen file. Note that only functions that

are currently compiled will be correctly selected when this option is used. If the Include all functions and Include all fields is requested and the function is in a Web enabled process, then web visual components for fields used in the function will also be added to the export list, provided the *NOWEBEXP flag is not set and the target system is an IBM i.

The 'Export Functions' column is ignored when using the Generic function.

Note: Two templates are available to export data from a LANSA file on IBM i to a LANSA file on PC: FRUTIL01 (up to 256 chars) and FRUTIL02 (greater than 256 chars).

When you have completed using this facility, use the Cancel function key to return to the Work with Export List Menu.

Add Logical Views to the list

Warning: Careful thought should be given to the consequences of using this feature. The purpose is to allow a new logical view to be created and exported/imported into another system without having to export/import the physical file and existing logical views.

Note: The compiled logical view and I/O module are imported into the target system. The I/O module is imported automatically, as the existing one will not recognize the new logical view. When the file is made operational on the exporting system the only difference between its definition and that on the importing system should be the new logical view, which is to be exported/imported. If there are other differences, the I/O module may not function as expected on the target system.

When a file is selected with an 'L' in the *Sel* column of the *Add files to list* screen, a screen similar to the following example is used to manipulate the export list:

DC@P620016 Add Logical View(s) to List

Sel	Logical View	Description
Х	PSLMST1	Personnel master 1
	PSLMST2	Personnel master 2
_	PSLMST3	Personnel master 3

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages F22=ShowLong

This facility allows you to build temporary lists of logical views known to LANSA and display them on the screen. From these displayed lists, logical views can be chosen for inclusion in your export list.

When individual logical views belonging to a file are selected, the export will contain the same information as if the file itself had been selected with 'Comp Form' 'Y' and 'Copy Data' 'N'. The difference occurs when the import is done. If the version of LANSA supports the import of selected logical views, then all the associated file definitions (views, virtual fields etc), but only the selected logical view objects and the I/O module will be imported. But if the import of selected logical views is not supported by the LANSA system then all definitions and all objects will be imported.

These lists will contain all the logical views for the file.

A displayed list may not fit on one screen. This is indicated by a '+' sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

Files shown in the displayed list which are already included in the export list are shown with an 'X'. The 'X' cannot be removed. Thus a logical view cannot be removed from the export list by using this facility. Use the option 'Review/delete objects already in list' to do this. This option is described in detail in a later section.

Additionally, you can show the long name for a logical view in the list by placing the cursor on it and pressing **F22**. Refer to Long Names for more details.

When you want to choose a logical view from the displayed list for inclusion into the export list enter any non blank character beside it in the column headed 'Sel'.

When you have completed using this facility, use the Cancel function key to return to the Work with Export List Menu.

Add I/O or OAM Modules to the list

When a file is selected with an 'O' in the *Sel* column in the *Add files to list* screen the OAM and/or I/O module will be exported.

The OAM and/or I/O module will be added to the export lists making the *Comp From* will be **O** and *Data Also* **N**. The export will not export the physical or the logical file but will export the OAM and/or I/O module with the definition of the file and the trigger program for the file if DB triggers are enabled.

To remove this module, put O in the *Sel* file of the export list screen and it will be dropped from the Export list.

Add Processes/Functions to the List (Select by Process Name)

When the option to 'Add processes/functions to list' (selection by process name) is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620006 Add Process/Function to List
List : XXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXX
Enter full or partial name of the process(s) to be worked with or leave blank to select from all processes <u>PSL</u>
Process/ Comp.Export Export
Sel Function Description Form Fields Files
PSLSYS Personnel System Main Menu Y
EMPLIST Full employee listing Y
ENROL Enrol an employee V
INOUIRE Browse/maint_employee and skill files Y
PHONE Employee business phone number list Y
SALARY Salary commitment reports Y
SEARCH General employee search Y
SNAME Telephone number search V
PSI TAB Personnel Table Maintenance Menu V
MDFPTAB Review/maintain/print department table V
MSECTAB Review/maintain/print section table V
Fnn=Help Fnn=Exit Fnn=Generic Fnn=Cancel Fnn=Messages F22=Show

This facility allows you to build temporary lists of processes (and associated functions) known to LANSA and display them on the screen. From these displayed lists, processes and/or functions can be chosen for inclusion into your export list.

These lists can either be built from all processes, or only from processes that have a certain generic name.

When any process is included in a displayed list all its associated functions are also shown for possible selection.

The process definition will be exported with its selected function/s regardless of whether it is selected or not. If a process is not selected with its associated functions it will run in interpretive mode after import.

A displayed list may not fit on one screen. This is indicated by a '+' sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

The Sel column

Processes and/or functions which are already included in the export list, are shown with an 'X', 'G' or 'F' beside them.

- An 'X' indicates the individual process or function has been selected.
- A 'G' indicates the process, and all its functions, has been selected by a generic process selection.
- An 'F' indicates the function has been selected by a generic function selection.

The 'X', 'G' or "F' cannot be removed. Thus a process or function cannot be removed from the export list using this facility. Use the option 'Review/Delete Objects Already in the List' to do this.

Enter an 'Y' beside a process or function from the displayed list to include it in the export list.

Alternatively you may use the Generic function key to add the generic name into the export list. When a process is generically selected, all of its functions are included as well. Generic names are described in detail in Generic Object Names.

Comp Form

Beside the chosen process or function, enter 'Y' in this column to indicate that the process or function is to be exported in 'compiled' or 'ready to use' form. If this option is used the process or function will be usable as soon as it has been imported into the target system.

If this option is not used the process or function definition will have to be 'compiled' on the target system after it has been imported.

When using the Generic function the decision to export the process and

function's compiled form is entered through a pop-up. The decision made is applied to all processes/functions that are selected by this generic name.

Add Process/Function to List DC@P620006 List Enter full or parti with or leave blank : Generic Process Selection : PSLU Process / : Use cursor to select export options : t Export **Compiled Form** Sel Function : : s Files Х PSLUTL CALC : Then use cursor to initiate action : N Ν CALENDR : Continue with export options Х SDEPTAB : Cancel generic add : N Ν Х SPSLMST : Х SSECTAB : F1=Help F14=Messages Fnn=Cancel Ν Х SSKLTAB Select skill from skill table Y Fnn=Help Fnn=Exit Fnn=Generic Fnn=Cancel Fnn=Messages

Export Fields

To automatically include in the export list all fields that are used by a function, enter a 'Y' in this column beside the chosen function.

If the *Include all fields* is requested and the function is in a Web enabled process, then web visual components for the fields used in the function will also be added to the export list, provided the the target system is NOT an IBM i and *NOWEBEXP flag is not set.

The 'Export Fields' column is ignored when using the Generic function.

Export Files

An option is provided to automatically include in the export list all file definitions that are used by a chosen function. To do this enter a 'Y' in this column beside the chosen function.

The 'Export Files' column is ignored when using the Generic function.

The option to automatically choose fields and files used by a function will only perform correctly if the chosen function is currently compiled.

It is strongly recommended that if a function is chosen for export that its associated process should also be chosen. Additionally, the ability to choose an individual function within a process should only be used in simple amendment situations. In all other cases select the process and all its associated functions.

Web detail export

Web details for Process and Functions will be exported provided the process is web enabled, the export target system is NOT an IBM i and system flag *NOWEBEXP is not set in the system data area DC@OSVEROP.

You can turn on or off the *NOWEBEXP flag via Include Web Details in Export in Export and Import settings.

If web details are to exported for a process and *LW3 type system variables have been created for the process (eg *LW3PBGI_pppppppp), these *LW3 system variables for the process will be added to the list. These *LW3 type system variables will not be added when using the Generic function.

If the web details are to be exported and a browse list used in the function has been customized by use of *LW3 type system variables, these *LW3 system variables for the browse list will be added to the list. These *LW3 type system variables will not be added when using the Generic function.

Show Long Names

You can show the long name for a process or function in the list by placing the cursor on it and pressing **F22**. Refer to Long Names for more details.

When you have completed using this facility, use the *Cancel* function key to return to the Work with Export List Menu.

Add Functions to the List

When the option to 'Add functions to list' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620006 Add Function to List

Enter full or partial name of the function(s) to be worked with or leave blank to select from all functions S

	Comp.Expoi	rt Export	
Sel Functn Process De	scription	Form Fields Files	
_ SALARY PSLSYS	Salary commitme	nt reports Y	
_ SEARCH PSLSYS	Employee search	Y	
_ SNAME PSLSYS	Telephone number	r search Y	
_ SSECTAB PSLUTL	Select section	Y	
_ SSELECT PSLUTL	Prompt selection	Y	
_ SSKLTAB PSLUTL	Select skill	Y	

Fnn=Help Fnn=Exit Fnn=Generic Fnn=Cancel Fnn=Messages F22=ShowLo

The Add Function to List facility allows you to build temporary lists of functions known to LANSA and display them on the screen. From these displayed lists functions can be chosen for inclusion into your export list. These lists can either be built from all functions, or only from functions that have a certain generic name.

The process definition will be exported with selected function/s.

A displayed list may not fit on one screen. This is indicated by a '+' sign in the

lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

The Sel column

Functions shown in the displayed list which are already included in the export list are shown with an 'X', G' or 'P' beside them.

- An 'X' indicates the individual function has been selected.
- A 'G' indicates the function has been selected by a generic function selection.
- A 'P' indicates the function has been selected by a generic process selection.

The 'X', 'G' or 'P' cannot be removed. Use the option 'Review/delete objects already in list' to remove a selection from the export list.

To choose a function from the displayed list for inclusion in the export list enter an 'Y' beside it in the column headed 'Sel'.

Alternatively you may use the Generic function key to add the generic name into the export list. Generic names are described in detail in Generic Object Names. Care should be taken to not include a function by a generic process selection and by a generic function selection. If this is done the function definition and optional compiled form will be included twice in the export.

The **Comp. Form**, **Export Fields** and **Export Files** columns have the same functionality as those on the *Add Processes/Functions to the List* option. Refer to Add Processes/Functions to the List (Select by Process Name) for details.

Show Long Names

You can show the long name for a function in the list by placing the cursor on it and pressing **F22**. Refer to Long Names for more details.

When you have completed your selection, use the *Cancel* function key to return to the *Work with Export List Menu*.
Add System Variables to the List

When the option to 'Add system variables to list' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620007 Add System Variable(s) to List

Enter full or partial name of the system variable(s) to be worked with or leave blank to select from all system variables . ._*

Sel	System variabl	e Description
_	*BLANK	Blank / blanks variable
_	*BLANKS	Blank / blanks variable
_	*COMPANY	Name of current company / organisation
_	*CPFREL	Current CPF release level
_	*DATE	Numeric date in installation format
_	*DATEC	Character date in installation format
_	*DAY	Current day (numeric)
_	*DAYC	Current day (character)
_	*DDMMYY	Numeric date in format DDMMYY
_	*DDMMYYC	Character date in format DDMMYY
_	*FUNCTION	Current function name
_	*JOBMODE	Current job mode (batch or interact ive)

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This facility allows you to build temporary lists of LANSA system variables and display them on the screen. From these displayed lists system variables you can choose those to include in your export list.

These lists can be built from either all system variables, or only from system variables with a specified generic name.

A displayed list may not fit on one screen. This is indicated by a '+' sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

The Sel column

System variables already included in the export list are shown with an 'X' beside them. The 'X' cannot be removed. Thus a system variable cannot be removed from the export list by using this facility. Use the option 'Review/Delete Objects Already in the List' to do this.

Enter a 'Y' beside a specific variable, to include it in the export list.

The associated system variable evaluation program is automatically included in the export list as a non-LANSA object. The source library for this object defaults to *LIBL (search current jobs library list). This value may not be correct and may have to be modified later using the 'Review/Delete Objects Already in the List' option.

When you have finished using this facility, use the *Cancel* function key to return to the *Work with Export List Menu*.

Add Multilingual Variables to the List

When the option to 'Add multilingual variables to list' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620013 Add *MTXT Variable(s) to List

Enter full or partial name of the *MTXT variable(s) to be worked with or leave blank to select from all *MTXT variables . . . _____

Sel	Multilingual Variable Value		
_	*MTXTDEMABWIN02901	Select Department	
_	*MTXTDEMABWIN05001	Select Section	
_	*MTXTDEMABWIN07301	Select Employee	
_	*MTXTDEMCALEN02801	Calendar	
_	*MTXTDEMCALEN05801	MONDAY	
_	*MTXTDEMCALEN06001	TUESDAY	
_	*MTXTDEMCALEN06201	WEDNESDAY	
_	*MTXTDEMCALEN06401	THURSDAY	
_	*MTXTDEMCALEN06601	FRIDAY	
_	*MTXTDEMCALEN06801	SATURDAY	
_	*MTXTDEMCALEN07001	SUNDAY	+

Fnn=Help Fnn=Exit Fnn=Generic Fnn=Cancel Fnn=Messages

This facility allows you to build temporary lists of LANSA multilingual variables and display them on the screen. From these displayed lists, multilingual variables can be chosen for inclusion in your export list.

Multilingual variables will only be available for selection if the current partition is a multilingual partition.

Lists can be built either from all multilingual variables, or only from

multilingual variables with a specified generic name.

A displayed list may not fit on one screen. This is indicated by a '+' sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

The Sel column

Multilingual variables shown in the displayed list, and which are already included in the export list are shown with an 'X' or 'G' beside them. The 'X' or 'G' cannot be removed. Thus a multilingual variable cannot be removed from the export list by using this facility. Use the option 'Review/Delete Objects Already in the List' to do this.

Enter 'Y' beside a multilingual variable in the list if you want to include it in the export list.

Alternatively you may use the Generic function key to add the generic name into the export list. Generic names are described in detail in Generic Object Names.

When you have finished using this facility, use the *Cancel* function key to return to the Work with Export List Menu.

Add Application Templates to the List

When the option to 'Add application templates to list' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620008 Add Template(s) to List

Enter full or partial name of the application templates to be worked with or leave blank to select from all application templates . B

Sel	Application Template	Description
_	BASEPGM	Basic Program Layout
_	BASESUB	Basic Subroutine Layout

Fnn=Help Fnn=Exit Fnn=Generic Fnn=Cancel Fnn=Messages

This facility allows you to build temporary lists of LANSA application templates and display them on the screen. From the displayed list you can choose application templates to include in your export list.

Lists can be either built from all application templates, or only application templates with a specified generic name.

A displayed list may not fit on one screen. This is indicated by a '+' sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

The SEL column

Application templates in the displayed list which are already included in the export list are shown with an 'X' or 'G' beside them. The 'X' or 'G' cannot be removed using this facility. Use the option 'Review/Delete Objects Already in

the List' to do this.

Enter 'Y' to choose an application template from the list to include it in the export list.

Alternatively you may use the Generic function key to add the generic name into the export list. Generic names are described in detail in Generic Object Names.

When you have finished using this facility, use the *Cancel* function key to return to the *Work with Export List Menu*.

Add All Partition Objects to the List

When the option to *Add all partition objects to list* is chosen from the Work with Export List Menu, a search is made of all LANSA objects defined within the current partition.

The export lists for LANSA objects are cleared and a generic '*ALL' entry is added for each object type.

Once this step is complete the export list should be reviewed by using the 'Review/delete objects already in list' option which is described in detail in Review/Delete Objects Already in the List.

During this review step, the default values used for exporting file definitions, processes and functions should be carefully checked and changed if not suitable.

This option is quite fast as only one entry is made for each object type affected, irrespective of how many objects exist in the partition. When it has completed the Work with Export List Menu will be re-displayed with a completion message.

Add Non-LANSA Objects to the List

This option is only available for export lists with a target system type of AS/400.

When the option to *Add other/non-LANSA objects to list* is selected from the *Work with Export List Menu*, a screen similar to the following example is used to manipulate the export list:

Add Other Objects to List DC@P620009 List Enter details of all other / non - LANSA objects to be added to the export list: Note : Target library may be specified as a substitution variable Source Object Target Copy data Object Library Library as well Type _ ____ _ _ _ _____ ____ Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages F20=Find

This facility allows you to add to the export list details of objects that are not directly under the control of LANSA. Such objects will be included into the export run and imported into the target system as part of the import run. An example of an object that is not under the direct control of LANSA is a message file. Message files are a native part of the IBM i operating systems and are widely used within LANSA.

Only new objects can be added to the list via this facility. Objects that are already in the list cannot be reviewed, changed or deleted via this facility. To do this use the 'Review/Delete Objects Already in the List' facility.

To specify details of other/non-LANSA objects, key in all required details and press *Enter*.

Alternatively, you may use the **Find** option. This will display a window where you can enter the criteria for a DSPOBJD command (object, library and type). Press *Enter* when you have defined your criteria and objects that match your selection will be displayed. Select the required objects by entering a value in the *Sel* column. The selected objects will be added to the *Other Objects* list. You may then enter the *Target Library* and *Copy data* values and press *Enter*.

If all required objects will not fit onto one screen use the roll up and roll down keys to scroll backwards and forwards through the list of objects.

If the object details are acceptable the screen will be re-displayed requesting that more object details are input. Accepted object details are not re-displayed and cannot be reviewed or changed via this facility. Refer to the 'Review/Delete Objects Already in the List' facility.

Object

This must be a valid name as per IBM i naming conventions (described in Valid Names on the IBM i in *Common Input Options*). Additionally the object must currently exist in the source system.

Source Library

Must be specified as either a valid library name or as special value *LIBL.

Special value *LIBL indicates that the current job's library list should be searched to locate the object. During the actual export run, it will be the batch job's library list that is used to locate the object. If this is different from the interactive job's library list the object may not be found, or worse, the wrong object may be exported.

Object Type

Should be specified as per IBM i conventions. The exception is type *FILE which must **not** be used. The exact type of file must be specified. Allowable values for object type are as follows:

Туре	Description	Туре	Description

*CHTFMT	Chart format	*SPADCT	Spelling aid dictionary
*CLS	Class	*SSND	Session description
*CMD	Command	*TBL	Translation table
*DTAARA	Data area	*PF	Physical file
*DTAQ	Data queue	*LF	Logical file
*EDTD	Edit description	*DSPF	Display file
*FCT	Forms control table	*PRTF	Printer file
*GSS	Graphics symbol set	*SRCPF	Source physical file
*JOBD	Job description	*SAVF	Save file
*JOBQ	Job queue	*SPLF	Spool file
*JRN	Journal	*DKTF	Diskette file
*JRNRCV	Journal receiver	*TAPF	Tape file
*MSGF	Message file	*BSCF	BSC communications file
*MSGQ	Message queue	*CRDF	Card file
*OUTQ	Output queue	*CMNF	Communications file
*PGM	Program	*DDMF	Distributed data mixed file
*PRTIMG	Print image	*MXDF	Mixed device file
*SBSD	Subsystem description		

Target Library

Specified as either a valid library name or as a special 'substitution value'.

If a 'substitution value' is used, the importer can actually nominate the actual library to be used as the target library during the import run. Substitution values are described in more detail in a later section of this chapter.

Special value *LIBL is not acceptable as a target library name.

Copy data as well

Enter Y in this column beside the required object. It only applies to physical files (object type *PF).

When this option is used with a non-LANSA physical file, the data in the file is exported as well. However, it is your responsibility to convert or replace data in any existing version of the file on the target system. This can usually be achieved by executing IBM i commands before and/or after the import run on the target machine. This facility is described in Review/Delete Objects Already in the List.

When you have finished using this facility, use the *Cancel* function key to return to the *Work with Export List Menu*.

Add Message Files to PC Export Lists

This option is only available for export lists with a target system type of PC. It may be used to include your application message files in a PC export.

DC@P620025 Add Message File to PC List

Enter details of the message files to be added to the export lists :

Name Library Lang Int. Name *LIBL *LIBL

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This facility allows you to include message files in the details that are to be exported to a LANSA system on a PC.

Name

The name of the message file as it is 'known' to the IBM i. The program checks that this message file exists in the library list.

The LANSA system message file, DC@M01, may not be exported using this export facility. You must use the export system definitions option within PCMAINT to move the LANSA message file DC@M01 to your PC environment.

Library

This is set to *LIBL. This indicates that the library list is used to locate the message file to check the file exists and when the export is executed to locate the message file data.

Language

The language identifier entry fields are shown only if the current partition is multilingual. The entered language must be valid for the current partition. The entered language should be the language associated with that specific message file.

Message file Internal Name

The internal name entry fields are shown only if the current partition is multilingual. The message file name as it is used in LANSA applications should be entered here. The program does NOT check the validity of this name, so care should be taken to avoid spelling errors.

Example for an export to the PC in a multilingual partition:

Name Library Lang Int. Name USERMSG1 <u>*LIBL</u> ENG USERMSG1 (English) USERMSG1ITL *LIBL ITL USERMSG1 (Italian)

In this example,

USERMSG1 will be exported as USERMSG1, the messages being associated with the English language.

USERMSG1ITL will be exported as USERMSG1, the messages being associated with the Italian language.

Add All Objects Worked on Under a Task Identifier

When the option to 'Add all objects worked on under a task identifier to the list' is chosen from the Work with Export List Menu a screen similar to the following example is used to display a list of task identifiers that are valid for export:

DC@P620014 Select Task Identifier(s)

Select a Task Identifier(s), so that all objects in current partition worked on under the Task Identifier will be automatically added to list

- Sel Task ID Description
- _ 00000001 Design A/C module database Ref. 0330
- _ 00000002 User change request Ref. 0331
- _ 00000003 Implement new A/C module Ref. 0332
- _ 00000004 User change request Ref. 0333

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This facility allows you to add all objects to the export list (LANSA fields, files, functions, processes, application templates, system variables, multilingual variables and web components) for the current partition that have been worked on under a task identifier(s), by selecting a task identifier(s) from a list of valid task identifiers.

All task identifiers that are currently set to 'CLS' (Closed) status, and that the user is authorized to, will be available for selection to export.

When a task identifier has been selected, all objects (LANSA fields, files, functions, processes, application templates, system variables, multilingual variables and web components) for the current partition that have been worked on under the task identifier will be automatically included in the export list.

Any field, file, process, function, application template, system variable, multilingual variable and web component that is not already in the list is automatically added to the list.

This operation is performed while you wait.

Once all objects have been included into the export list it should be reviewed by using the 'Review/Delete Objects Already in the List' option.

During this review step, the default values used for exporting file definitions, processes and functions should be carefully checked and changed if not suitable.

The time taken by this option is dependent upon the number of objects that have been worked on under the task identifier for the current partition. When it has completed the Work with Export List Menu will be re-displayed with a completion message.

Add Export Lists to the List

When the option to 'Add export lists to list' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620017 Add Export List(s) to List

Enter full or partial name of the export lists to be worked with or leave blank to select from all export lists

		Target	
Sel	List D	escription Sys	tem Type
_	APEXPL	Accounts Payable Module	AS/400
_	AREXPL	Accounts Receivable Module	AS/400
_	GLEXPL	General Ledger Module	AS/400
_	HREXPL	Human Resources Module	AS/400
_	PUEXPL	Purchasing Module	AS/400
_	SPEXPL	Sales Processing Module	AS/400
_	PREXPL	LANSA/Project Master	AS/400
_	SVEXPL	All system variables	AS/400

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This facility allows you to build temporary lists of LANSA export lists and display them on the screen. From these displayed lists, export lists can be chosen for inclusion into your export list.

These lists can be built from either all export lists, or only from export lists with a specified generic name.

A displayed list may not fit on one screen. This is indicated by a '+' sign in the

lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

Export lists shown in the displayed list which are already included in the export list are shown with an 'X' beside them. The 'X' cannot be removed. Thus an export list cannot be removed from the export list by using this facility. Use the option 'Review/Delete Objects Already in the List' to do this.

When you want to choose an export list from the displayed list for inclusion into the export list enter any non blank character beside it in the column headed 'Sel'.

The definition of the export list (list entries, substitution variables, and before/after import commands) as opposed to the objects listed in the selected export list is what will be exported.

When you have finished using this facility, use the Cancel function key to return to the Work with Export List Menu.

Review/Delete Objects Already in the List

When the option to 'Review/Delete Objects Already in the List' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620010 Review/Delete Objects in List

Drop From	Target Comp	Сору
List Object/Type/Libra	ry or Process/Description	Library Form Data
_ @@UPID /FIEL	D / //Field update / ac	
_ ACTABB /FIELI	D / /Act Abbreviation	
_ ACTRPD /FIELI	D / /ct repeal date	
_ DEPTAB /FILE	/DC@DEMOLIB/Departme	ent code t <u>\$\$DTALIB\$\$</u> <u>Y</u>
_ PSLSYS /PROCI	ESS / /Personnel System	1 <u>Y</u>
_ EMPLIST /FUNC	TION /PSLSYS /Full emple	oyee lis <u>Y</u>
_ DEPTMENT /WE	BCOMP /VISUAL /IDEPT	MENT

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=ChgView F22=ShowL

The screen displays details of all objects currently defined in the export list.

A displayed export list may not fit on one screen. This is indicated by a '+' sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed export list.

The list of objects is displayed in object type sequence. You may use the **Position to TYPE...** to position the display of objects to a page containing the start of the requested object type. The value entered for type may be the type as shown in the object list or an abbreviation (FLD, FIL, PF, LOG, LF, PROC, FUNC, FUN, SV, SYS, MTXT, MLT, TMP, LST, HTML, COMPONENT, COMP, XSL, OTH, B, T). T will position the display to the top of the list and B to the bottom.

Drop From List

Enter Y' beside the object to be deleted and press enter. The object is dropped and the revised list is re-displayed.

Target Library

The Target Library of any LANSA file definition or non-LANSA physical file can be altered by over-typing and pressing enter. The new name entered must be a valid library name or else a 'substitution value'.

If a 'substitution value' is used, the importer can nominate the library to be used as the target library during the import run. Substitution values are described in more detail in a later section of this chapter.

Comp Form (Compiled Form)

This option indicates that the file, process or function is to be exported in 'compiled' or 'ready to use' form. If this option is taken, the file, process or function will be usable as soon as it has been imported into the target system.

If this option is not taken, the file, process or function will have to be 'made operational' (for files) or 'compiled' (for processes and functions) on the target system after it has been imported.

For Web XSL style sheets, this option reflects the value entered in option *Publish on import*' in the 'Adding Web XSL Style Sheets to the List' screen.

This column cannot be altered when it contains an 'L', which indicates that individual logical views have been selected for the file.

The compiled form associated with any LANSA file definition, process or function may be altered by over-typing and pressing Enter.

The *Comp Form* is available for some of Visual LANSA types in a full RDMLX partition.

Copy Data

This option, associated with any LANSA file definition or non-LANSA physical file, can be altered by over-typing and pressing *Enter*.

A LANSA file used with this option, will cause the data in the file to be exported as well and it will replace any data already in the file on a target machine. If this option is not used, data already in the file on a target machine will be left intact (after undergoing any necessary conversion to match the new file layout).

A non-LANSA physical file used with this option, will cause the data in the file to be exported as well. However, it is the your responsibility to convert or

replace data in any existing version of the file on the target system. This can usually be achieved by executing IBM i commands before and/or after the import run on the target machine. This facility is described in Define IBM i Command to Execute.

Show Long Name

You can show the long name for an object in the list by placing the cursor on it and pressing **F22**. Refer to Long Names for more details.

Define Substitution Variables Used in a List

When the option to 'Define substitution variables used in list' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P	620011	Define Substitution Variables		
List	: TEST2	Export details in compiled form		
Variable Substitution value / Default value Prompt user Name Prompt text to display if prompt req When importing \$\$DTALIB\$\$				
\$\$SYSNAM\$\$ Specify the 4 letter prefix/name of your system			Y	
			_	
Fnn=H	elp Fnn=Ez	xit Fnn=Cancel Fnn=Messages		

A substitution value is a symbolic name that can be used:

- As the target library for any LANSA file definition being imported in a machine. The facility that allows LANSA file definitions to be included into an export list is described in Add Files to the List.
- As the target library for any non-LANSA object being imported into a machine. The facility that allows non-LANSA objects to be included into an export list is described in Add Non-LANSA Objects to the List.
- Anywhere in an IBM i command that is to be executed before or after an

import run. The facility that allows commands to be defined for execution before or after an import run is described in a later section of this chapter.

Substitution variables are useful because they let the importing site specify what value should actually be used for some activity, rather than the exporting site having to know in advance and 'hard-code' into the export list what the value should be.

For example, an exporting site can indicate that a physical file should be imported into a library called QGPL. Thus the importer has no chance to actually change the library name when importing the information in his/her system.

However, by using a substitution variable, the exporter can indicate that the file should be imported into a 'variable' library called \$\$DTALIB\$\$. When the job is run, LANSA will ask the importer what value should actually be used in place of variable \$\$DTALIB\$\$.

Substitution variables names always have the format:

\$\$XXXXX\$\$

where XXXXXX is any set of non-blank letters or numbers.

Every export list contains one special substitution variable called \$\$DTALIB\$\$ which cannot be deleted. This is the default variable used by LANSA when importing file definitions into any system.

As many other variables as required can be defined within an export list provided that their names are unique within the list and they have the correct name format of \$XXXXXX\$.

An import job produces a report just like an export job does. All interactions with the importer, with regard to substitution variables, are recorded on the report.

By using the 'Define substitution variables in list' facility it is possible to:

- Define new substitution variables.
- Alter details of existing substitution variables.
- Delete existing substitution variables.

To delete an existing variable, simply blank out/field exit over all details of the variable and press enter.

To define a new variable or alter an existing variable, enter/modify details as follows:

• The Variable Name must be in the form \$\$XXXXX\$\$ where XXXXXX is

any combination of numbers and letters (must be uppercase). Preferably the xxxxxx portion should be meaningful in some way.

- The Substitution Value / Default Value only demands mandatory entry if the variable is not to be prompted on import (see 'Prompt When Input' explanation below). If there is to be no prompt on import, the importing user will not be asked to supply a value for the variable and the value specified here will be used.
- When used with a prompted variable, the value entered here is the default value used if the importer either does not reply to the message, or else indicates that the default value should be used.
- The Prompt When Input value indicates whether or not the importing user should be prompted to supply a value for the variable.
- Enter 'Y' to indicate that the importer should be prompted. In this case prompt text will be required.
- Leave blank or enter 'N' to indicate that the importer should not be prompted. In this case a substitution / default value for the variable will be required.
- The Prompt Text specifies what message should be sent to the importer as a prompt for the variable when running an import job. Prompt text must be specified when a variable is prompted. The message should be as meaningful as possible in the space provided and normally ends with a question mark (?).
- A special feature allows the substitution / default variable to be specified as a LANSA system variable name. This means that more flexibility can be introduced into the preparation of export lists for import with no operator intervention.

This example uses the shipped system variable *PARTDTALIB (Partition Data Library) as a default value for the substitution value. When this list is imported this system value will be set to the correct library name (e.g.: DC@MODLIB) and then the operator will be prompted with DC@MODLIB as the default reply because 'Prompt' is Y (yes).

VariableSubstitution value / Default valuePrompt userNamePrompt text to display if prompt reqWhen importing

\$\$DTALIB\$\$ <u>*PARTDTALIB</u> Y

Specify library into which database files are to be imported

However, this example uses the shipped system variable *PARTDTALIB, but specifies N (no) for 'Prompt'. This means that the substitution value will be set to the partition data library name and then the import will proceed without asking the operator to confirm this value.

Variable	Substitution value / Default value Prompt user				
Name	Prompt text to display if prompt req When imp	orting			
\$\$DTALIB\$\$ <u>*PARTDTALIB</u> N					
Specify library into which database files are to be imported					

Any shipped LANSA system variable that is evaluated by program M@SYSVAR, or any user defined system variable at all, can be used in this manner.

Only alphanumeric system variables up to a maximum length of 50 characters should be used this way.

Define IBM i Command to Execute

When the option to *Define IBM i command to execute before/after import* is chosen from the 5.12.3 Work with Export Lists menu, a screen similar to the following example is used to manipulate the export list:

DC@P620012 Commands to Execute on Import

Execute Sequence Ignore Bef/Aft Number Error IBM i command

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Since the LANSA export/import facilities support the transfer of non-LANSA objects between systems, a method by which user defined IBM i operating system commands can be executed as part of the import job is also required. To define an operating system command as part of the export list the following information must be specified:

- 1. When the command is to be executed (before or after the import run).
- 2. The order that the command is to be executed in (relative to any other commands specified).
- 3. What should happen if an error is detected when the command is executed.
- 4. The command that is to be executed. The format of the command varies according to the type of machine the export list is being prepared on and the ultimate destination of the exported data.

An import job produces a report just like an export job does. All user-defined commands executed are recorded on the report, including any error message details resulting from the command.

By using the *Define IBM i command to execute before/after import* facility it is possible to:

- Define new commands.
- Alter details of existing commands.
- Delete existing command details.

To delete an existing command, simply blank out/field exit over all details of the command and press enter.

To define a new command or alter an existing command, enter/modify details according to the following:

The Execute Bef/Aft field is used to specify when the command is to be executed. It must be either B (before import) or A (after import).

The Sequence Number field is used to specify when the command should be executed relative to other commands in the same before or after group. When inserting new commands use integer sequence numbers like 1, 2, 3,etc. This will allow the later insertion of any missing commands by using sequence numbers like 1.2, 2.04, etc.

The Ignore Error field is used to indicate what should happen if an error is detected when actually executing the command. Specify Y if any errors are to be ignored and the next command executed. Enter N if the errors are not to be ignored. When an error occurs, and it is not ignored, the import job will fail with fatal errors.

The IBM i command field allows the specification of the actual command to be executed. After the command has been specified it is checked for syntax via the IBM supplied program QCMDCHK.

Because the IBM supplied syntax checker programs are used, it is also possible

to prompt commands when they are being defined.

To do this, precede the command to be used by a question mark (?) and press enter. A standard operating system command prompt will result. Fill in all required parameters and press enter again. The final version of your command will then be re-displayed.

This feature can also be used to 're-prompt' existing commands. Simply insert a question mark before the command and press enter.

Remember that any command can also contain substitution variables. The method of defining the use of substitution variables in export lists is described in Define Substitution Variables Used in a List.

When substitution variables are actually replaced in a command just prior to execution, trailing blanks in the substitution value are stripped off. For example, if a command used a substitution variable called \$\$PGMLIB\$\$, and a command defined to execute before import looked like this:

DLTPGM PGM(\$\$PGMLIB\$\$/TESTPGM)

and at execution time variable \$\$PGMLIB\$\$ contained 'QGPL', then the command actually executed would look like this:

DLTPGM PGM(QGPL/TESTPGM)

Similarly a command to be executed may actually be expanded if the substitution variable's value is actually longer than the space used by the substitution variable within the command string. For example, consider the command:

SNDMSG MSG('\$\$USRMSG\$\$') TOMSGQ(\$\$MSGQUE\$\$)

If at execution time \$\$USRMSG\$\$ contained 'HELLO THERE, HOW ARE YOU' and \$\$MSGQUE\$\$ contained 'WS1', then the actual command executed would be:

SNDMSG MSG('HELLO THERE, HOW ARE YOU') TOMSGQ(WS1)

Add Web HTML or XML Components to the List

When the option to 'Add web HTML components to list' or 'Add web XML components to list' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620018 Add Web Components to List

Enter full or partial name of the Web Component to be worked with or leave blank to select from all Web Components

Sel	Component	Type	Description	Valu	e	
_	ADVERT	BANNE	ER Advertising	g banners	ADVERT	
_	DEPTMENT	VISU	AL Drop dow	n for depar	tment IDEPTME	ENT
_	LOADINST	TEXT	Default load	instruct	onLoad='timerO	NE=
_	PAYMENT	VISUA	AL Check box]	PAYMENT	
_	STDFOOTE	R PAGI	E Standard F	ooter	STDFOOTER	
_	TABLE A	AFILE	Table file	WWW	/LIB /TABFIL	

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This facility allows you to build temporary lists of LANSA for the Web components and display them on the screen. From the displayed list, web components can be chosen for inclusion into your export list.

These lists can be built from either all web components, or only from components with a specified generic name.

Web components shown in the displayed list which are already included in the export list are shown with an 'X' or 'G' beside them. The 'X' or 'G' cannot be removed. Use the option 'Review/Delete Objects Already in the List' to remove

web components from the export list.

When you want to choose a web component from the displayed list for inclusion into the export list enter any non blank character beside it in the column headed 'Sel'. Alternatively you may use the Generic function key to add the generic name into the export list.

The definition of the web component and associated web pages, if any, and associated banner component, if any, will be exported.

When you have finished using this facility, use the Cancel function key to return to the Work with Export List Menu.

Add Technology Services to the List

When the option to 'Add Technology service to list' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620021 Add Technology services to List

Enter full or partial name of the Technology service to be worked with or leave blank to select from all Technology services _____

Sel	Application	n Desc	ription
_	LANSA	PPC_XHTN	IL HTML for Pocket PC 2003
_	LANSA	XHTML	Extensible Hypertext Markup Language

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This facility allows you to build temporary lists of LANSA for the Technology services and display them on the screen. From the displayed list, web XML applications can be chosen for inclusion into your export list.

These lists can be built from either all Technology services, or only from Technology services with a specified generic name.

Technology services shown in the displayed list which are already included in the export list are shown with an 'X' or 'G' beside them. The 'X' or 'G' cannot be removed. Use the option 'Review/Delete Objects Already in the List' to remove Technology services from the export list.

When you want to choose a Technology service from the displayed list for inclusion into the export list enter any non blank character beside it in the column headed 'Sel'. Alternatively you may use the Generic function key to add the generic name into the export list.

The definition of the Technology service and its XSL stylesheets will be exported.

When you have finished using this facility, use the Cancel function key to return to the Work with Export List Menu.

Add Weblets to the List

When the option to 'Add Weblets' is chosen from the Work with Export List Menu a screen similar to the following example is used to manipulate the export list:

DC@P620022 Add Weblets to List

Enter full or partial name of the Weblet to be worked with or leave blank to select from all Weblets

Sel	Name	Description	
_	std_anchor	Standard Hyperlink	
_	std_attachment_	panel Panel with attachment layout manage	r
_	std_banner	Standard Banner	
_	std_boolean	Standard Boolean Visualization	
_	std_button	Standard Button	
_	std_char	Standard Char Visualization	
_	std_checkbox	Standard Checkbox	
_	<pre>std_click_image</pre>	Standard Selectable Image	
_	std_date	Standard Date Visualization	
_	std_datetime	Standard DateTime weblet	
_	std_dhtml_menu	Standard DHTML Menu	+

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This facility allows you to build temporary lists of LANSA for the Weblets and display them on the screen. From the displayed list, Weblets can be chosen for inclusion into your export list.

These lists can be built from either all Weblets, or only from Weblets with a specified generic name.

Weblets shown in the displayed list which are already included in the export list

are shown with an 'X' or 'G' beside them. The 'X' or 'G' cannot be removed. Use the option 'Review/Delete Objects Already in the List' to remove Weblets from the export list.

When you want to choose a Weblet from the displayed list for inclusion into the export list enter any non blank character beside it in the column headed 'Sel'. Alternatively you may use the Generic function key to add the generic name into the export list.

The definition of the Weblet and its published XSL stylesheet will be exported. When you have finished using this facility, use the Cancel function key to return to the Work with Export List Menu.

Add Visual LANSA Components to the list

When the option to *Add Visual LANSA components to list* is chosen from the Work with Export List Menu, the Add Visual LANSA Components to List screen is used to manipulate the export list.

DC@P620024 Add Visual LANSA Components to List

Enter full or partial name of the Visual LANSA components to be worked with or leave blank to select from all components _____

Filter by type: _____

Sel Component	Description	Type
STD_BTMAP	Standard bitmap	Bitmap
STD_CRSR	Standard cursor	Cursor
STD_FORM	Standard form	Form

Fnn=Help Fnn=Exit Fnn=Generic Fnn=Cancel Fnn=Messages F22=ShowLo

This facility allows you to build temporary lists of visual LANSA components from the LANSA data dictionary and display them on the screen. Components can be chosen from the displayed list for inclusion in your export list.

Components in the displayed list that are already included in the export list are shown with an 'X' or 'G' beside them. To choose a component from the displayed list for inclusion into the export list, enter a 'Y' beside it in the column headed 'Sel'. Alternatively you may use the Generic function key to add the generic name into the export list.

A 'Comp Form' (Compiled Form) column is included in full RDMLX partitions. A 'Y' entered in this column indicates that the compiled form of the component is included for those component type which may be compiled.

Additionally, you can show the long name for a VL component in the list by

placing the cursor on it and pressing **F22**. Refer to Long Names for more details.

The list may be filtered by type of component.

Possible values for the filter are: FORM, FRM, WAM, REUSABLE, RUP, BITMAP, ICON, VIS_STYLE, STYLE, VISUAL, ACTIVEX, CURSOR, PRIMITIVE, PRIM.

5.12.4 Steps to Import Objects

The steps involved in importing objects from a CD-ROM, magnetic tape or a save file can be visualized like this:


Submit an Import Run to Batch

When the *Import objects into this partition* option is selected on the Housekeeping Menu a prompt screen like this will be displayed:

Submit Import Run to Batch DC@P621101 Submit batch Import or Print contents: YES YES, NO, PRT Using Job name <u>LANSAIMP</u> Job description <u>*LIBL/QBATCH</u> Job queue <u>*JOBD</u> Allow Long Name Changes / Allow Type Changes . . <u>NO / NO YES</u>, NO Import from tape or optical device named _ using tape file seq.no. or optical file name ***SEARCH Or** Import from save file named Send messages and inquiries to message queue . . <u>WS2</u> Source system type <u>AS/400</u> AS/400 YES, NO Assign new internal names YES Delete \$\$ files / Omit Frameworks and Groups. . . <u>NO</u> / <u>NO</u> YES, NO Default language Import all languages in exported data _ YES, NO Select languages to be imported or *ALL *ALL Sel Language Description DEU German ENG English +

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages F16=Settings

This import run confirmation screen requests that details of the actual import job (which will run in batch) be specified. The details required to successfully run the import job are:

Submit batch Import or Print Contents

This field may have a value of YES, NO or PRT.

Enter YES to submit a job to batch to import details from the Save file or device file into this partition.

Enter PRT to submit a job to batch to print a report which lists the contents of the Save file or device file. The file specified must have been created by a LANSA export. This may be used to check the contents of a Save file or device file before importing.

Enter NO to cancel the job submission.

Using Job Name

Job Description

Job Queue

Output Queue

Allow Long Name Changes

The value must be YES or NO.

The default value is NO, which indicates that any long name changes in the import will cause the import to fail.

If YES is specified, then long name changes in the import will be allowed.

Allow Type Changes

The value must be YES or NO.

The default value is NO, which indicates that any field or component type changes in the import will cause the import to fail.

If YES is specified, then field or component type changes in the import will be allowed.

Import from Tape or Optical Device Named

Use this field when you want to import objects from optical device or magnetic tape.

Specify the name of the actual optical or tape device. For instance, names like OPT01 and TAP01 are commonly used.

Using File Sequence Number/Optical file name

This field is used with the tape or optical device name to specify the file sequence number/optical file name to be used for the import of objects from tape or optical device. The default tape file sequence number is *SEARCH. The optical device name can be up to 80 characters long.

Or Import from Save File Named

Use this field if you want to import objects from a save file.

A save file is an object created and maintained by the operating system and it can be treated as if were a magnetic tape that is actually stored on disk. Refer to the operating system command CRTSAVF (create save file) in the IBM supplied manual for *details* about creating and manipulating save files.

Save files are commonly used in the following situations:

- When you are exporting/importing between LANSA systems that are resident on the same CPU.
- When you wish to transfer the exported objects via a communications link to another machine.

Send Messages and Inquiries to Message Queue

Specifies where any inquires or messages issued by the batch import job should be sent. The default value is the message queue associated with the workstation from which the job is being submitted. Any other valid message queue name can be specified.

Source System Type

Specifies the type of system from which the data on the CD-ROM, disk, magnetic tape or save file was originally exported. The default value is the same type of system as the one being used to import the objects. Allowable value is AS/400.

The value specified here tells the import job in what format the data on the CD-ROM, tape or disk is to be processed.

Assign New Internal Names

The value you specify here must be YES or NO.

The default value is YES, which indicates the internal object and help text pointer names associated with objects being imported should be re-assigned to prevent any possible conflict with existing internal object names within the current partition.

If the data set you are importing from was prepared on a LANSA system earlier than Version 4.0, program change (PC) level D4, you should always use the default YES option.

If your organization did not prepare the import data set, or it has been given to you by any other organization outside of your own control, you should also use the default YES option. Use the NO option only if you had control over the preparation of the import data set, and fully understand the implications and processing changes that using this option causes.

Use of the NO option provides 3 significant benefits:

- 1. The import run will execute slightly more quickly.
- 2. All objects will have the same names as they did in the partition that exported them. This makes problem tracing from a production partition back to a development partition slightly easier.
- 3. Processes that have been exported in compiled form can be fully imported in compiled form, ready to run. This removes the need to recompile the processes after the import run has completed (even though this is actually done automatically for you). This means that importing LANSA systems can run fully compiled processes, rather than interpretive mode processes, without having to have an RPG compiler present.

It is important that you understand what is involved in internal object naming used by LANSA, and why it can cause a conflict when the NO option is used.

Whenever an 'object' is created within a partition an 'internal' object name is assigned and allocated to it.

The final form of the name has varying prefixes and suffixes, depending upon the object type, but the name always contains a component that has a format like this:

unnnnn

where 'u' is the 'unique prefix' assigned to the partition and 'nnnnnn' is a unique number assigned from within the partition.

This explains why you are requested to ensure that every single partition created within your organization has a different 'unique prefix'. This means that no matter which partition you choose to create an object in, you will generate a unique object name (within your organization). Thus you can probably use the NO option here without causing any problems.

Note that when you assign a 'unique prefix' to partitions in your organization you should include partitions within the same LANSA system, partitions within different LANSA systems on the same CPU, and partitions within LANSA systems located on different CPUs.

You can probably see now why you should not use the NO option when you did not prepare the import data set from within your organization. This is because the import data set might contain internal object names that duplicate the names of objects that already exist in your partition.

Once you have imported using YES, you can then export again and use NO in other partitions within your network, because the new exported data set will use internal object names that are now okay for use within all your partitions.

Important Note: If you have previously imported a process in an import run using the YES option (or before this facility was available, which is equivalent to using the YES option), and you are now going to (re)import it using the NO option, ensure that all of its associated RDML functions are also (re)imported in the same run. If you fail to do this you will find that the (re)imported compiled process may attempt to access objects by an internal name that is not correct. If this problem occurs it can be simply overcome by either recompiling the process, or by running the import run again so that all functions associated with the process are imported correctly.

This problem should only occur if your organization switches over from using the YES option to using the NO option and you forget to export all functions associated with a process.

Once you have decided on an internal organization value of YES or NO for this option, you should stick to the chosen value and avoid swapping back and forward between the values. Continual swapping will result in having to recompile processes so that they 'lock in' to the correct set of internal names for their associated functions.

Delete \$\$ files

The value must be YES or NO.

The default value is NO, which indicates that a \$\$ version of a file being imported is not automatically deleted. During import, the current version of a file to be imported is renamed to a \$\$ version of that file. If a \$\$ version already exists, a message will be issued requesting the action to be taken. A 'y' for yes, reply will delete the \$\$ version and complete the import of the file. A 'n', for no, reply will retain the old \$\$ version and not import the new file version.

A YES value for this option will automatically delete the \$\$ version of a file encountered in importing without issuing the message which requires a response. This option may be used if you wish to run import jobs in an unattended mode.

Where 9 and 10 character file names are not unique in their first 8 characters, a \$\$ version for the different file may erroneously be found to exist. It must still be deleted.

Omit Frameworks and Groups from imported data

Use this field to indicate if Frameworks and Groups for all objects in this import list are to be included or omitted from the imported data.

The default is NO.

If NO is entered, the Frameworks and Groups will be included in the imported data. In this case, the Frameworks and Groups will be available after this list is imported into a LANSA partition.

If YES is entered, the Frameworks and Groups will be omitted from the imported data. In this case, the Frameworks and Groups for all objects in the import list will not be included in the import, and will therefore not be available after this list is imported into a LANSA partition.

Default Language

This option will only appear if the current partition is a multilingual partition.

Use this field to indicate which language should be used as the default language when importing data from a non-multilingual partition. The default is the current language.

Import All Languages in Exported Data

This option will only appear if the current partition is a multilingual partition.

Use this field to indicate that all the languages that are included in the exported data are to be imported into this multilingual partition. The default is NO.

If YES is entered:

- Each language that is included in the exported data that is not currently defined in this partition will have its language definition added to this partition definition from the exported data.
- Each language that is included in the exported data that is already defined in this partition will not have its definition updated.
- Language details for all imported objects (field, files, processes, functions, etc.) will be imported for each language included in the exported data.

If YES is entered, then languages cannot be selected for the option *Select Languages to be Imported or *ALL*.

Select Languages to be Imported or *ALL

This option will only appear if the current partition is a multilingual partition. Use this field to indicate which of the partition languages in the exported data should be included in the imported data. The default is *ALL. If *ALL is entered, then all of the partition languages (which appear in the list below) will be included in the imported data if that language is included in the exported data.

If this field is left blank, then you can select the required partition languages to be included in the imported data from the list of partition languages below.

If languages are selected using this option, then YES cannot be entered for the option *Import All Languages in Exported Data*.

F16=Settings: displays the system setting relevant to the export and import. For details, refer to Export and Import settings. You will only be able to change these settings if you have access to the Administration Menu - Review System Settings.

5.12.5 Exporting Objects - Tips and Techniques

The following things about exporting should be noted before attempting to export objects to another system:

Exporting and Importing between RDML and RDMLX partitions

- RDML objects may be exported and imported between partitions without restriction.
- RDMLX objects may be exported from a partition enabled for full RDMLX and imported into another partition enabled for full RDMLX.
- RDMLX objects may not be imported in an RDML partition.
- An AS/400 export which contains a mixture of RDML and RDMLX objects may be imported into an RDML partition. The RDML objects will be imported. The RDMLX objects will not be imported. They will be reported with a warning message. For example,

Definition of field STD_DATEX was not successfully exported and so it cannot be imported.

A Report Is Produced Which Details Each Step of the Export Job

The report takes the form of a series of messages that are listed on the report. Each message has a unique identifying number, a type (completion, warning or fatal) and a line of text that describes what event took place.

At the end of the report a summary of the number of completion, warning and fatal messages issued during the export job is printed on the report.

If any fatal message is issued the exported data is unusable (regardless of whether or not anything was actually written out onto the tape or disk). In this case, investigate the fatal messages on the report, correct the cause and attempt the export operation again.

An example of a fatal error would be an error on the tape drive that occurred while attempting to write to the tape.

When one or more fatal messages are issued the export job will end abnormally (as indicated by the operating system message sent back to the submitting workstation).

If any warning message is issued the exported data is usable, but may not be in exactly the form you expected. In this case, investigate all warning messages on the report and decide whether you have to correct a problem and run the export job again. An example of a warning message would be that you requested a certain RDML function be exported in compiled form, but the function is not currently compiled.

When one or more warning messages are issued the export job will still end normally (as indicated by the operating system message sent back to the submitting workstation).

When the Export Job Is Submitted from a Workstation

When the export job is submitted from a workstation, a summary message will be sent back to the workstation at job completion indicating the number of completion, warning and fatal messages issued by the job (as per the summary lines at the end of the export report).

The Format of Tapes / Disks

The format of tapes / disks used varies with the source (i.e. exporting) and target (i.e. importing) system types. Currently only the IBM i is supported for exports via tapes as per the following table:

Source System Target System		Format Of Data On Tape / Disk	
IBM i	IBM i	Operating system save restore	

Transfer of Programs in Compiled Form

Some restrictions to the export / import of programs in compiled form apply as shown in the following table. The term 'observable' refers to a program that has all its symbolic debug information intact. In LANSA terms an 'observable' RDML function or I/O module is one that is compiled to be 'debug capable'.

In copying from on RISC system to another, observable or non-observable will be subject to operating system release compatibility.

Clearing/Initializing Tapes and Disks Before Exporting

It is advisable to clear any tapes or disks to be used before commencing the export job. The most common export problems involve un-cleared or uninitialized tapes or disks.

Operating system commands are used to initialize and/or clear tapes and disks. Refer to the commands INZTAP, CLRDKT and INZDKT in the IBM supplied operating system manuals for *details*.

Re-Running Import Jobs

The need to re-run an import job on a target system is a distinct possibility. Machine failures and incorrect replies to inquiry messages guarantee that from time to time an import job will have to be re-run.

As such, all the logic used by LANSA to import fields, files, processes and functions has been designed so that it can be easily re-run with no specific actions required by the importer other than to start the job executing again.

This factor should be taken into account when adding other / non-LANSA objects to an export list and any accompanying operating system commands to be executed before or after the import run. When taking any action with an operating system command, consider how the command will react if it has already been executed in a previous run of the import job.

Minimize Operating System Security Interference

The LANSA export and import facilities make use of IBM i operating system features to save and restore data.

This area of the operating system is subject to a large number of operating system security checks. The continual interference of such checks in your export/import job may confuse you and seem to indicate that the LANSA export/import facilities are not working correctly.

To avoid this situation, it is recommended that you run all export and import jobs under a user profile that is the operating system security officer QSECOFR, or, part of the system security officer group, or at the very least, has special authority *SAVSYS allowed.

The Result Area DC@RESLT in QTEMP Is Set By an Export Job

This data area, which is created after some LANSA operations complete, can be accessed by other programs to test the result of the attempted export operation. Refer to the DC@RESLT Data Area for *details* of this data area and what it contains.

Use the 'Omit RDML Code' Option with Extreme Caution

Clearly mark all tapes with 'Import will delete source code'.

Avoid Large Export Lists

To reduce the amount of time taken to investigate and re-run an export job in error, limit the size of export lists where ever possible, for example when exporting a whole partition set up a separate export list for each object type opposed to one export list for the whole partition:

LIST01 All fields in a partition

LIST02 All files in a partition

LIST03 All processes and functions in a partition

To avoid large export lists it is recommended that you set up and run numerous medium sized export jobs opposed to 1 very large export job.

This tip has added significance when the target system type is PC. When importing into LANSA, the SQL database does not commit the data until the end of the import operation. Therefore large imports require large amounts of disk storage, and the import may 'crash' if insufficient storage was previously allocated. For example, if you were 1 hour into a large import and the import failed, resulting in all the data being backed out - you would have to run the whole import again.

Multilingual Considerations for WAMs and Weblets

WAMs and Weblets have published documents. These documents are published for each Technology Service and partition language combination. On IBM i, published documents are exported together with the repository details. You should always export the default language, as it is used during import if you are not exporting the importing partition default language.

Example of 'Other' Object Export

Since the LANSA export/import facilities support the transfer of non-LANSA objects between systems, you may specify user defined IBM i operating system commands which can be executed as part of the import job. Typical examples would be the transfer of objects such as message files, non-LANSA created database files and programs.

This is an example of the types of commands that would most likely be executed before and after an import of these non-LANSA objects:

DC@P620012 Commands to Execute on Import List : EXAMPLE Exporting other obects Execute Sequence Ignore Bef/Aft Number Error IBM i command

В	_1.00 Y	DLTMSGF MSGF(USRLIB/USR01MSG)
B NEW0	2.00 N OB J(XXD'	RNMOBJ OBJ(\$\$DTALIB\$\$/DTAFIL OBJTYPE(*F] FAFIL)
В	_3.00 Y	DLTPGM PGM(USRLIB/USRPGM)
A TOFII	1.00 N Le(\$\$DTA1	CPYF FROMFILE(\$\$DTALIB\$\$/XXDTAFIL) LI B\$\$/DTAFIL) MBROPT(*REPLACE)
Fnn=Hel	p Fnn=Exi	Fnn=Cancel Fnn=Messages

The above commands will delete the message file and program and rename the database file before the import, and then copy file the data from the old file to the imported file after the import.

All of the actions are executed automatically for you by Import when dealing with LANSA objects.

Avoiding Inquiry Message to the Operator When Importing

Every export list contains one special substitution variable called \$\$DTALIB\$\$ which cannot be deleted. This is the default variable used by LANSA when importing file definitions into any system.

If you don't want the associated 'question' to be asked on import, then you should provide a literal (e.g.: QGPL) or system variable name (e.g.: *PARTDTALIB) as the default value, and set 'Prompt user when Importing' to 'N' for the \$\$DTALIB\$\$ substitution variable.

This approach can be applied to any substitution variable that you have defined in an export list.

When the target system for an export is **PC** the file library details must be provided as follows:

- Libraries must be specified using the substitution variables.
- If a hardcoded library value is provided, instead of a substitution variable, the file will be imported into the library used on the exporting system.
- The substitution variables must resolve to a system variable which will exist

on the target system. Do not set the substitution variable to a hardcoded library.

• If the substitution variable cannot be resolved when importing on the target system the file library will default to the current partition library.

Requests to 'Prompt user when Importing' will be ignored when importing to PC.

5.12.6 Importing Objects - Tips and Techniques

The following things about importing should be noted before attempting to import objects into another system:

A Report Is Produced Which Details Each Step of the Import Job

The report takes the form of a series of messages that are listed on the report. Each message has a unique identifying number, a type (completion, warning or fatal) and a line of text that describes what event took place.

At the end of the report a summary of the number of completion, warning and fatal messages issued during the import job is printed on the report.

If any fatal message is issued the imported data may be unusable. In this case, investigate the fatal messages on the report, correct the cause and attempt the import operation again.

An example of fatal error would be an error on tape drive that occurred while attempting to read from the tape.

When one or more fatal messages are issued the import job will end abnormally (as indicated by the operating system message sent back to the submitting workstation).

If any warning message is issued the imported data is usable, but may not be in exactly the form you expected. In this case, investigate all warning messages on the report and decide whether you have to correct a problem and run the import job again.

When one or more warning messages are issued the import job will still end normally (as indicated by the operating system message sent back to the submitting workstation).

When the Import Job Is Submitted From a Workstation

When the import job is submitted from a workstation, a summary message will be sent back to the workstation at job completion indicating the number of completion, warning and fatal messages issued by the job (as per the summary lines at the end of the import report).

Mounting of Tapes and CD-ROMs before Importing

It is advisable to mount any tapes or CD-ROMs to be used before commencing the import job. The most common import problems involve tape or CD-ROM handling.

Replying to Inquiry Messages

The person who prepared the data being imported may have set it up so that a number of questions must be answered just after the import job commences execution (refer to Define Substitution Variables Used in a List for details).

These questions are sent to the message queue nominated when the import job was submitted as a series of inquiry messages.

When one arrives it has to be answered within one hour of arrival. Until an answer is given the import job is suspended and may hold up other jobs waiting to execute. If an import job appears to be suspended (i.e. it is doing nothing and not using any system resources) then it is probably waiting for a reply to an inquiry.

These inquiry messages take two forms (depending upon how the exporter set them up):

- 1. Some messages have a default reply that the exporter thought might be correct in most situations. The default reply is shown to you in the message text, and by replying with an (asterisk) you can indicate that the default reply should be used. Otherwise you can override the default with a specific reply of your own.
- 2. Other messages do not have a default reply. In this case you must provided a specific reply of your own.

Whenever you are making a specific reply of your own, check what you have replied very carefully before pressing the enter key. Almost no effective validation of your reply can be done by LANSA. An incorrect reply can completely ruin an import run and require that it be run again.

All replies are automatically converted to uppercase before being processed by LANSA, so it does not matter whether your reply is in uppercase or lowercase, or a mixture of both.

If you are unsure of exactly what to reply to an inquiry, contact someone who does know before entering any reply.

Having to reply to inquiry messages may not be necessary.

Refer to Avoid Large Export Lists in Exporting Objects - Tips and Techniques for details of how import messages can be avoided by having the person who prepares the export lists pre-define the answers.

Minimize Operating System Security Interference

The LANSA export and import facilities make use of IBM i operating system

features to save and restore data.

This area of the operating system is subject to a large number of operating system security checks. The continual interference of such checks in your export/import job may confuse you and seem to indicate that the LANSA export/import facilities are not working correctly.

To avoid this situation, it is recommended that you run all export and import jobs under a user profile that is the operating system security officer QSECOFR, or, part of the system security officer group, or at the very least, has special authority *SAVSYS allowed.

The Result Area DC@RESLT In QTEMP Is Set By an Import Job

This data area, which is created after some LANSA operations complete, can be accessed by other programs to test the result of the attempted import operation.

Refer to the DC@RESLT Data Area for *details* of this data area and what it contains.

If Export Run Used 'Omit RDML Code' Proceed With Caution

If the export run used to prepare the data set that you are about to import from used the 'Omit RDML Code' option, proceed with extreme caution.

New functions will be imported with no associated RDML source code. Replacement versions of existing functions will also be imported with no source code. Any existing RDML source code will be deleted during the import run (to maintain compatibility).

Using this feature presents a major recovery hazard unless proper system backups are kept.

Use the NO Answer for 'Assign New Internal Names' with Caution

Only use the NO answer to the 'Assign new internal names' option on the import prompt (or direct call interface) if you understand the complete implications of this feature. Refer to Assign New Internal Names for *details*.

Importing of Export Lists - Resetting of entries

When an export list is imported into a partition, LANSA will check if the object referenced on each list entry exists in the partition. If the object is found, the library field of the entry will be reset to reflect the current partition and a warning message is printed on the Import run listing.

CCSID Data Conversion

On systems that have QCCSID (Coded Character Set Identifier) and QLANGID (Language) parameters set to non-standard (not default) values, imported data will be converted according to the default CCSID of the QLANGID parameter and not the QCCSID of the system.

For example:

QCCSID = 00500 (international code page) QLANGID = CAT (Catalan)

The default CCSID for CAT (Catalan) is 00284. Therefore, all data will be converted to code page 284 and not code page 500, the QCCID value.

Multilingual Considerations for WAMs and Weblets

WAMs and Weblets have published documents. These documents are published for each Technology Service and partition language combination. If the export doesn't have published documents for the importing partition's default language, then the import program will copy (if available) the published documents from the export's default language into the importing partition's default language.

For example, if your exporting partition has language ENG as the default language and your importing partition has language FRA as the default language, if the export doesn't include language FRA, then the ENG documents are copied into documents for language FRA. This ensures that you have published documents for your partition's default language. The FRA documents are then copied to populate the other importing partition's languages for which published documents are not available.

IFS Objects

When IFS Objects are imported, the current job user will be changed to the system owner and once the IFS Objects processing is complete the current user will be re-instated.

If for the authority reasons, the current user cannot be changed to the system owner, the IFS Object's owner will be the current job user. If this is not correct you might have to change the owner of these to the system owner or to whatever owner required.

5.12.7 Direct Exporting of Objects in a List

To export to a PC system

To export to an IBM i system

To directly invoke the export facility without using any of the normal LANSA menus or prompts when the target is a LANSA for i system, the LANSA EXPORT command may be used. This enables you to invoke an export job at any time and from any type of application program.

To do this, use the LANSA command, or place a call to the LANSA program, passing parameters / arguments as described below:

Command Parameter	Call Argument Number / Type	Description
REQUEST	1 / A(10)	Pass as 'EXPORT'
PROCESS	2 / A(10)	Not required - pass as blanks
FUNCTION	3 / A(10)	Not required - pass as blanks
PARM01	4 / A(7)	Name of previously defined list that specifies object(s) to be exported. The list must have a Target System Type of AS/400.
PARM02	5 / A(10)	Device or device file name if exporting to disk or magnetic tape. Otherwise pass as blanks.
PARM03	6 / A(21)	Save file name if exporting to a save file. Use qualified name form if desired, otherwise pass as blanks.
PARM04	7 / A(7)	File sequence number if device name used in PARM02. Default is *END if left blank.
PARM05	8 / A(3)	Omit RDML source code from exported data. Pass as 'YES' or 'NO'. Default is 'NO' if left blank.
PARM06	9 / A(256)	Languages to be exported. Up to 64 x 4

		character language code(s) or *ALL to export all languages. Only applicable to multilingual partitions.	
PARM07	10 / A(3)	Not in use.	
PARM08	11 / A(3)	Include LANSA/Document details in the export. Pass as 'YES' or 'NO'. Default is 'NO' if left blank.	
		This parameter will be set to 'NO' by the program if the 'Omit RDML source code from exported data' parameter is not passed as 'NO'.	
PARM09	12 / A(3)	Omit frameworks and groups from the export. Pass as 'YES' or 'NO'. Default is NO if left blank.	
PARM10	13 / A(1)	Not required - pass as blanks	
PARTITION	14 / A(3)	Name of partition in which the list is defined.	
LANGUAGE	15 / A(4)	Language that system is to use. If not specified, the partition default language will be assumed.	

As an example consider the following operating system SBMJOB (submit job) command that begins an export job in batch:

SBMJOB RQSDTA('LANSA REQUEST(EXPORT) + PARM01(LIST01) PARM02(QTAPE) PARTITION(DEV)')

This command will submit a job to batch. The batch job will then invoke LANSA and request that all objects defined in a list called LIST01 be exported out onto a device called QTAPE (presumably a tape drive). The list is defined in the DEV partition.

It is not recommended that this facility be used to export objects interactively (i.e. directly from a workstation) because of the relatively large number of database I/Os performed.

To export to a PC system

To directly invoke the export facility without using any of the normal LANSA menus or prompts when the target is a LANSA PC system, the LANSA PCEXPORT command may be used. This enables you to invoke an export job at any time and from any type of application program.

To do this, use the LANSA command, or place a call to the LANSA program, passing parameters / arguments as described below:

Command Parameter	Call Argument Number / Type	Description
REQUEST	1 / A(10)	Pass as 'PCEXPORT'
PROCESS	2 / A(10)	Not required - pass as blanks
FUNCTION	3 / A(10)	Not required - pass as blanks
PARM01	4 / A(7)	Name of previously defined list that specifies object(s) to be exported. The list must have a Target system type of WIN, PC or OS/2.
PARM02	5 / A(1)	Check Out Usage. This must be 'R' (read only) or 'M' (maintenance).
PARM03	6 / A(8)	Task identity. If the Check Out Usage is 'M' this must be a valid task with an open status (OPN or WRK).
		If the Check Out Usage is 'R' (read only), this parameter is ignored (ie treated a blank).
PARM04	7 / A(10)	PC Name. If the Check Out Usage is 'M' this must be a valid PC Name.
		If the Check Out Usage is 'R' (read only), this parameter is ignored (ie treated as blank).
PARM05	8 / A(1)	Not required - pass as blanks
PARM06	9 / A(1)	Not required - pass as blanks
PARM07	10 / A(1)	Not required - pass as blanks
PARM08	11 / A(1)	Not required - pass as blanks

PARM09	12 / A(1)	Not required - pass as blanks	
PARM10	13 / A(1)	Not required - pass as blanks	
PARTITION	14 / A(3)	Name of partition in which the list is defined.	
LANGUAGE	15 / A(4)	Language that system is to use. If not specified, the partition default language will be assumed.	

As an example consider the following operating system SBMJOB (submit job) command that begins an export to PC job in batch:

SBMJOB RQSDTA('LANSA REQUEST(PCEXPORT) + PARM01(LISTPC) PARM02(M) PARM03(MYTASK01) PARTITION(DEV)')

This command will submit a job to batch. The batch job will then invoke LANSA and request that all objects defined in a list called LISTPC be exported. The export will create a folder in QDLS named LISTPC. The objects will be available for maintenance under task MYTASK01. The list is defined in the DEV partition.

It is not recommended that this facility be used to export objects interactively (i.e. directly from a workstation) because of the relatively large number of database I/Os performed.

5.12.8 Direct Importing of Objects

5.12.4 Steps to Import Objects describes how a job can be submitted from within LANSA to import all objects from optical device, magnetic tape or save file.

However, it is possible to directly invoke the import facility without using any of the normal LANSA menus or prompts.

This enables you to invoke an import job at any time and from any type of application program.

To do this, use the LANSA command, or place a call to the LANSA program passing parameters / arguments as described below:

Command Parameter	Call Argument Number / Type	Description
REQUEST	1 / A(10)	Pass as 'IMPORT'
PROCESS	2 / A(10)	Not required - pass as blanks
FUNCTION	3 / A(10)	Not required - pass as blanks
PARM01	4 / A(10)	Device or device file name if importing from optical device or magnetic tape. Otherwise pass as blanks.
PARM02	5 / A(21)	Save file name if importing from a save file. Use qualified name form if desired. Otherwise pass as blanks.
PARM03	6 / A(10)	Name of message queue to which messages and inquiries are to be sent.
PARM04	7 / A(6)	Source system type. AS/400.
PARM05	8 / A(80)	Tape File sequence number or optical file name if device name used in PARM01. Default for tape device is *SEARCH if left blank. The optical device name can be up to 80 characters long.
PARM06	9 / A(1)	Import all languages in exported data. Either Y

		(Yes) or N (No). Only applicable to multilingual partitions.
PARM07	10 / A(256)	Languages to be imported. Up to 64 x 4 character language code(s) or *ALL to import all languages in list. Only applicable to multilingual partitions.
PARM08	11 / A(4)	Default language. Default is the current language if left blank. Only applicable to multilingual partitions.
PARM09	12 / A(6)	Characters 1-3 Assign new internal names option. Pass as YES or NO. Defaults to YES if not specified or if passed as any value other than NO. Refer to Submit an Import Run to Batch for details and warnings about the NO option.
		Characters 4-6 Delete \$\$ files. Pass as YES or NO. Defaults to NO if not specified. If YES, any \$\$ version of the files being imported will be automatically deleted. The file name to be deleted is composed of \$\$ plus the first 8 characters of the file name. Where 9 and 10 character file names are not unique in the first 8 characters, this may result in additional \$\$ files being deleted. (For the processing to complete, these files must be deleted.)
PARM10	13 / A(19)	Characters 1-10. Reserved. Leave blank.
		Characters 11-13. Allow Name Changes. Pass as YES or NO. Defaults to NO if not specified. This means that any long name changes in the import will cause the import to fail. If YES, then long name changes in the import will be allowed. Characters 14-16. Allow Type Changes.

		Pass as YES or NO. Defaults to NO if not specified. This means that any field or component type changes in the import will cause the import to fail. If YES, then field or component type changes in the import will be allowed. Characters 17-19. Omit importing frameworks and groups. Pass as 'YES' or 'NO'. Default is NO if left blank.
PARTITION	14 / A(3)	Name of partition into which objects are to be imported.
LANGUAGE	15 / A(4)	Language that system is to use. If not specified, the partition default language will be assumed.

Consider the following examples:

Example 1

Using CMD to submit the job. Note that PARM02 and PARM04 have single quotes:

```
SBMJOB CMD(LANSA REQUEST(IMPORT)
PARM02('QGPL/EXPWEB1') PARM03(QSYSOPR) PARM04('AS/400')
PARM06(N) PARM07(*ALL) PARM08(ENG) PARM09(YESYES)
PARTITION(MX1)) JOB(TEST1) OUTQ(*CURRENT)
```

Example 2

Using RQSDTA to submit the job. Note that PARM02 and PARM04 have TWO single quotes:

```
SBMJOB rqsdta('LANSA REQUEST(IMPORT)
PARM02("QGPL/EXPWEB1") PARM03(QSYS OPR) PARM04("AS/400")
PARM06(N) PARM07(*ALL) PARM08(ENG) PARM09(YESYES)
PARTITION(MX1)') JOB(TEST1) OUTQ(*CURRENT)
```

This command will submit a job to batch. The batch job will then invoke LANSA and import all objects on the save file, MYSAVF, into multilingual

partition DEV. New internal names will be created.

It is not recommended that this facility be used to import objects interactively (i.e. directly from a workstation) because of the relatively large number of database I/Os performed.

5.13 Application Templates

Application templates in LANSA are provided to enable RDML programs to be generated automatically by a 'question and answer' session.

An application template consists of Application Template Commands and RDML commands.

The application template commands control how, when and what RDML commands are generated. They usually do this by asking the user a question, then by using the answer, generate different RDML commands.

The application template may be used as a 'skeleton' or 'model' for the generation of a complete RDML program or to generate a commonly used section of RDML code or logic.

'End user' access to application templates is supported when creating a new RDML function. An end user may select an application template that will generate a complete RDML program by answering the questions defined in the template.

Programmer access to application templates is supported when both creating a new function, and directly from the RDML editor. The programmer may choose many different application templates to automatically construct different parts of the RDML program they are creating or maintaining.

Some application templates are shipped with the LANSA product.

You can modify templates that are shipped with the product to exactly match your installation standards.

You can create your own application templates for subsequent use by your programmers or end users.

• The following topics deal with the mechanics of creating and maintaining application templates:

5.13.1 Invoke the Work With Application Templates Facility

5.13.2 Work with Application Template Definitions

5.13.3 Creating Application Templates

5.13.4 Review/Change Application Template Definition

5.13.5 Review/Change Application Template Commands

5.13.6 Maintain Template HELP Panels

5.13.7 Create a New HELP Panel

5.13.8 Review/Change a HELP Panel Definition5.13.9 Review/Change a HELP Panel Body

- For details of how an application template is invoked and then used by a programmer or an end user to generate an RDML program, refer to Creating a New Function and Editing a Function's RDML Commands.
- For details of how the application template commands are constructed and used, refer to Template Commands and Variables in the *Technical Reference Guide*.

5.13.1 Invoke the Work With Application Templates Facility

The steps involved in invoking the 'Work with application template definitions' facility can be visualized like this:



5.13.2 Work with Application Template Definitions

When the option to 'Work with Application Template definitions' is selected from the Housekeeping Menu the first screen presented looks something like this:

DC@P750101	Maintain Application Templates
Position to:	
Type options (and A	pplication Template) and press Enter
1=Create 2=Review/0 8=Review/Change H	Change definition 4=Delete 5=Review/Change commandelep panels 17=Last Action Details 33=Propagate
Option Template A	pplication Template Description
BASEPGM I BBIFE Con BBDOW C	Basic Program Layout struct an IF / ELSE / ENDIF program block onstruct a DOWHILE program block
Fnn=Help Fnn=Exi	t Fnn=Cancel Fnn=Messages Fnn=ChgSeq

Note that normal LANSA object security exists over application templates. This screen will only show those templates you are authorized to use. Security will also determine the options you are allowed to take for each application template.

Working from the 'Maintain Application Templates' screen, enter one of the following options beside a list entry:

1=Create: indicates that you wish to create a new application template. This action is only valid on the first input line displayed.

The application template name specified must be a valid IBM i object name as per the rules described in Valid Names on the IBM i in *Common Input Options*.

When option 1 for create is used, and if the new application template name is valid, a further screen will be displayed requesting that you enter further details. This screen is described in more detail later.

2=Review/Change definition: indicates that the definition of an application template is to be reviewed and possibly altered (if authorized to do so). This option is not valid on the first input line displayed.

When option 2 for review/change is used, a screen showing the detailed definition of the template will be displayed. This screen is described in more detail later.

4=Delete: deletes an existing application template (if authorized to do so). This option is not valid on the first input line displayed.

A confirmation of delete screen will be displayed, if you are authorized to delete the application template. Then the **Maintain Application Templates** screen will be re-displayed with the deleted application template omitted.

DC@P750108	Delete Application Template
Template name Description	: XXXXXXXXXX : XXXXXXXXXXXXXXXXXXXXX
Confirm this ten	nplate is to be deleted
Delete this templa	ate YES, NO
WARNING: Deletion	n of this template will remove it from ALL partitions.
Fnn=Help Fnn=Exit	Fnn=Cancel Fnn=Messages

5=Review/Change commands: indicates that the template and RDML commands of an application template are to be reviewed and possibly altered (if

authorized to do so). This option is not valid on the first input line displayed. When option 5 for review/change is used, the next screen will be displayed to review and/or change the template commands. This screen is described in more detail later.

8=Review/Change Help panels: indicates that the HELP panels associated with an application template are to be reviewed and possibly altered (if authorized to do so). This option is not valid on the first input line displayed.

When option 8 for review/change is used, the next screen will be displayed to review and/or change the Help panels. This screen is described in more detail later.

17=Last Action Details: indicates that you wish to review the last action details recorded for the application template.

Enter this option against any existing template shown on the display. A screen will be displayed showing the last action details. This screen is described in Online Access to Last Action Details.

33=Propagate: propogates the selected templates. For more information, refer to Propagating Objects from the IBM i

The *ChqSeq* (Change Sequence) function key will toggle the list display between display sequence and Template Name sequence. When the list is in template name sequence, a *Position to* facility is enabled.

5.13.3 Creating Application Templates

When option '1=Create' is chosen from the 'Maintain Application Templates' menu a screen similar to the following example is used to enter details of the new template:

DC@P750102 Create Application Template

Type new Application Template details and press Enter.

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The following input options apply when creating an application template from the 'Create Application Template' screen:

Identifier

Mandatory. The unique identifier of this new application template. Note that application templates are shared across all partitions, they are not unique in each partition.

Short Description

Mandatory. A short description of the application template. This will appear on all lists of templates.

Extended Description

Optional. A more detailed description of the application template. Used for documentation purposes only.

Display Sequence

Mandatory, but always prefilled to <highest sequence number + 100>. This sequence number determines the sequence of templates displayed in any list.

Copy from Application Template

Optional. The name of an existing application template that this template is to be copied from. This includes all HELP panels for the template being copied.

When you have completed all the new application template details, you will be returned to the 'Maintain Application Templates' Menu. You should then complete the remaining parts of the definition i.e. application template commands and HELP panels - refer to the following sections for further details.

5.13.4 Review/Change Application Template Definition

When option '2=Review/Change definition' is chosen from the 'Maintain Application Template' Menu a screen similar to the following example is used to review and/or change (if authorized) the template definition:

DC@P750103 Review/Change Definition

Template: BBIFConstruct an IF / ENDIF program block

Type changes to template details and press Enter

Short description Construct an IF / ENDIF program Extended description Mainly used by RDML programmers to construct an IF/ENDIF block in their programs Display sequence 100.00

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Working from the 'Review/Change Definition' display it is possible to:

- Review the application template definition details
- Change the application template definition. To do this use the CHANGE function key to place the screen in change mode and make any desired changes.

The following input options apply to changing an application template:

Short Description

A short description of the application template. This will appear on all lists of templates.

Extended Description

A more detailed description of the application template. Used for documentation

purposes only.

Display Sequence

This sequence number determines the sequence of templates displayed in any list that is visible to end users or programmers.

5.13.5 Review/Change Application Template Commands

When option '5=Review/Change commands' is chosen from the 'Maintain Application Template' Menu a screen similar to the following example is used to initiate the review and/or change (if authorized) of the commands that define the template:

DC@P750104 Edit template by IBM Editor

Template : CONIF Construct an IF / ENDIF program block

Specify details of any other **functions or application templates** that you wish to **browse** or **copy from** while using EDTSRC on this template :

Process/Template name Function name

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Working from the 'Edit template by IBM Editor' display it is possible to:

- Review the application commands that define the template.
- Change the commands that define the template.
- Indicate that details from other Templates or RDML Functions should also be copied into the source work area for subsequent browsing or copying.

This facility is virtually identical to that provided for the editing of RDML commands via the IBM supplied editor SEU.

Unless you are completely comfortable with the coding and construction of normal RDML programs and using the IBM editors to edit them, it is
recommended that you do not use this facility.

It is recommended that the IBM Source Edit Utility (SEU) be used for editing templates. However, if SEU is not available the template commands may be maintained using the Edit File command (EDTF).

The construction of application templates and the use of application template commands is described in Template Commands and Variables in the *Technical Reference Guide*. Make absolutely sure that you have read and understood all the relevant sections in this guide before attempting to use this facility.

5.13.6 Maintain Template HELP Panels

When option '8=Review/Change Help panels' is chosen from the 'Maintain Application Template' Menu a screen similar to the following example is used to initiate the review and/or change (if authorized) of the help panels associated with a template:

DC@P	750105 N	Aaintain Template HELP Panels	
Templa	ite : CONI	F Construct an IF / ENDIF program block	
Type options (and Help Panel Identifier) and press Enter. 1=Create 2=Review/Change definition 4=Delete 5=Review/Change Help panel			
Option	Panel Id	Help Panel Description	
- - -	HELP010 HELP020 HELP030	The IF / ENDIF Construct The IF / ENDIF Construct The IF / ENDIF Construct	
Fnn=He	elp Fnn=Exit Fi	nn=Cancel Fnn=Messages	

Working from the 'Maintain Template HELP Panels' display, enter one of the following options beside a list entry:

1=Create: indicates that you wish to create a new HELP panel. This action is only valid on the first input line displayed.

The HELP panel name specified must be a valid IBM i object name as per the rules described in Valid Names on the IBM i in *Common Input Options*.

When option 1 for create is used, and if the new HELP panel name is valid, a screen will be displayed requesting that you enter further details. This screen is

described in more detail later.

2=Review/Change definition: indicates that the definition of a HELP panel is to be reviewed and possibly altered (if authorized to do so). This option is not valid on the first input line displayed.

When option 2 for review/change is used, a screen showing the detailed definition of the HELP panel will be displayed. This screen is described in more detail later.

4=Delete: deletes an existing HELP panel (if authorized to do so). This option is not valid on the first input line displayed.

A confirmation of delete screen will be displayed (if you are authorized to delete the application template. Then the above screen will be re-displayed with the deleted application template help panel omitted.

DC@P750109	Delete Template HELP Panel
Template nameDescriptionHelp PanelDescription	: XXXXXXXXXX : XXXXXXXXXXXXXXXXXXXXX
Confirm this he ly	p panel is to be deleted
Delete this help p	anel YES, NO
WARNING: Deletion	of this template will remove it from ALL partitions.
Fnn=Help Fnn=Exit	Fnn=Cancel Fnn=Messages

5=Review/Change Help panel: indicates that full HELP panel is to be reviewed and possibly altered (if authorized to do so. This option is not valid on the first input line displayed.

When option 5 for review/change is used, the next screen will be displayed to review and/or change the actual HELP panel. This screen is described in more detail later.

5.13.7 Create a New HELP Panel

When option '1=Create' is chosen from the 'Maintain Template HELP Panels' menu a screen similar to the following example is used to enter details of the new HELP panel:

DC@P750106 Create Template HELP Panel

Template : BBIF Construct an IF / ENDIF program block

Type new Help Panel details and press Enter.

Identifier	SAMPLE0010
Description	. <u>Sample Help Panel</u>
Display sequence	200.00
Copy from HELP panel	•••••
in application template .	<u>CONIF</u>

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The following input options apply when creating a new HELP panel from the 'Create Template HELP Panel' screen:

Identifier

Mandatory. Specifies the identifier of the HELP panel. HELP panel identifiers must be unique within an application template and must conform to the IBM i naming standards defined in Valid Names on the IBM i in *Common Input Options*. The name specified is the name shown as the panel identifier when the HELP panel is presented.

Description

Mandatory. Specifies the title of the HELP panel when it is displayed to a user. It is recommended that upper and lower case characters are used and that the value is centered in the space available. This ensures the best aesthetic effect

when it is displayed to a user of the HELP panel.

Display Sequence

Mandatory, but always prefilled to <highest sequence number + 100>. This sequence number determines the sequence that HELP panels will be displayed to a user when they elect to 'walk through' all the HELP panels available for an application template.

Copy from HELP panel / in application template

Optional. The name of an existing HELP panel from any existing application template can be copied to create this new template.

When you have completed all the new HELP panel details, you will be returned to the 'Maintain Template HELP Panels' Menu.

You would normally then enter a '5=Review/Change HELP panel' request against the help panel you have just created. This allows you to review and/or change the actual HELP panel body.

5.13.8 Review/Change a HELP Panel Definition

When option '2=Review/Change definition' is chosen from the 'Maintain Template HELP Panels' Menu a screen similar to the following example is used to review and/or change (if authorized) the HELP panel definition:

DC@P75010	7 Rev	view/Change HELP Panel
Template	: CONIF	Construct an IF / ENDIF program block
Use Change k	ey to modify	HELP panel definition.
HELP panel HELP panel	Identifier description .	

Display sequence <u>100.00</u>

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Working from the 'Review/Change definition' display it is possible to:

- Review the HELP panel's definition
- Change the HELP panel's definition. To do this use the CHANGE function key to place the screen in change mode and make any desired changes.

The following input options apply to changing a HELP panel's definition:

Description

Mandatory. Specifies the title of the HELP panel when it is displayed to a user. It is recommended that upper and lower case characters are used and that the value is centered in the space available. This ensures the best aesthetic effect when it is displayed to a user of the HELP panel.

Display Sequence

Mandatory, but always prefilled to <highest sequence number + 100>. This

sequence number determines the sequence that HELP panels will be displayed to a user when they elect to 'walk through' all the HELP panels available for an application template.

5.13.9 Review/Change a HELP Panel Body

When option '5=Review/Change Help panel' is chosen from the 'Maintain Template HELP Panels' Menu, a screen will be presented to you that shows an exact representation of what the HELP panel will look like when presented to a user of the application template.

An example of what a HELP panel might look like follows, but of course, your HELP panel may look nothing like this example:

HELP010 The IF / ENDIF Construct

This application template constructs a simple IF / ENDIF construct into your RDML program. If you use it you will be asked 2 questions ...

- 1. To specify a comment about what the IF command will test for. Your reply should be brief and concise. For example : 'If customer is over the credit limit'
- 2. To specify the actual condition that is to be tested by the IF command. Construct your condition according to the rules defined in the Technical Reference Guide. Do NOT surround the condition with quotes. Some examples are :

#AMOUNT *GT #CRDLMT -> amount spent is greater than credit limit)

#CUSTYPE = A --> customer type is an A

(#CUSTYPE = A) *AND (#AMOUNT *GT #CRDLMT) --

> customer type is an

A and amount spent is greater than credit limit

((#A * #B) - #C) *GT 24.56 --> A times B minus C is greater than 24.56

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Working from the 'Review/Change Help panel' display it is possible to:

- Review the HELP panel body in exactly the form that it will be shown to a user of the application template.
- Change the HELP panel layout.

When creating or changing a HELP panel body, please note the following:

- The entire screen from lines 2 to 21 is input capable. To change the layout of the panel, enter the desired changes and press enter.
- The HELP panel body screen will continue to be re-displayed until you do not make any changes to the layout of the HELP panel.
- If you accidentally erase your entire HELP panel by using the field exit key, immediately use the MENU/CANCEL function key to return to the 'Maintain Template HELP Panels' menu. Elect to change the help panel body again, and it should be restored to what it was at the last time you pressed the enter key.
- The same 'special characters' supported for the input of LANSA field, process and function level HELP text can be used in your application template HELP panels:

Special Character	Effect
%(percentage)	Display in high intensity
{(left parenthesis)	Underline
@(at)	Display in reverse image
~(accent)	Blink
}(right parenthesis)	Revert to normal display mode
\(backslash)	Revert to normal display mode

Some examples of using these special characters are:

HELP Text Input Using SEU

Displayed By LANSA As

the%item number\must	the \$#item number#\$ must
the{item number}must	the \$@item number@\$ must
%{VALIDATION RULES\	\$%VALIDATION RULES%\$
%{VALIDATION RULES}	\$%VALIDATION RULES%\$
{%VALIDATION RULES}	\$%VALIDATION RULES%\$

Where multiple enhancement characters are used together with no spaces between them they are 'consolidated' into one enhancement character.

For instance '% $\{@$ ' (high intensity, underline, blink) would be consolidated into one enhancement character that is used in the same position as the '@'.

The '%' and '{' are replaced by blanks. This explains why in the last example above the underline ('{') attribute starts under the 'V' in 'VALIDATION' and not one character before it.

5.14 Impact Analysis

Impact Analysis is designed to give you a way of assessing the impact of changes on applications. It helps to find where LANSA application changes are to be made and the extent of those changes.

It is possible to search for LANSA objects which have particular features and to find the relationship of those objects to other LANSA objects. A LANSA object in this context is a Field, File, Function, System Variable or Multilingual Variable. A list of the LANSA objects found by these searches can be built up and held on an Impact List. Objects are searched for in the current partition.

The Impact List consists of a named list containing entries of LANSA objects. Each entry has associated 'why it was added to the list' reasons.

The Impact List can be used in the Work with Fields, Work with Files, Work with Functions, Work with System Variables and Work with Multilingual Variables panels. When an Impact List is used in these Work with panels only the objects on the Impact List are displayed. By this means it is possible to work from a subset of objects to make changes to the Repository. Refer to 'Selecting entries from the LANSA menus' for details.

The entries on an Impact List can be retrieved into an RDML function by use of the special Built-In Function GET_ILENTRY_LIST.

5.14.1 Work with Impact Lists

The 'Work with Impact Lists' option is on the Housekeeping Menu. The initial display for the 'Work with Impact Lists' option will show existing Impact Lists and description.

DC@P410001	Work with Impact	t Lists	
Type options and p	press enter		
3=Copy to Export	List 4=Delete	5=Review	
6=Print 7=Copy T	askID 17=Last Actio	n Details 32/33=Merge	entries/From
	• ,•		

Opt	List	Description
	CR1234	Change Req 1234 - amend discount calculation
	CR2200	Change Req. 2200 - new branch
	FSIAMT	Fred's invoice amount investigations

Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Cancel Fnn=Messages

A new Impact List can be created by using the Add/Create function key. A screen requesting the input of a name and description will be displayed.

The List name must be a valid IBM i name. A brief description of the list must be specified.

Options

Options available against a list on the 'Work with Impact Lists' screen are:

3=Copy to Export List: This option will copy the entries on the Impact List to an Export List of the same name. If the Export List does not exist, one will be created. If the Export List does exist, the existing export list entries will be overwritten. This copy requires exclusive use of both lists. You may specify the Target System Type for the export list (e.g. AS/400, WIN). A screen requesting

confirmation of the copy request is displayed before the copy is performed. The Export List can be worked with using the normal 'Work with list of objects to be exported' option on the Housekeeping Menu.

4=Delete: This option will delete the Impact List and all its entries and associated reasons. A screen requesting confirmation of the request will be displayed before the delete is performed. This action requires exclusive use of the list.

5=Review: This option will display the list entries. The 'Work with List Entries' screen enables entries to be added and deleted. Refer to **5.14.3** Work with **Impact List Entries** for details. This action requires exclusive use of the list.

6=Print: This option submits a batch job to print the Impact List. The list entries are printed. The reasons associated with each entry may optionally be printed.

7=Copy TaskID: This option allows you to select TaskIDs. Objects in the current partition which have been worked on under the selected Task Ids are added to the impact list. Refer to 5.14.2 Select Task Identifier(s) for details. This action requires exclusive use of the Impact List.

17=Last Action Details: This option will display the Last Action Details for the Impact List. This will show when the list was created and last modified.

32/33=Merge entries/From: 32 must be entered against one list and 33 against a second list. The details from the 33 list will be merged with details on the 32 list. The 33 list will be unchanged. A screen requesting confirmation of the request will be displayed.

5.14.2 Select Task Identifier(s)

When Option 7 'Copy TaskID' is used against an entry on the 'Work with List Entries' screen, a screen similar to the following example is used to display a list of task identifiers that you are authorized to work with.

DC@P410018 Select Task Identifier(s)

Select a Task Identifier(s), so that all objects in current partition worked on under the Task Identifier will be automatically added to list

Sel Task ID Description

- _ 00000001 Design A/C module database Ref. 0330
- _ 00000002 User change request Ref. 0331
- _ 00000003 Implement new A/C module Ref. 0332
- _ 00000004 User change request Ref. 0333

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

This facility allows you to add all objects (LANSA fields, files, processes, functions, system variables and multilingual variables) for the current partitions that have been worked on under the selected task identifier(s) to the impact list.

If a process object exists on the selected task identifier all functions in that process will be added to the impact list.

Once all objects have been added to the impact list the 'Work with List Entries' display will be shown with the added objects highlighted.

The added objects will have an associated 'why it was added ' reason . This reason will contain the originating task identifier.

5.14.3 Work with Impact List Entries

When a Review of an Impact List is requested a screen showing existing entries in the list will be displayed. This will lock the Impact list for exclusive use.

DC@P410007 Work with List Entries Position to ...____ List : CR1234 Change Req 1234 - amend discount calculation Type options and press enter 2=Reasons 4=Drop Entry 6=Add Note 8=Details 24=Related Related: 31-Field, 32-File, 33-Fun, 34-SV, 35-MT Opt Type Object Description LIST List Level Information FIELD DISCAMT **Discount Amount** FIELD DISPCNT **Discount Percent** FILE INVDET INVLIB Invoice Details MT *MTXTDISCMSG Discount of 10% if paid by

FUN INPR1 INF105 Calculate outstanding

FUN INPR5 INF507 Refunds

____ SV *DISCDEFAULT Discount default Percent

Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Details Fnn=Cancel Fnn=Messages Fnn=Find Fnn=Save

The entries displayed can be:

LIST: A list level entry. The result of batch searches are held as reasons against this entry. This entry is automatically created for a list and may not be deleted. **FIELD:** A field. The field name and description are shown.

FILE: A file. The file name and library and the file description are shown.

MT: A multilingual text variable. The variable name and the value in the current language are displayed.

FUN: A function. The Process name and Function name and the function description are shown.

SV: A system variable. The variable name and description are shown.

Function keys

Add/Create, Details, Find and Save functions exist on the 'Work with List Entries' screen, along with the standard function keys.

Add/Create function key: Use of this function key initiates a Profile Search which adds the found objects to the list of entries. A Profile Search looks for objects with particular features.

For example

Fields with the word 'date' in the description and a length = 6

or Fields with a name starting AMT

or Files in library INVLIB and containing the word 'due' in the description

See 5.14.4 Impact List - Profile Search for a detailed description of this feature.

Details function key: The cursor must be positioned on an entry in the list and this function key pressed. The details of that object will be displayed. Refer to 5.14.7 Display Details of Object/Reason for further information. These details may also be displayed by using option 8 against the list entry.

Find function key: Use of this function will initiate a search on the entries in the Impact List. Entries which meet the criteria specified will be highlighted on the 'Work with List Entries' display. Refer to 5.14.8 Find Impact List Entries for further information.

Save function key: Use of this function will save the current contents of the list. The list entries and reasons are built up and displayed as a working list. The changes are not committed until this Save is used or they are saved on exit from the 'Work with List Entries' display.

Options

Options available against a particular entry on the 'Work with List Entries' screen are:

2=Reasons: This option will display the reasons why this entry was added to the list and any user added Notes. The 'Reasons for an Entry' screen will be displayed.

4=Drop Entry: This option will flag the particular entry as ready to be deleted. It will no longer be shown on the list of entries. Actual deletion will be done when the list entries are saved by use of the Save function key or saved on exit

from the 'Work with List Entries' display.

6=Add Note: This option will display a screen which will allow the user to add his own reason against the entry. The Note is 65 characters of free form text. This note will be shown along with other reasons associated with the entry when the 'Reasons for an Entry' is requested.

8=Details: This option will display details of this entry. Refer to 5.14.7 Display Details of Object/Reason for further information.

24=Related: This requests a related search be performed against this entry. The Related search will find other objects which relate to this entry and add them to the list of entries.

For example, if the entry is a FIELD and related Functions are requested, the related search will find

- functions which contain the Field
- the prompt function for the Field
- functions used in Rules / Triggers for the Field

and add these functions to the list of entries.

Refer to 5.14.5 Impact List - Related Search for details.

Related: 31-Field, 32-File, 33-Fun, 34-SV, 35-MT: These options enable you to quickly request a Related Search for a particular type of object. These Related Searches will run interactively.

Position to.....

This consists of Type and Object values which can be used to position the display of entries to a page containing the values entered. Values for Type are FIELD, FILE, MT, FUN, SV.

For example

- Position toFIELD DAT will display the page of entries that contains the first field equal to or greater than DAT.
- Position toFUN ABC will display the page of entries that contains the functions in process ABC.

5.14.4 Impact List - Profile Search

When the Add/Create function key is used on the 'Work with List Entries' screen a Profile Search is initiated. A Profile Search looks for objects with particular attributes. The objects found are added to the entries in the list. The search may be run in batch or interactively. If run interactively the list is redisplayed with the added entries highlighted.

To perform a Profile Search you must:

- 1. Select the type of object to be searched for.
- 2. Enter the characteristics to be searched for and the Run in Batch option.

Step 1: Select the type of object to be searched for

The type of object to be searched for is selected from the box presented when the Add/Create function key is used.



Step 2: Enter the characteristics to be searched for

The characteristics to be searched for and the Run in Batch option are entered on a screen of possible attributes. This screen is presented after the selection of the type of object and will vary according to the type selected.

• The search criteria is entered as an Operation (such as =) and a Value.

- If an Operation is entered then Value must also be entered for that attribute.
- If a Value is entered then the corresponding Operation must be entered.
- When several Operations and Values are entered objects are searched for which match all the criteria.

For example:

Field Type	EQ A
Length	EQ 6

Default value EQ *YYMMDDC

Only fields which are alpha with a length of 6 and a default value of ***YYMMDDC** will match and be added to the list.

It is possible to enter a list of values for some attributes. That attribute is considered matched if it is any of the values in the list.

For example:

Field Type	EQ	А
Length	EQ	6
Default value	LS	*YYMMDDC *DDMMYYC

The fields which match and are added to the list will be alpha with a length of 6 and have a default value. That default value will be *YYMMDDC or *DDMMYYC.

It is possible to search on the Last Action Details date and action.

For example:

File Name EQ EMP

Last Action LT 19970518 CM (YYYYMMDD Action)

The files which match and are added to the list will start with 'EMP' and were last compiled before 18th May 1997.

If no Action is specified in the Last Action Details search all actions are applicable

For example:

File Name EQ EMP

Last Action GT 19970430 __ (YYYYMMDD Action)

The files which match and are added to the list will start with 'EMP' and were last actioned in some way after 30th April 1997.

Operations

EQ	equal to
=	equal to
NE	not equal to
GE	greater than or equal to
>=	greater than or equal to
GT	greater than
>	greater than
LE	less than or equal to
<=	less than or equal to
LT	less than
<	less than
IN	Contains
LS	List
LK	Like (SQL pattern)

Not all Operations are valid for some attributes. See individual attributes for valid Operations. The testing for an IN (contains) operation is not case sensitive. The testing for EQ, = and NE operations is case sensitive. The value specified with an IN (contains) operation should be left aligned. Do not enclose values in quotes unless quotes are part of the value. Where multiple values can be specified for an attribute but an EQ, = or NE operation is used, the value should be entered in the first value position. Values for Description and Help Text plus Label and Column Heading for fields are assumed to be in the language currently being used in the partition and only the current language contents of these attributes are searched.

The LK (Like) operation uses the full value field in an SQL WHERE command.

This means trailing blanks are significant. If you do not wish to match trailing blanks, you must fill the value with % (e.g., Field name LK %MONTH%%%%).

Examples of LK operation:

- Field name LK __MM__ This finds fields DDMMYY, YYMMDD. It does not find field DDMMYYC because the 4 trailing blanks in the value are included in the matching pattern.
- Field name LK __MM%%%%% This finds fields DDMMYY, DDMMYYC, DDMMYYYY, DDMMYYYYC, YYMMDD, YYMMDDC.

Select Fields for Profile Search

When you select Fields by Profile Search this screen is displayed:

DC@P410008 Add FIELD Entry by Profile Search
Run in Batch YES Enter Operation and Value/s for search criteria Field name
Label Col Hdg 1 Col Hdg 2 Col Hdg 3 O/P attribute Col Hdg 3 I/P attribute Edit word Edit code Edit word Default Col Hdg 3
Alias name

The Run in Batch option can be YES or NO. For details of Run in Batch see 5.14.9 Run Impact List searches in Batch.

Entries for Field Profile search:

Attribute	Operation	Value

Field name	EQ,= ,NE,blank LK	blank, full or partial name Pattern example: %DATE%%%%% MM	
Description	EQ,= ,NE,IN, blank		
Field type	EQ,= ,NE,blank		
Length	EQ,= ,GT,> , GE,>=,LT,< , LE, <=,NE,blank	must be numeric or blank	
Decimals	EQ,= ,GT,> , GE,>=,LT,< , LE, <=,NE,blank	must be numeric or blank	
Label	EQ,= ,NE,IN, blank		
Col Hdg 1	EQ,= ,NE,IN, blank		
Col Hdg 2	EQ,= ,NE,IN, blank		
Col Hdg 3	EQ,= ,NE,IN, blank		
Reference Field	EQ,= ,NE,LS, blank		
O/P attribute	EQ,=		

	,NE,LS, blank	
I/P attribute	EQ,= ,NE,LS, blank	
Edit code	EQ,= ,NE,blank	
Edit word	EQ,= ,NE,blank	
Default value	EQ,= ,NE,LS, blank	
Alias name	EQ,= ,NE,blank	
System field	EQ,= ,NE,blank	blank, YES, NO
_	FO	
Prompt Pro/Fun	EQ,= ,NE,blank	The value consists of two parts, a Process and a Function. Possible combinations are : both blank (no search), both entered (search on Process and Function), Process entered with blank Function (search on Process only), Function entered with blank Process (search on Function only).
Prompt Pro/Fun Help Text	EQ,= ,NE,blank EQ,= ,IN, blank	The value consists of two parts, a Process and a Function. Possible combinations are : both blank (no search), both entered (search on Process and Function), Process entered with blank Function (search on Process only), Function entered with blank Process (search on Function only).
Prompt Pro/Fun Help Text Rule/Trigger	EQ,= ,NE,blank EQ,= ,IN, blank EQ,= ,NE,LS, blank	The value consists of two parts, a Process and a Function. Possible combinations are : both blank (no search), both entered (search on Process and Function), Process entered with blank Function (search on Process only), Function entered with blank Process (search on Function only). blank, RV(range of values), LV(list of values), CF(file lookup), SL(evaluate expression), CL(call program), DC(date check), TR(trigger)

action), date entered with blank action (search on date and any action).
Possible values for action: blank,
AE(exported to IBM i), CI(checked in),
CM(compiled), CO(checked out),
CR(created), DI(Deliver To), IM(imported),
MD(modified), WE(exported to
workstation), MC (modified by checkin), MI
(modified by import).

Select Files for Profile Search

Γ

When you select Files by Profile Search this screen is displayed:

DC@P410009 Add FILE Entry by Profile Search
Run in Batch YES Enter Operation and Value/s for search criteria File name
HST Maintained by
Record format Name I/O module Lib
Commitment AUTOCOMMIT parameter
ShareSecureStrip DebugSuppress IOM0034 MsgI/O module createIgnore decimal errorALTSEQCreate Batch Ctl RcdDB Trigger usedDB Trigger Program
CRTPF options File Rule/Trigger Access Route file Batch Control file Versions different Last Action (XXXXMMDD Action)
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The Run in Batch option can be YES or NO. For details of Run in Batch see 5.14.9 Run Impact List searches in Batch.

Entries for File Profile search:

Attribute	Operation	Value

File name	EQ,= ,NE,blank	blank, full or partial name
Library	EQ,= ,NE,blank	
Description	EQ,= ,NE,IN, blank	
HST (High Speed Table)	EQ,= ,NE,blank	blank, YES, NO
Maintained by	EQ,= ,NE,blank	blank, LANSA, OTHER
Record Format Name	EQ,= ,NE,blank	
I/O module Lib	EQ,= ,NE,blank	blank, F, M
Commitment	EQ,= ,NE,blank	blank, YES, NO
AUTOCOMMIT parameter	EQ,= ,NE,blank	blank, YES, NO
Share	EQ,=,NE,blank	blank, YES, NO
Secure	EQ,=,NE,blank	blank, YES, NO
Strip Debug	EQ,=,NE,blank	blank, YES, NO
Suppress EQ,=,NE,blank		blank, YES, NO
I/O module EQ,=,NE,blank create		blank, YES, NO
Ignore decimal error	EQ,=,NE,blank	blank, YES, NO
ALTSEQ	EQ,=,NE,blank	
Create Batch Ctl EQ,= Rcd ,NE,blank		blank, YES, NO

DB Trigger used	EQ, =, NE,blank	blank, Yes, No
DB trigger program	EQ, = , NE,blank	blank, full or partial name
CRTPF options	EQ,=,NE,IN, blank	
File Rule/Trigger	EQ,=,NE,LS, blank	blank, RV(range of values), LV(list of values), CF(file lookup), SL(evaluate expression), CL(call program), DC(date check), TR(trigger)
Access Route file	EQ,=,NE,blank	blank, full or partial name
Batch Control file	EQ,=,NE,blank	blank, full or partial name
Versions different	EQ,=,NE,blank	blank, Y or N This checks the active and latest versions of a file. EQ Y will find files whose active and latest versions differ. EQ N will find files whose active version = latest version.(i.e. compiled files)
Last Action	EQ,=,GT,> , GE,>=,LT,< , LE, <=,NE,blank	The value consists of two parts, a date in YYYYMMDD format and an action. Possible combinations are: both blank (no search), both entered (search on date and action), date entered with blank action (search on date and any action). Possible values for action: blank, AE(exported to IBM i), CI(checked in), CM(compiled), CO(checked out), CR(created), DI(Deliver To), IM(imported), MD(modified), WE(exported to workstation), MC

		(modified by checkin), MI (modified by import).
--	--	---

Note: Only the active and latest versions of files are searched. Both active and latest versions of a file may match the search criteria and a reason for both versions will be attached to the entry.

Select Functions for Profile Search

When you select Functions for Profile Search, this screen is displayed:

DC@P410009 Add FUNCTION Entry by Profile Search
Run in Batch YESEnter Operation and Value/s for search criteriaType of FunctionA (A=All, C=Compiled, N=Not compiled)
Process name Process Description Menu Style Optimise comm GUI enabled WEB enabled XML enabled Function name Function Description
RDML Code
Help Text
Last Action (YYYYMMDD Action)
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The Run in Batch option can be YES or NO. For details of Run in Batch see 5.14.9 Run Impact List searches in Batch.

Entries for Function profile search:

Attribute	Operation	Value
Type of Function		A - all Functions will be searched C - only compiled functions will be searched

		will be searched
Process name	EQ,= ,NE,blank	blank, full or partial name
Process Description	EQ,= ,NE,IN, blank	
Menu Style	EQ,= ,NE,blank	
Optimize comm	EQ,= ,NE,blank	blank, Y or N
GUI enabled	EQ,= ,NE,blank	blank, Y or N
WEB enabled	EQ,= ,NE, blank	blank, Y or N
XML enabled	EQ,= ,NE, blank	blank, Y or N
Function Name	EQ,= ,NE,blank	blank, full or partial name
Function Description	EQ,= ,NE,IN, blank	
RDML Code	IN,blank	
Help Text	EQ,= ,IN, blank	
In Help Text type		A - all Help text will be searched P - only Process Help text will be searched F - only Function Help text will be searched
Last Action	EQ,= ,GT,> , GE,>=,LT,<	The value consists of two parts, a date in YYYYMMDD format and an action. Possible

, LE, <=,NE,blank	combinations are: both blank (no search), both entered (search on date and action), date entered with blank action (search on date and any action).
	Possible values for action: blank, AE(exported to IBM i), CI(checked in), CM(compiled), CO(checked out), CR(created), DI(Deliver To), IM(imported), MD(modified), WE(exported to workstation), MC (modified by checkin), MI (modified by import).

Select SV Entry for Profile Search

When you select SV Entry for Profile Search, this screen is displayed:

DC@P410011 Add SV Entry by Profile Search				
Run in Batch YESEnter Operation and Value/s for search criteriaVariable Name				
Data type Length Decimals				
Set by calling Type				
Last Action (YYYYMMDD Action)				
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages				

The Run in Batch option can be YES or NO. For details of Run in Batch see 5.14.9 Run Impact List searches in Batch.

Entries for System Variable profile search:

Attribute	Operation	Value
Variable Name	EQ,= ,NE,blank LK	blank, full or partial name Pattern example: *LW3BL%LW%%%%%%%%%%%%%%
Description	EQ,= ,NE,IN, blank	
Method of	EQ,=	blank, STATIC or DYNAMIC

derivation	,NE,blank	
Data type	EQ,= ,NE,blank	blank, ALPHA or NUMBER
Length	EQ,= ,GT,> , GE,>=,LT,< , LE, <=,NE,blank	must be numeric or blank
Decimals	EQ,= ,GT,> , GE,>=,LT,< , LE, <=,NE,blank	must be numeric or blank
Set by	EQ,= ,NE,blank	enter value for both Set by and Type or either or neither.
Туре		blank, 3GL or FUN If a value is entered for Set by and Type is blank, the type is assumed to be 3GL.
Last Action	EQ,= ,GT,> , GE,>=,LT,< , LE, <=,NE,blank	The value consists of two parts, a date in YYYYMMDD format and an action. Possible combinations are: both blank (no search), both entered (search on date and action), date entered with blank action (search on date and any action).
		Possible values for action: blank, AE(exported to IBM i), CI(checked in), CM(compiled), CO(checked out), CR(created), DI(Deliver To), IM(imported), MD(modified), WE(exported to workstation), MC (modified by checkin), MI (modified by import).

Note : System Variables are system-wide objects while all the other objects dealt with by Impact Analysis are partition-wide. Consequently care should be taken when changes to System Variables are being considered since they may be

used in other partitions and that use will not be reported by Impact Analysis within the current partition.
Select Multilingual Variables for Profile Search

When you select Multilingual Variable by Profile Search, this screen is displayed:

DC@P410012 Add MT Entry by Profile Search
Run in Batch YESEnter Operation and Value/s for search criteriaVariable Name
Length
Language
Text
Last Action (YYYYMMDD Action)
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

The Run in Batch option can be YES or NO. For details of Run in Batch see 5.14.9 Run Impact List searches in Batch.

Entries for Multilingual variable profile search:

	0	_
Attribute	Operation	Value
Variable Name	EQ,= ,NE,blank LK	blank, full or partial name Pattern example: *MTXT%COMED%%%%%%%%%%
Length	EQ,= ,GT,> , GE,>=,LT,< , LE,	must be numeric or blank

	<=,NE,blank	
Language	EQ,= ,NE,blank	
Text	EQ,= ,NE,IN, blank	If no language is specified, all languages are searched.
Last Action	EQ,= ,GT,> , GE,>=,LT,< , LE, <=,NE,blank	The value consists of two parts, a date in YYYYMMDD format and an action. Possible combinations are: both blank (no search), both entered (search on date and action), date entered with blank action (search on date and any action). Possible values for action: blank, AE(exported to IBM i), CI(checked in), CM(compiled), CO(checked out), CR(created), DI(Deliver To), IM(imported), MD(modified), WE(exported to workstation), MC (modified by checkin), MI (modified by import).

Select Changed Objects for Profile Search

E

The *Add Changed Object* screen is displayed when an *Add changed objects* is selected from the *Add Entry by Profile Search* pop-up. This search enables you to find all objects that have been changed in some way since a particular date. This is achieved by searching for the relevant object type with the Last Action date and a list of relevant actions.

DC@P410020 Add Changed Objects			
Run in Batch YESEnter Operation and Value/s for search criteriaObject Name			
Object type X object types required Field File Function System Variable Multilingual Variable			
Last Action (YYYYMMDD) (Action)			
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages			

The *Run in Batch* option can be YES or NO. For details of running in batch, refer to 5.14.9 Run Impact List searches in Batch.

Entries for Changed Object search:

Attribute	Operation	Value
Object	EQ, + , NE,	blank, full or partial name

Name	blank	
Object type		At least one of the listed object types must be selected.
Last Action	EQ,= , GT, > , GE, >=, LT, < , LE, <=, blank	The value consists of two parts, a date in YYYYMMDD format and an Action. A date must be entered. From zero to 10 Actions may be entered. Objects with Last Action matching any of the supplied actions will be found and returned. Possible values for action are: blank, AE(exported to IBM i), CI(checked in), CM(compiled), CO(checked out), CR(created), DI(Deliver To), IM(imported), MD(modified), WE(exported to workstation), MC(modified checkin), MI (modified import).

5.14.5 Impact List - Related Search

When Option 24 is used against an Entry on the 'Work with List Entries' screen a Related Search is initiated. A Related Search looks for objects which have a connection to the current object entry. The objects found are added to the entries in the list. Each entry added will have a 'why it was added' Reason attached. This reason will show the relationship that exists. The search may be run in batch or interactively. If run interactively the list is redisplayed with the added entries highlighted.

By using the Related search a developer can find interdependencies of objects and use this information when planning a change to a LANSA system.

To perform a Related Search you must:

- 1. Enter YES or NO for Run in Batch
- 2. Select the type of related objects to be searched for.

Select the type of Related object to be searched for

The type of Related object to be searched for is selected from the box presented when Option 24 is entered against an entry.

Related Search :
FIELD DISCAMT :
Run in Batch YES YES/NO :
Use cursor to make a selection :
Add Related Fields :
Add Related Files :
Add Related Functions :
Add Related System Variables :
Add Related Multilingual Variables :
Cancel Add Request :
Fnn=Help Fnn=Cancel Fnn=Messages

:....:

The Run in Batch option can be YES or NO. For details of Run in Batch see 5.14.9 Run Impact List searches in Batch.

The Related Search may also be initiated by Options 31-35 against an entry on the Work with List Entries screen.

Option action:

31-requests related Fields to be searched for.

32-requests related Files to be searched for.

33-requests related Functions to be searched for.

34-requests related System Variables to be searched for.

35-requests related Multilingual Variables to be searched for.

These searches will be run interactively.

Note 1: System Variables are system wide objects while all the other objects dealt with by Impact Analysis are partition wide. Consequently care should be taken when changes to System Variables are being considered since they may be used in other partitions and that use will not be reported by Impact Analysis within the current partition.

Note 2: Only the active and latest versions of files are searched. Both active and latest versions of a file may relate to the object and a reason for both versions will be attached to the entry.

5.14.6 Reasons for an Entry

When Option 2 is used against an Entry on the 'Work with List Entries' screen, the associated 'why it was added' Reasons and any User Notes are displayed. By examining the Reasons displayed you can determine if the object is relevant to your investigations.

: Reasons for an Entry :
: FILE INVDET INVLIB Invoice Details :
: FIELD DISCAMT is on this FILE :
: FIELD DISCAMT is a key to a logical(INVDETV1) of this FILE :
:
: Cursor and F8 for detail. Cursor and F11 to drop. ENTER to Resume. :
:
:
:
:

Further details are available by positioning the cursor on a Reason and pressing function key 8.

By positioning the cursor on a reason and pressing function key 11, a reason will be flagged as ready to be deleted. It will no longer be shown on the reasons display. Actual deletion will be done when the list Entries are saved by use of the Save function key or saved on exit from the 'Work with List Entries' display.

5.14.7 Display Details of Object/Reason

The display of Details will occur when cursor position and function key are used on the 'Work with List Entries' screen for details of an Object or on the 'Reasons for an Entry' screen for details of a reason. The details are also displayed when Option 8 is used against a list entry.

DC@P410014 Display Details of Object/Reason

List : CR12	34 Change Req 1234 - am	nend discount calculation		
FILE INVDET INVLIB Invoice Details				
FIELD DISC	CAMT is on this FILE			
Find				
*Fields on Fi	le : INVDET INVLIB	Version No. 8		
CUSNO	Customer Number	Key 1 A 10		
INVNO	Invoice Number	Key 2 A 10		
INVTOT	Invoice total	P 11 2		
DISCAMT	Discount Amount	P 92		
DATEND	Customer Number	S 80		
GLPER	General Ledger period	S 20		

Fnn=Help Fn=Start Fn=End Fnn=Cancel Fnn=Messages Fnn=F/Fwd Fnn=F/Bkwd

The details shown will vary according to what details were requested.

A scan facility is provided on this screen. It is similar to that provided in the RDML editor. The display can be positioned to the start or end. A string can be entered in the Find.... value and scanned for forwards or backwards. Wild characters of ? are permitted within the string.

5.14.8 Find Impact List Entries

The display of "Find Impact List Entries" screen will occur when a function key is used on the 'Work with List Entries' screen. The find criteria is entered on the screen and the 'Work with List Entries' screen is redisplayed with the entries which meet the criteria highlighted.

This can be used as a check to ensure all objects on an impact list have been actioned. For example, an impact list of files and functions which require changing is created. The changes are made. Then this feature can be used to check all objects have been modified and compiled.



The criteria is entered as an Operation (such as =) and a value.

Attribute	Operation	Value	
Last Action	EQ,= ,GT,> , GE,>=,LT, < , LE, <=,blank	The value consists of two parts, a date in YYYYMMDD format and an action. Possible combinations are: both blank (no search), both entered (search on date and action), date entered with blank action (search on date and any action).	

Criteria for Find Impact List Entries search:

	Possible values for action: blank, AE(exported to IBM i), CI(checked in), CM(compiled), CO(checked out), CR(created), DI(Deliver To), IM(imported), MD(modified), WE(exported to workstation).
--	--

5.14.9 Run Impact List searches in Batch

Profile and Related searches can be run in batch. The Run in Batch option available on these searches is YES or NO. YES will run the search in batch and NO will run the search interactively. YES will display a screen on which can be specified the job control values.

It is advisable to run searches in batch if the LANSA partition contains a large number of objects or large RDML functions. The searches can be run interactively but may take some time to complete. Also interactive searches will only return up to 10000 matches. There is no limit on the number of matches for batch searches.

Batch searches require exclusive use of an Impact List. Therefore the Impact List should not be reviewed or locked in any way when the batch job is expected to run.

A message with the number of matches found by batch searches will be added to the LIST object as a Reason. This message will also contain the Job Name that was used on the batch job. Therefore the use of unique and meaningful job names will help in identifying the results of several searches.

5.14.10 Considerations when using Impact Lists

Any number of entries and reasons can be held on an Impact List but it is not advisable to build very large lists since the time taken to store and manipulate entries for display in 'Work with List Entries' will be lengthy. If large numbers of entries must be dealt with, consideration should be given to the technique of building several smaller lists to work with and then merging the lists when entries have been finalized.

Only the first 10000 entries will be displayed in the 'Work with List Entries'. All entries in a list can be printed.

System Variables are system wide objects while all the other objects dealt with by Impact Analysis are partition wide. Consequently care should be taken when changes to System Variables are being considered since they may be used in other partitions and that use will not be reported by Impact Analysis within the current partition.

Impact analysis examines objects and relationships at the time of running. Therefore if changes are made to the LANSA repository after running, the objects and relationships may no longer exist and details of reasons and objects may be unavailable for displaying within Impact analysis.

Virtual Field Derivation Code is only examined to establish what fields are being used and reported as a Field to File and a File to Field relationship. The code should be viewed to understand its effect.

The relationship of functions to field and files is established by using crossreference files built up when a function is compiled. Thus the relationships found by Impact Analysis for functions to fields, functions to files, field to functions and files to functions will be those that exist for compiled functions. This may differ from the RDML code if the code has been changed since the function was compiled.

No cross-reference files for variable use in functions exist. Consequently a lengthy scan of RDML code must be done to establish function to system variable, function to multilingual variable, system variable to function and multilingual variable to function relationships. If the RDML code has been changed since the function was compiled, the use of variables in the compiled function may be different.

If the partition has been enabled for RDMLX, it is recommended that the crossreferencing facilities in Visual LANSA be used to examine relationships between objects. Profile searches in Impact Analysis may still be used to establish the existence of an object.

5.15 The Multilingual Development Textual Data Facility

LANSA internal use Only.

5.16 Common Input Options

This chapter details the input options for various fields that are used throughout the LANSA system. They are included here to save repeating the information throughout this guide.

The options covered are:

5.16.1 Submit This Job?

5.16.2 Job Name

5.16.3 Job Description and Library

5.16.4 Job Queue And Library

5.16.5 Output Queue and Library

5.16.6 Valid Names on the IBM i

5.16.1 Submit This Job?

Requests that the submission of some batch job(s) be confirmed by entering YES.

If YES is not entered, or anything other than YES is entered, the batch job will not be submitted. In this case the previous menu is re-displayed with a message indicating that the job was NOT submitted.

If YES is entered the batch job is submitted and the previous menu re-displayed with a message indicating that the job was successfully submitted. This message also indicates the job name, user profile and job number of the job submitted. Also indicated is the job queue to which the job was submitted.

5.16.2 Job Name

Requests the name of the job to be submitted.

It is usually pre-filled with an appropriate name and does not have to be changed.

If changed, the new name entered must be a valid job name.

5.16.3 Job Description and Library

Requests the name (and optionally the library of residence) of the job description that is to be used when a batch job is submitted.

Job descriptions are a native part of the IBM i operating systems and are required whenever a batch job is submitted. Refer to the appropriate IBM manuals for more details.

If you specify a job description name only (i.e. no specific library), then the special value *LIBL is used for the library name. *LIBL indicates that the current job's library list should be searched for the first occurrence of a job description with the specified name.

Library name is specified in the format:

<LIBRARY NAME>/<JOB DESCRIPTION NAME>

For example QGPL/QBATCH specifies that job description QBATCH in library QGPL should be used.

The special value *USRPRF can be used on the IBM i only as the job description name. This indicates to LANSA that the job description associated with your IBM i user profile should be used.

Once you have specified a job description LANSA checks that the job description exists and that you are authorized to use it. If either of these checks fail the screen will be re-displayed with an error message indicating the cause of the problem.

5.16.4 Job Queue And Library

Requests the name (and optionally the library of residence) of the job queue onto which a batch job is to be submitted.

Job queues are a native part of the IBM i operating system and all batch jobs must be submitted onto a job queue. Refer to the appropriate IBM manuals for more details.

If you specify a job queue name only (i.e. no specific library), then the special value *LIBL is used for the library name. *LIBL indicates that the current job's library list should be searched for the first occurrence of a job queue with the specified name.

Library name is specified in the format:

<LIBRARY NAME>/<JOB QUEUE NAME>

For example QGPL/QBATCHL specifies that job queue QBATCHL in library QGPL should be used.

The special value *JOBD can be used on the IBM i as the job queue name. This indicates to LANSA that the job queue associated with the job description you have nominated elsewhere on the screen should be used.

Once you have specified a job queue LANSA checks that the job queue exists and that you are authorized to use it. If either of these checks fail the screen will be re-displayed with an error message indicating the cause of the problem.

5.16.5 Output Queue and Library

Requests the name (and optionally the library of residence) of the output queue onto which any output (reports/printout) produced by the batch job is to be placed.

Output queues are a native part of the IBM i operating systems and all batch jobs must be submitted onto a job queue. Refer to the appropriate IBM manuals for more details.

If you specify an output queue name only (i.e. no specific library), then the special value *LIBL is used for the library name. *LIBL indicates that the current job's library list should be searched for the first occurrence of an output queue with the specified name.

Library name is specified in the format:

<LIBRARY NAME>/<OUTPUT QUEUE NAME>

For example QGPL/QPRINT specifies that output queue QPRINT in library QGPL should be used.

The special value *JOBD can be used on the IBM i as the output queue name. This indicates to LANSA that the output queue associated with the job description you have nominated elsewhere on the screen should be used.

Additionally the following special values can be specified:

- *DEV, which indicates that the output queue associated with printer device nominated in the PRTDEV parameter of the nominated job description should be used.
- *USRPRF, which indicates that the output queue associated with the user profile under whose profile the batch job runs is to be used. Normally a batch job runs under the user profile of the person who originally submitted it.
- *CURRENT, which indicates that the output queue currently being used by the submitting job (i.e. the workstation job) should be used by the submitted job.

For more information about these special values refer to the appropriate IBM supplied manual.

Once you have specified an output queue LANSA checks that the output queue exists and that you are authorized to use it. If either of these checks fail the screen will be re-displayed with an error message indicating the cause of the problem.

5.16.6 Valid Names on the IBM i

All "objects" (files, programs, etc) created on an IBM i must have a name. Any object name must conform to the following rules:

- The first character of the name must be in the range A -> Z or one of @, # or \$.
- Subsequent characters must be in the range A -> Z, or in the range 0 -> 9, or one of @, #, \$ or _. This rule excludes imbedded blanks from any object name.
- The name must not be longer than 10 characters.

6. Administration

To display the *Administration Menu* screen, select *Work with Administration Tasks* on the Main System Menu (Advanced).

DC@P8118 Administration Menu

Enter number of function or place cursor on same line and press Enter.

- 1. Initialize Partition
- 2. Review system settings
- 3. Remove LANSA User
- 4. Authorize Visual LANSA Developer

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Also see

6.1 Partition Initialization

6.2 Review System Settings

6.3 Remove LANSA User

6.4 Authorize Visual LANSA Developer

From the Administration menu, you can:

- Initialize a partition as described in 6.1 Partition Initialization
- Review and change System Settings are described in 6.2 Review System Settings
- Remove a user from this LANSA system as described in 6.3 Remove LANSA User.
- Authorize a Visual LANSA Developer to access the system as described in 6.4 Authorize Visual LANSA Developer.

6.1 Partition Initialization

To reach LANSA's Partition initialization facility, select *Initialise Partition* from the *Administration Menu*.

The Partition Initialization facility enables you to import various Save files as listed on the screen. The options are supplied with the LANSA install or upgrade software or are stored in the LANSA libraries as Save files.

The list of options that you can import is displayed according to the partition's definitions.

Note: If LANSA for the Web is installed and the Web option (LWEBSF) is selected, the partition will be Web enabled after the import is completed. *Web enable* is displayed only if LANSA for the Web is installed for this LANSA system.

Input Options

Job Description and Library

Refer to Job Description and Library

Job Queue and Library

Refer to Job Queue and Library

Out Queue and Library

Refer to Output Queue and Library

Optical Device Name

The optical device name that will be used by this import.

Current list of options

Select from the available options that are listed on the screen.

Remember that during an Install/Upgrade, existing Save files are replaced with new versions.

6.2 Review System Settings

Review of System Settings screen is available from the *Administration* menu which is opened from the Main System Menu (Advanced).

This facility enables you to review and change settings which apply to your LANSA system. These settings are held on the system data areas, DC@A01, DC@A07 and DC@OSVEROP. The option to change the settings will only be available if you have authority to change all these data areas.

The initial screen lists categories of settings. Select the category you wish to review.

DC@P8118 Review System Settings

Select category of settings to review.

Export and Import Task tracking Field and File defaults Compile and Edit options Execution and security Display and print controls Non SSA/CUA partition defaults General information

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Whenever changes are made to data areas DC@A01, DC@A07 or DC@OSVEROP settings, please ensure that these "post-change" procedures are completed:

- 1. Make the same change to all LANSA systems installed on **all** IBM i machines in your organization so that all settings are identical.
- 2. When Visual LANSA users are involved, use the LANSA REQUEST(PCMAINT) command to (re)export the changed system and/or partition definitions into a shared folder. Then use the "Import" facility from

the "Tools" action bar item to import the changed system definition into each and every PC system.

The maintenance of identical system and partition configuration options across all LANSA systems on all IBM i and PCs is vital to the smooth running of a LANSA system.

6.2.1 Export and Import settings

To display the *Work with Export and Import Settings* screen, select *Administration* from the Main System Menu (Advanced), choose *Review system settings* from the *Administration Menu* and then select *Export and Import* from the *Review of System Settings* menu.

DC@P811901 Work with Export and Import Settings				
Compile and export IBM i target release: *CURRENT Export LANSA object authority				
Include VL components in IBM i export/import : Y Include VL components between IBM i and VL : Y Include Web details in export : Y Include Web details in import : Y Include XML details in export : Y Include XML details in import : Y				
Import user exit program User defined Export/Import message logging program: and library :				
Reference field propagation in import : Y				
Export \$\$DTALIB\$\$ substitution default value .: *PARTDTALIB and Prompt on Import default .: Y				
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change				

#1 Screen question is reverse of setting. A Y answer will not set the DC@OSVEROP option. N answer will set the option.

The settings on this screen are retrieved from the data areas. Change authority to these values will be available if you have IBM i change authority to the data

areas and LANSA authority to the export and import menu options.

Setting	DC@A01	DC@A07	DC@OSVEROP
Target release		631-638 Also in compile settings category	
Authority	395-396		
VL components in IBM i			*NOLADCOMP #1
VL components IBM i to Visual LANSA			*NOL4WCOMP #1
Web export			*NOWEBEXP # 1
Web import			*NOWEBIMP #1
XML export			*NOXMLEXP #1
XML import			*NOXMLIMP #1
Import user exit		601-610	
User defined message logging library and program		649-668	
Reference field propagation			*IMPREFFLDNOPROP #1
<pre>\$\$DTALIB\$\$ substitution default</pre>		729-748	
Prompt on Import default for \$\$DTALIB\$\$ substitution		749	

Compile and export IBM i Target Release

Positions 631-638 in data area DC@A07.

Specifies the TGTRLS value that IBM i compiles and saves in export are to use. The TGTRLS parameter is appended to LANSA compile and export requests. Valid values include:

*CURRENT

*PRV

VxRyMz where VxRyMz is a valid IBM i version.

*N which means that the default for the command is used. New LANSA systems are shipped with this value.

If left blank, the target release used is *CURRENT, not the default for the command.

Export LANSA object authority

Positions 395-396 in data area DC@A01.

Specifies the level of LANSA authority required to the object for it to be exported.

UD = Use authority

MD = Modify authority

DD = Delete authority

Include VL components in IBM i export/import

Setting *NOLADCOMP in data area DC@OSVEROP.

This controls how Visual LANSA component-related information is exported and imported between LANSA for the IBM i systems. A N (no) answer will set *NOLADCOMP and disable the export/import of VL components.

Warning: Use of this flag should be considered carefully to avoid loss of component details.

Include VL components between IBM i and VL

Setting *NOL4WCOMP in data area DC@OSVEROP.

A N (no) answer disables Visual LANSA component-related information from being transferred between a LANSA for i system and Visual LANSA systems. The flag affects the export, check out and check in functions. **Warning:** Use of this flag should be considered carefully to avoid loss of component details.

Include Web details in export

Setting *NOWEBEXP in data area DC@OSVEROP.

A N (no) answer will set *NOWEBEXP and disable the export of all Web details. This includes Web components as well as web details associated with fields, functions and system variables.

Include Web details in import

Setting *NOWEBIMP in data area DC@OSVEROP.

A N (no) answer will set *NOWEBIMP and disable the import of all Web details. This includes Web components as well as web details associated with fields, functions and system variables.

Include XML details in export

Setting *NOXMLEXP in data area DC@OSVEROP.

A N (no) answer will set *NOXMLEXP and disable the export of all XML details. This includes XML components as well as web details associated with fields, functions and system variables.

Include XL details in import

Setting *NOXMLIMP in data area DC@OSVEROP.

A N (no) answer will set *NOXMLIMP and disable the import of all XML details. This includes XML components as well as web details associated with fields, functions and system variables.

Import user exit program

Positions 601-610 in data area DC@A07.

Enter an Import User Exit program name if required. During a standard import these bytes are tested. If non-blank, they are assumed to identify a user exit program that should be invoked each time that an IBM i level object is created or changed during the import. Such a program must receive 3 x Char(10) parameters identifying: (1) the object name, (2) the library, and, (3) the object type (in IBM i "*" style object type notation). This facility is provided only for IBM i.

User defined Export/Import message logging program and library

Positions 649-668 in data area DC@A07.

If this value is non-blank, the specified program in the library specified will be

called during export and import when a message is available. The library may be *LIBL. Parameters passed to the user exit program are:

DC@IDS A(1024)	Output only
Message type A(1)	Input/output
Message ID A(7)	Output only
Message text A(132)	Output only

For details, refer to User Exit - Export/Import Message logger.

Reference field propagation in import

Setting *IMPREFFLDNOPROP in DC@OSVEROP.

A N (no) answer will set *IMPREFFLDPROP. If present when an Import executes, it indicates that Reference Field characteristics are not propagated to fields that reference those Reference Fields.

Note that if a Reference Field is changed subsequently, the changes are propagated to the fields that reference it, as usual.

The use of this flag is not recommended as fields may become out of synch with their nominated reference field. Therefore it should be removed from DC@OSVEROP (by setting it to Y) and set to N in Visual LANSA.

Export \$\$DTALIB\$\$ substitution default value and Prompt on Import default

Position 729-749 in Data Area DC@A07

These values are used as the default when an AS/400 export list is created. The substitution value is used to provide a default value for the \$\$DTALIB\$\$ variable. Typically, it would be set to *PARTDTALIB which will resolve to the partition's data library.

The *Prompt on Import* is used as the default value for the *Prompt user when Importing* flag on the \$\$DTALIB\$\$ substitution when an export list is created.

6.2.2 Task Tracking Settings

To display the *Work with Task Tracking Settings* screen, select *Administration* from the Main System Menu (Advanced), choose *Review system settings* from the *Administration Menu* and then select *Task tracking* from the *Review of System Settings* menu.

DC@P8117 Work with Task Settings				
System wide settings				
Product level task tracking				
prefix $\ldots : *T$ check in unlocking $\ldots : Y$				
Developer task tracking				
prefix : *U check in unlocking : Y				
Minimum task tracking				
prefix: *N check in unlocking: Y				
Retain task history on REORG				
Lock functions to same task as process N				
Partition XXX settings				
Task tracking active				
User require task to work				
Prompt/Confirm task id				
Allow user to change task				
Disable special task security N				
Task tracking for imports				
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change				

The settings on the *Work with Task Settings* screen are retrieved from the data areas and the partition details. Authority to change these values will be available if you have IBM i change authority to the data areas, LANSA authority to the partition maintenance and task tracking menus and LANSA authority to change the current partition values.

Setting	DC@A01	DC@A07	DC@OSVEROP	
Prefix and Unlocking		640 - 648		
History		639		
Lock functions			*TTG6FUNCLOCKING	
Task tracking flags retrieved for the current partition.				

Product level task tracking prefix and check in unlocking

Positions 644-645 and 648 in data area DC@A07.

These are optional settings. The prefix is a two character value and is associated with an *Unlock Object after Check In* flag. When this flag is set to "Y", an object is unlocked from the task both on the IBM i and on the workstation when it is checked in. At the same time, it is changed to read-only on the workstation. To change an object again on the workstation, you must check it out for update.

This may be used if you wish to use Task identifiers assigned to products. It is suitable for small teams which have little cross-over between products. It is recommended product-oriented task identifiers use "*T" as the first two characters. If you follow this recommendation, you could, for example, create task identifiers *T00000A and *T00000B for a Product A and Product B. Users need to be authorized to product-oriented tasks as is usual.

Developer level task tracking prefix and check in unlocking

Positions 642-643 and 647 in data area DC@A07.

These are optional settings. The prefix is a two character value and is associated with an *Unlock Object after Check In* flag. When this flag is set to "Y", an object is unlocked from the task both on the IBM i and on the workstation when it is checked in. At the same time, it is changed to read-only on the workstation. To change an object again on the workstation, you must check it out for update.

This option may be used if you wish to have Task identifiers assigned to individual developers. It is suitable for small teams where there is little cross-over between developers. It is recommended developer-oriented task identifiers use "*U" as the first two characters. It is strongly suggested that an appropriate naming convention be used to identify tasks with users. If you use the recommended prefix, you could, for example, create task identifiers such as

*U000BOB or *U00MARY. Only one user can be authorized to a developeroriented task.

Minimum level task tracking prefix and check in unlocking

Positions 640-641 and 646 in data area DC@A07.

These are optional settings. The prefix is a two character value and is associated with an *Unlock Object after Check In* flag. When this flag is set to "Y", an object is unlocked from the task both on the IBM i and on the workstation when it is checked in. At the same time, it is changed to read-only on the workstation. To change an object again on the workstation, you must check it out for update.

This level of task tracking is suitable for small teams with trusted developers. It uses one task for the entire system. Only system internal task tracking activities are performed. There is no locking out of objects and you cannot export by task or inquire on task history. The recommended prefix for the single permanent task is "*N". A suggested name for this task is "*NONE". All users can use this task and they do not need to be authorized to it.

Retain task history on REORG

Position 639 of data area DC@A07.

"Y" indicates that task history is to be retained. "N" indicates that the next *Request Submit LANSA Re-org* job will have *Purge Task History* starting from date pre-filled with today's date.

Lock functions to same task as process

Setting *TTG6FUNCLOCKING in data area DC@OSVEROP.

If this value is "Y", then all functions are required to be locked with the same Task ID as the parent process. This flag only applies when the *Allow user to change tasks* partition configuration flag is set to "N". If an inconsistency between these settings exists, a warning message will result. You may Enter to accept the setting despite the warnings or cancel to abandon the changes.

Partition settings

Refer to Configure Task Tracking in Partition Definitions - Create, Change or Delete.

6.2.3 Field and File Defaults

To display the *Work with Field and File Defaults* screen, select *Administration* from the Main System Menu (Advanced), choose *Review system settings* from the *Administration Menu* and then select *Field and File defaults* from the *Review of System Settings* menu.

DC@P8118 Work with Field and File Defaults Field defaults
Input attributes
- alpha fields : FE
- IIIIIIello IIello FE RD
- alpha fields
- numeric fields :
File defaults
File SIZE parameter default: 10000
2000
3
File LVLCHK parameter default : *YES *YES, *NO
File commitment control default : N Y,N
Always build using SQL : Y Y,N
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

The settings on the *Work with Field and File Defaults* screen are retrieved from these data areas:

Setting	DC@A01	DC@A07	DC@OSVEROP
Field attributes	158 - 317		
File Size	112 - 141		

LVLCHK parameter	142 – 145	
Commitment control	391	
Always build using SQL		*SQL_BUILD

Field default attributes

Positions 158 - 317 in data area DC@A01.

These specify the default attributes for fields. The values specified must be valid field attributes. Refer to information on fields for a detailed list of valid attributes.

File default SIZE parameter

Positions 112 - 141 in data area DC@A01.

This default SIZE parameter is used when initially creating LANSA files. The initial number of records, the increment number of records and the maximum increments may be specified. The parameter must be a valid IBM i size parameter.

File default LVLCHK parameter

Positions 142 - 145 in data area DC@A01.

This default level check, LVLCHK, parameter is used when initially creating LANSA files.

File default commitment control

Position 391 in data area DC@A01.

This default commitment control value is used when initially creating LANSA files. It specifies whether or not the file is to be placed under commitment control. Refer to information on files for a detailed description of the commitment control parameter.

Always build using SQL

Set this option to Y to indicate that physical files and logical files on IBM i are to be built using SQL as much as possible when in an RDMLX partition. The physical file will be built as an SQL table. This keyword will have no effect on how logical files are built, and logical files sharing the access path of a corresponding SQL index no longer occurs.

Note that there are some valid CRTPF /CHGPF parameters that will not be able

to be applied to physical files created as tables using SQL, and similarly some valid CRTLF / CHGLF parameters that will not be able to be applied to logical files created as indexes. One such parameter for both is MAXMBRS with a value other than 1 or *SAME.

RDML files built when the *SQL_BUILD option is in effect can be imported to RDML partitions provided they do not have DB triggers set on, however if they are rebuilt in the RDML partition they lose the effects of the *SQL_BUILD option.
6.2.4 Compile and Edit Settings

To display the *Work with Compile and Edit Settings* screen, select *Administration* from the Main System Menu (Advanced), choose *Review system settings* from the *Administration Menu* and then select *Compile and Edit options* from the *Review of System Settings* menu.

DC@P8118 Work with Compile and Edit Settings
Process and function compile defaults
Produce source listing : N Y,N
Optimize RPG code : N Y,N
Ignore Decimal data errors in RPG : N Y,N
Enable debug in RPG : N Y,N
Compile process : N Y,N
Web Validate numerics
Always build for DBCS (IBM i RDML objects): N Y,N
Process, function and file compiles
Compile and export IBM i target release : *CURRENT
Compile using RPG/IV code Y Y,N
ILE bind RPG/IV modules into program: Y Y,N
OTHER file I/O modules
Use *DATETIME conversion option : Y Y,N
Use *VARCHAR conversion option: Y Y,N
More
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

I/O module compiles	
I/O modules to have use adopted authority : Y Y,N	
Support Client, SuperServer or Server	
Support high speed record blocking in PF key order . : Y Y,N	
Support high speed record blocking in RRN order: Y Y,N	
Support LANSA/Server ODBC interface	
Allow extended files to be added to HST	
Use long (256 character) user stamping (USRC, USRU, USRX) : N	Y,N
Suppress FATAI Crude Flement Complexity Rating in function · N	T7NT
Suppress PATAL Grude Liement Complexity Raung in function, it	Y,IN
Suppress PATAL Grude Liement Complexity Raung in function . It	Y,IN
Editor options	Y,IN
Editor options Default RDML editorL E,U,L	Y,IN
Editor options Default RDML editor:L E,U,L Enable alternate editor user exitsN Y,N	Y,IN
Editor options Default RDML editor:L E,U,L Enable alternate editor user exitsN Y,N	Y,IN
Editor options Default RDML editor:L E,U,L Enable alternate editor user exits:N Y,N	Y,IN
Editor options Default RDML editor:L E,U,L Enable alternate editor user exits:N Y,N	Y,IN

The Work with Compile and Edit Settings are shown on two screens. The page up and down keys may be used to move from one screen to the other. Changes are not committed until you press Enter on second screen.

Setting	DC@A01	DC@A07	DC@OSVEROP
Source listing	146		
Optimize code	147		
Decimal error	148		
Debug	400		
Compile process	461		

Web numerics			*WEBNUMVAL
Always build for DBCS			*DBCS_BUILD
Release level		631-638	
RPGIV			*RPGIV
ILE			*ILE
*DATETIME			*OTHER_DATETIME
*VARCHAR			*OTHER_VARCHAR
Adopt			*IOMNOADOPT
authority			Note – screen question is reverse of flag. So answer of Y does not set *IOMNOADPOPT. Answer of N sets *IOMNOADOPT.
Support client, etc			*IOMXSERVER
Blocking in PF key order			*IOMBLOCKBYKEY
Blocking in RRN order			*IOMBLOCKBYRRN
Support ODBC			*ODBC
Allow HST			*HSTABEXTEND
Use long user stamping fields			*LONG_USER_AUDIT
Suppress Crude warning			*CRUDEWARNONLY
Default	393		

editor			
Editor user exit	401		

Produce source listing default

Position 146 in data area DC@A01.

Specifies the default value for producing a source listing from a compile. This option may be overridden for individual compiles.

Optimize RPG code

Position 147 in data area DC@A01.

Specifies the default value for compiling the RPG program that results from the function compile. This option may be overridden for individual compiles.

Ignore decimal data errors in RPG

Position 148 in data area DC@A01.

Specifies the default value for how decimal data errors should be dealt with in the compiled RPG program that results from the function compile. This option may be overridden for individual compiles.

Enable debug in RPG

Position 400 in data area DC@A01.

Specifies the default value for the function compile. It specifies whether the function should be able to be used in DEBUG mode. Refer to Compiling a Process for detailed information. This option may be overridden for individual compiles.

Compile process

Position 461 in data area DC@A01.

Specifies the default value for the 'Compile the process as well as the functions' option on the function compile. It specifies whether the process should be turned into compiled form, as well as the function. Refer to Compiling a Process for detailed information. This option may be overridden for individual compiles.

Web validate numerics

Equivalent setting is *WEBNUMVAL in data area DC@OSVEROP.

Specifies the default value for the *Validate numerics* option on the function compile in web enabled processes. It specifies if input numeric fields should be validated via JavaScript according to the allowable number of digits before and after the decimal point. This option may be overridden for individual compiles.

Always build for DBCS

Equivalent setting is *DBCS_BUILD in data area DC@OSVEROP

Specifies to always build IBM i RDML objects (DDS files and programs) for DBCS, irrespective of the current build language. This value does not impact the build of RDMLX objects on IBM i or the build of Windows objects. When *DBCS_BUILD is not specified, RDML IBM i objects will be built according to the current build language type.

When *DBCS_BUILD is specified, all RDML IBM i objects will be built as though the current build language is a DBCS language. For example, all DBCS keyboard shift attributes of J, E or O for fields will be generated into the DDS.

Default is N.

Target Release

Positions 631-638 in data area DC@A07.

Specifies the TGTRLS value that IBM i compiles and saves in export are to use. The TGTRLS parameter is appended to LANSA compile and export requests.

Valid values include:

*CURRENT

*PRV

VxRyMz where VxRyMz is a valid IBM i version.

*N which means that the default for the command is used. New LANSA systems are shipped with this value.

If left blank, the target release used is *CURRENT, not the default for the command.

Compile using RPG/IV code

Setting *RPGIV in data area DC@OSVEROP.

Specifies that each program is to be compiled using RPG/IV code, then bound as a single module ILE type program.

Before you attempt to use any of the RPGIV and ILE related switches it is strongly recommended that you first read ILE Implementation.

ILE bind RPG/IV modules into program

Setting *ILE in data area DC@OSVEROP.

Activates the second level of ILE implementation. The compile will statically bind any GUI and multilingual program into the function program and use supplied service programs to dynamically call (CALLB) LANSA internal programs. *** Must be used in conjunction with *RPGIV ***

Before you attempt to use any of the RPGIV and ILE related switches it is strongly recommended that you first read ILE Implementation.

Use *DATETIME conversion option in OTHER file I/O modules

Setting *OTHER_DATETIME in data area DC@OSVEROP.

Indicates that the conversion option *DATETIME is to be used when OTHER file I/O modules are compiled. This allows date (L), time (T) and timestamp (Z) fields to be accessible in LANSA.

Use *VARCHAR conversion option in OTHER file I/O modules

Setting *OTHER_VARCHAR in data area DC@OSVEROP.

Indicates that the conversion option *VARCHAR is to be used when OTHER file I/O modules are compiled. This allows variable length (VARLEN or varchar) fields to be accessible in LANSA in an RDML partition.

Note that this setting does not allow variable length character fields to be used as keys within LANSA. If the physical file or any logical views made known to LANSA have a varchar field as a key, the I/O module will fail to compile.

I/O modules to have use adopted authority

Setting *IOMNOADOPT in data area DC@OSVEROP. An answer of N will set the *IOMNOADOPT in DC@OSVEROP.

When *IOMNOADOPT is set, I/O modules created by LANSA will have USEADPAUT(*NO) i.e. do not use program adopted authority for I/O modules.

Support Client, SuperServer or Server in I/O modules

Setting *IOMXSERVER in data area DC@OSVEROP.

Set this option to "Y" to indicate that I/O modules should be compiled to allow support of:

• LANSA Client applications.

and/or

• LANSA Open applications using blocked I/O methods or the "receive immediate" option

and/or

• Visual LANSA SuperServer applications.

It is recommended that you set this option to "Y".

Setting *IOMXSERVER, *IOMBLOCKBYKEY, *IOMBLOCKBYRRN and *ODBC to Y increases the number of files declared in an I/O module, the amount of static (literal initialized) storage used by an I/O module and the number of subroutines in an I/O module.

Support high speed record blocking in PF order

Setting *IOMBLOCKBYKEY in data area DC@OSVEROP.

Set this option to "Y" to indicate that I/O modules should be compiled to support high speed record blocking in physical file key order. You must set this option when using:

- LANSA Client
- the LANSA Open *BLOCKBYRRNnnnn selection option, or the "receive immediate" option.

The "Support Client, SuperServer or Server" option (*IOMXSERVER) must also be set to "Y" if you set this option.

It is recommended that you set this option to "Y".

Support high speed record blocking in RRN order

Setting *IOMBLOCKBYRRN in data area DC@OSVEROP.

Set this option to "Y" to indicate that I/O modules should be compiled to support high speed record blocking in relative record number order. You must set this option when using:

- LANSA Client
- the LANSA Open *BLOCKBYRRNnnnn selection option or the "receive immediate" option.

The "Support Client, SuperServer or Server" option (*IOMXSERVER) must also be set to "Y" if you set this option.

It is recommended that you set this option to "Y".

Support LANSA Open ODBC interface

Setting *ODBC in data area DC@OSVEROP.

Set this option to "Y" to indicate that I/O modules should be compiled to allow support of the LANSA Open ODBC Interface. The "Support Client, SuperServer or Server" option (*IOMXSER/ER) must also be set to "V" if you

SuperServer or Server" option (*IOMXSERVER) must also be set to "Y" if you set this option.

Allow extended files to be added to HST

Setting *HSTABEXTEND in data area DC@OSVEROP.

Setting this option to "Y" allows database files with record lengths up to 1988 bytes to be added to a user index for high speed lookup.

Warning: Review to Database File Attributes before using this option.

Warning: It is strongly recommended that if this option is changed, then all files tagged as high speed tables, all read-only functions that use these files and all other I/O modules and Dboptimized functions that use high speed tables for lookup validation rules be recompiled AFTER deleting the current user index. The user index is DC@TBLIDX if changing setting to "Y" (adding *HSTABEXTEND) or DC@TBLIDY if changing setting to "N" (removing *HSTABEXTEND).

Use long (256 character) user stamping (USRC, USRU, USRX)

Setting *LONG_AUDIT_USER in data area DC@OSVEROP.

Setting this option to "Y" allows for user stamping fields to be supplied with a user name of up to 256 characters. Refer to detailed information in Output Attributes before using this feature.

Suppress FATAL Crude Element Complexity Rating in function

Setting *CRUDEWARNONLY in data area DC@OSVEROP.

Setting this option to "Y" indicates that a function that would cause the "Crude Element Complexity Rating" to return a FATAL will cause a WARNING only.

Warning: It is not recommended to use this setting as the function may subsequently fail to compile.

Default RDML editor

Position 393 in data area DC@A01.

Possible values:

"L" - edit by standard LANSA editor

"E"- edit by IBM's SEU or EDTSRC editors.

"U"- edit by alternate user exit editor.

Enable alternate editor user exits

Position 401 in data area DC@A01.

Setting this option to "Y" will enable alternate editor user exits when editing help text and/or RDML commands. Refer to detailed information on User Exit Programs before using this feature.

6.2.5 Execution and Security Settings

To display the *Work with Execution and Security Settings* screen, select *Administration* from the Main System Menu (Advanced), choose *Review system settings* from the *Administration Menu* and then select *Execution and security* from the *Review of System Settings* menu.

DC@P8118 Work with Execution and Security Settings
Century date determination Comparison year 50 Century for less or = to comparison year : 19 Century for greater than comparison year : 20
Reclaim resources on exit from LANSA: Y Replace library list on exit from LANSA: Y
Exchange all fields between functions on prompt . : Y Allow LANSA exchange for RPG, etc: Y
Enable cursor location sensitivity
Concertine incongeotine Change

Import user exit program:
in library :
Use Function level security Y
External security matching for LANSA files : N
Disable end user file level security: N
Relax restrictions on trigger functionality : N
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

The Work with *Execution and Security Settings* are shown on two screens. The page up and down keys may be used to move from one screen to the other. Changes are not committed until your press Enter on the second screen.

Setting	DC@A01	DC@A07	DC@OSVEROP
Date determination	488-493		
Reclaim resources	480		
Replace library list	481		

Exchange fields	499		
RPG exchange	487		
Cursor location sensitivity	436		
Attention key enabled	462		
Interpretative process message	495		
Field help from dictionary	476		
Use Panel Group for help			*UIMHELP
Permanent file overrides			*PERMFILOVR
Function Routing *LIBL			*FUNRTRLIBL
Import user exit		601-610	
Set environment user exit		611-620	
Submit user exit		621-630	
Prompt user exit enabled	498		
User Prompt handler		11-20	
User Error handler		1-10	
User defined access program and library		669-688	
User defined IJDT program and		689-708	

library			
User defined CIDT program and library		709-728	
Execution locking on process	422		
Function level security	475		
External security on files	486		
Disable end user security on process/function	496		
Disable end user security on files	497		
Relax restrictions on triggers			*EXTENDED_TRIGGERS

Date determination

Positions 488-493 in data area DC@A01.

These three values are used to determine the century for six and four character dates. Such dates have their year compared to the *Comparison year* value.

The first century., the *Century for less or = to comparison year*, is used if the year is less than or equal to the comparison year value. The second century, the *Century for greater than comparison year* is used if the year is greater than the comparison year value.

You must set these values.

Reclaim resources on exit from LANSA

Position 480 in data area DC@A01.

When this value is set to "Y" a RCLRSC (reclaim resources) command will be executed on exit from LANSA.

Replace library list on exit from LANSA

Position 481 in data area DC@A01.

When this value is set to "Y", a RPLLIBL (replace library list) command will be executed on exit from LANSA. This command resets the job's library list to what it was on entry to LANSA.

Exchange all fields between functions on prompt

Position 499 in data area DC@A01.

When this value is set to "Y", all other fields used by a function that will fit into the space left in the exchange list will be exchanged on a prompt request.

Allow LANSA exchange for RPG, etc

Position 487 in data area DC@A01.

A value of "Y" for this option allows the EXCHANGE list capability in RPG / CL / COBOL etc programs so that values can be returned from LANSA to these programs (using EXCHANGE). Also values can be put on the LANSA exchange list from RPG / CL / COBOL etc programs. Refer to the EXCHANGE command for more details. Recompilation is necessary for changes to this setting to take effect.

N is the default.

Enable cursor location sensitivity (prompt, help)

Position 436 in data area DC@A01.

This value must be set to "Y" to enable field prompting and field level help. Recompilation is necessary for changes to this value to take effect.

Enable LANSA attention key

Position 462 in data area DC@A01.

When this option is enabled, the use of the attention key will result in the main system menu being re-displayed for you to select what you want to do. At any time the task/activity that you were performing before using the attention key can be resumed by using function key 3. Note also that some entries on the field, file, process or housekeeping menus may be displayed with a message indicating they are "Suspended or N/A". This indicates that they cannot be effectively used in conjunction with the activity that is currently suspended (usually because program recursion would occur). If you select one of these "Suspended or N/A" entries your suspended activity will be immediately resumed just as if you had used function key 3.

Suppress "Process in interpretative mode" message

Position 495 in data area DC@A01.

Controls the display of the "Process will run in interpretive mode, compiled version not usable" message. This message is displayed as a warning when a process has been selected for use, informing the user that a "new" compiled version of a Process should be created.

Y - Do not display warning message.

N - Display warning message. This is the default.

Generate field level help from dictionary

Position 476 in data area DC@A01.

Set this value to "Y" to enable the automatic generation of field level help text from dictionary definitions and validation rules. This automatically generated help text is appended to any applicable user defined help text.

Use Panel Groups for user defined help text

Setting *UIMHELP in data area DC@OSVEROP.

Set this value to "Y" to set the *UIMHELP option. This indicates that the IBM's Panel Groups are to be used for the presentation of user defined help text, rather than the LANSA help text display facility. The setting *V2WINDOWS must also be set. The *V2WINDOW option may be set in the 6.2.6 Display and Print Controls setting *Popup window to use OS400 window facility*.

Allow permanent file overrides

Setting *PERMFILOVR in data area DC@OSVEROP.

Set this value to "Y" to set the *PERMFILOVR option. Setting this option allows permanent file overrides to be used. When you specify permanent file overrides you are telling LANSA that "every time I use this file, I really want to use this other file". This is useful when you want to use files with names that are not acceptable to LANSA, such as files with a 10 character file name or with a full stop (.) in their name. Refer to The Permanent File Overrides Facility for details and examples.

Use function routing table from *LIBL

Setting *FUNRTRLIBL in data area DC@OSVEROP.

Set this value to "Y" to set the *FUNRTRLIBL option. When this option is set the Function Routing table X_FUNRTR will be accessed from *LIBL. If this is not set the Function Routing table in the partition module library will be used. Refer to Function Routing for further details.

Import user exit program

Positions 601 - 610 in data area DC@A07.

During a standard import these bytes are tested. If non-blank they are assumed to identify a user exit program that should be invoked each time that an IBM i level object is created or changed during the import. Such a program must receive 3 x Char(10) parameters identifying: (1) the object name, (2) the library, and, (3) the object type (in IBM i "*" style object type notation). This facility is provided only for IBM i.

Set environment user exit program

Positions 611 - 620 in data area DC@A07.

After LANSA has set up its environment for the partition these bytes are tested. If non-blank they are assumed to identify a user exit program that should be invoked each time that a LANSA partition is entered. Such a program must receive 3 parameters identifying: (1) the LANSA program library Char(10), (2) the partition ID Char(3), and (3) the language code Char(4).

SUBMIT command user exit program

Positions 621 - 630 in data area DC@A07.

The first time a LANSA function is submitted these bytes are tested. If nonblank they are assumed to identify a user exit program that should be invoked each time a function is submitted. Such a program must receive one parameter Char(2000) containing the SBMJOB command about to be used. The user exit program can modify the SBMJOB command before exiting and passing back the parameter.(e.g. Add MSGQ() parameter to SBMJOB).

Call user defined prompt key handler

Position 498 in data area DC@A01.

Set this value to "Y" to indicates that a user defined prompt key message handler program should be called in preference to the LANSA prompt key message handler, when the prompt key function key has been selected. If this value is set to "Y", a user defined prompt key message handler program name must be specified.

Name of user defined prompt key handler

Positions 11 - 20 in data area DC@A07.

If the "Call user defined prompt key handler" is set to "Y", then an attempt will be made to call the user defined prompt key message handler program specified in this option when the prompt key function key is used in your LANSA application.

Name of user defined program/file error handler

Positions 1 - 10 in data area DC@A07.

If non-blank this value should identify a user defined error handler. During LANSA execution, an attempt will be made to call this user defined program/file error handler program by the LANSA program/file error routine. The user defined program/file error handler can be used to update audit/error logs external to the LANSA system.

User defined LANSA access check program and library

Positions 669 - 688 in data area DC@A07.

If this value is non-blank, the specified program in the specified library will be called during LANSA access checking. The user-defined program can manipulate the processing of the LANSA access check for an object and it will return two parameters that will influence the processing The library may be *LIBL.

For details, refer to User Exit LANSA Security Check.

User defined LANSA IJDT access check program and library

Positions 689 - 708 in data area DC@A07.

If this value is non-blank, the specified program in the specified library will be called before the LANSA IJDT is executed. The user-defined program can manipulate the processing of the LANSA IJDT (I Just Did This) and will return two parameters that will influence the processing. The library may be *LIBL.

For details, refer to User Exit IJDT (I Just Did This).

User defined CIDT program and library

Positions 709 - 728 in data area DC@A07.

If this value is non-blank, the specified program in the specified library will be called during LANSA access checking. The user-defined program can manipulate the processing of the LANSA access check for an object and it will return two parameters that will influence the processing The library may be *LIBL.

For details, refer to User Exit CIDT (Can I Do This).

Execution locking on process

Position 422 in data area DC@A01.

If this value is set to "Y", a shared lock will be set on the process during

execution. If a shared lock on the process can not be established then execution of the process will fail.

Use function level security

Position 475 in data area DC@A01.

If this value is set to "Y", LANSA function level security is enabled. Otherwise, process level security is used. Refer to LANSA Objects and Security for full details on LANSA security.

Note that using function level security increases overall system resource use. Processes (NOT functions) compiled prior to a change to this value will ignore the change and should be recompiled.

External security matching for LANSA files

Position 486 in data area DC@A01.

This is set to a "Y" to indicate that LANSA database security should be reflected into IBM i object level security. This option only applies to files set up and maintained by LANSA. It does not apply to "OTHER" files.

Refer to External Security Matching and Reviewing and Changing Access to LANSA Objects before changing security settings.

Disable end user process and function security

Position 496 in data area DC@A01.

A value of "Y" indicates that end user process and function level security is disabled in this system. This setting will improve system performance, because no security checking is performed before accessing end user applications. This flag has no effect on access to process or function definition details in a development environment. This setting is appropriate for installations that use an external menu system to control and secure access to LANSA applications.

The setting of this flag is interpreted dynamically by applications. That is, it does not remain as set when the code is compiled, but is evaluated at runtime.

Refer to Reviewing and Changing Access to LANSA Objects before changing security settings.

Disable end user file level security

Position 497 in data area DC@A01.

A value of "Y" indicates that end user file level security is disabled in this system. This setting will improve system performance, because no file level security checking is performed in end user applications. This flag has no effect on access to file definition details in a development environment. This setting is appropriate for installations that use the approach "if the program is accessible from your menu then you can access all the files the program requires".

The setting of this flag is interpreted dynamically by applications. That is, it is not statically set when the code is compiled, but is evaluated at runtime.

Refer to Reviewing and Changing Access to LANSA Objects before changing security settings.

Relax restrictions on trigger functionality

Setting *EXTENDED_TRIGGERS in data area DC@OSVEROP.

A value of "Y" will set the option *EXTENDED_TRIGGERS which indicates that the normal LANSA rules that prevent database event triggers from calling other functions and using "user interface" commands should be relaxed.

This use of this option is not recommended in most circumstances.

6.2.6 Display and Print Controls

To display the *Work with Display and Print Controls* screen, select *Administration* from the Main System Menu (Advanced), choose *Review system settings* from the *Administration Menu* and then select *Display and print controls* from the *Review of System Settings* menu.

DC@P8118 Work with Display and Print Controls
Default for enable prompt key on screen commands default . : Y LANSA decimal point format: Show a leading zero: N Popup window to use OS400 window facility: Y Popup window to trim second function key line . : Y Use default STD_HEAD parameters: Y
Printer form width default : 132 Printer form length default : 066 Printer overflow line default : 060 Vertical bar character : Horizontal bar character : -
Field label fill character
Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

The *Work with Display and Print Controls* settings are retrieved from the data areas.

Setting	DC@A01	DC@A07	DC@OSVEROP
Prompt key	477		
Decimal format	494		
Leading zero			*LEADZERO=1
Popup window			*V2WINDOWS
Popup trim			*WINDOWTRIM
Printer width	149-151		
Printer length	412-414		
Printer overflow	415-417		
Default STD_HEAD	421		
Vertical bar character	484		
Horizontal bar character	485		
Field label fill	463		
Field column underline	464		
Hex for 132 widthworkstation	441-460		
Hex for color workstation	465-474		

Default for enable prompt key on screen commands

Position 477 in data area DC@A01.

If this value is set to "Y", the PROMPT_KEY parameter on DISPLAY, REQUEST and POP_UP RDML commands will be enabled by default. If set to "N" the prompt key is disabled by default. The default may be specifically changed for individual commands in your RDML code.

LANSA decimal point format

Position 494 in data area DC@A01.

This should relate to the IBM i system value QDECFMT. If QDECFMT = " "

(blank) then the LANSA decimal format will = "." (full stop). If QDECFMT = "I" or "J" then the LANSA decimal format will = "," (comma). Values:

"."- LANSA decimal format is a full stop.

","- LANSA decimal format is a comma.

other - LANSA decimal format is defaulted to a full stop.

Warning: If the value of this field does not relate to the value of QDECFMT then unpredictable results could occur with the use of numeric values.

Show a leading zero

Setting *LEADZERO=1 in data area DC@OSVEROP.

Setting this value to "Y" indicates that Visual LANSA code will display numeric values containing decimals with a leading zero. For example 0.12 or 0,12 depending on the LANSA decimal point format. This option would be suitable when QDECFMT="J".

Popup window to use AS400 window facility

Setting *V2WINDOWS in data area DC@OSVEROP.

Setting this value to "Y" indicates that pop-up windows created by LANSA should use the IBM i windowing facilities. The use of the option is recommended on all IBM i systems.

Popup window to trim second function key line

Setting *WINDOWTRIM in data area DC@OSVEROP.

Setting this value to "Y" indicates that an existing LANSA defined pop-up window has the 2nd function key line trimmed from the display.

Only use this option in conjunction with the *V2WINDOWS option to enable existing functions that have been specifically sized onto line 24 of the display device to recompile without change.

Use default STD_HEAD parameters

Position 421 in data area DC@A01.

This value specifies the system default for the STD_HEAD parameter on screen commands.

STD_HEAD parameter on screen commands specifies whether or not the standard LANSA design for the screen heading lines (lines 1 and 2) should be used.

Printer form width default

Positions 149 - 151 in data area DC@A01.

This value specifies the default printer form width. Typically 132.

Printer form length default

Positions 412 - 414 in data area DC@A01.

This value specifies the default printer form length (RPG limit). Typically 66.

Printer overflow line default

Positions 415 - 417 in data area DC@A01.

This value specifies the default printer form overflow line (RPG limit). Typically 60.

Vertical bar character

Position 484 in data area DC@A01.

This value specifies the vertical bar character to be used on reports such as RDML listings. If this value is left as blank (X'40') it defaults to ":" (X'6A'). This option may not be suitable for some national language sets and should be changed.

Horizontal bar character

Position 485 in data area DC@A01.

Horizontal bar character to be used on reports such as RDML listings. If this value is left as blank (X'40') it defaults to "-" (X'60'). This option may not be suitable for some national language sets and should be changed.

Field label fill character

Position 463 in data area DC@A01.

If specified, this value is used as the fill character for field labels which are automatically created. Field labels are automatically created if a field is created or changed and the Label is left blank.

Note: Recompilation is necessary for changes to this setting to take effect.

Field column heading underline character

Position 464 in data area DC@A01.

If specified, this value is used as the underline character for field column headings which are automatically created. Field column headings are automatically created if a field is created or changed and the Column Headings are left blank. Note: Recompilation is necessary for changes to this setting to take effect.

Hex code for 132 width workstations

Positions 441 - 460 in data area DC@A01.

These values are shown and may be entered as the hex digits. Eg a value of 40 is the hex for blank.

This is only used by the report painter to aid the user design report layouts. Refer to the appropriate IBM manual for list of codes.

Hex code for color workstations

Positions 465 - 474 in data area DC@A01.

These values are shown and may be entered as the hex digits. Eg a value of 40 is the hex for blank.

This is a list of display device codes which are color workstations. This list is used by LANSA at execution time to determine if the workstation being used is capable of displaying information in color. Refer to IBM documentation to determine the codes for workstations.

6.2.7 Non SAA/CUA Partition Defaults

To display the *Work with Non SAA/CUA Partition Defaults* screen, select *Administration* from the Main System Menu (Advanced), choose *Review system settings* from the *Administration Menu* and then select *Non SAA/CUA partition defaults* from the *Review of System Settings* menu.

DC@P8118 Work with Non SAA/CUA Partition Defaults	
Exit/System function key: 01Menu/Cancel function key: 02Message function key: 07Add function key: 09Change function key: 10Delete function key: 11Prompt function key: 04	
Display attributes on identification text Include high intensity : N Include reverse image : N Include underline N Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change	

The settings on the *Work with Non SAA/CUA Partition Defaults* screen are retrieved from the data areas.

These settings apply to non-SAA/CUA partition only.

SAA/CUA partition defaults are specified at partition level and can be viewed and maintained in the partition maintenance option.

Setting	DC@A01	DC@A07	DC@OSVEROP
Exit key	99-100		
Menu key	101-102		

Message key	103-104	
Add key	105-106	
Change key	107-108	
Delete key	109-110	
Prompt key	478-479	
INDENT_ATR high	418	
INDENT_ATR reverse	419	
INDENT_ATR underline	420	

Exit/System function key

Positions 99 - 100 in data area DC@A01.

Specifies the EXIT function key to be used by internal LANSA programs in non-SAA/CUA partitions.

Menu/Cancel function key

Positions 101 - 102 in data area DC@A01.

Specifies the MENU function key to be used by internal LANSA programs in non-SAA/CUA partitions.

Message function key

Positions 103 - 104 in data area DC@A01.

Specifies the MESSAGES function key to be used by internal LANSA programs in non-SAA/CUA partitions.

Add function key

Positions 105 - 106 in data area DC@A01.

Specifies the ADD function key to be used by internal LANSA programs in non-SAA/CUA partitions.

Change function key

Positions 107 - 108 in data area DC@A01.

Specifies the CHANGE function key to be used by internal LANSA programs in non-SAA/CUA partitions.

Delete function key

Positions 109 - 110 in data area DC@A01.

Specifies the DELETE function key to be used by internal LANSA programs in non-SAA/CUA partitions.

Prompt function key

Positions 478 - 479 in data area DC@A01.

Specifies the PROMPT function key to be used by internal LANSA programs in non-SAA/CUA partitions.

Display attribute on identification text - include high intensity

Position 418 in data area DC@A01.

Specify "Y" if the default IDENT_ATR parameter is to include high intensity. The INDENT_ATR parameter is associated with the DISPLAY, POP_UP and REQUEST commands in non-SAA/CUA partitions. It specifies display attributes that are to be associated with identification text (labels, descriptions, column headings, etc) that are displayed on the screen.

Display attribute on identification text - include reverse image

Position 419 in data area DC@A01.

Specify "Y" if the default IDENT_ATR parameter is to include reverse image.

The INDENT_ATR parameter is associated with the DISPLAY, POP_UP and REQUEST commands in non-SAA/CUA partitions. It specifies display attributes that are to be associated with identification text (labels, descriptions, column headings, etc) that are displayed on the screen.

Display attribute on identification text - include underline

Position 420 in data area DC@A01.

Specify "Y" if the default IDENT_ATR parameter is to include underline. The INDENT_ATR parameter is associated with the DISPLAY, POP_UP and REQUEST commands in non-SAA/CUA partitions. It specifies display attributes that are to be associated with identification text (labels, descriptions, column headings, etc) that are displayed on the screen.

6.2.8 General System Information

To display the *LANSA Configuration Details* screen, select *Administration* from the Main System Menu (Advanced), choose *Review system settings* from the *Administration Menu* and then select *General information* from the *Review of System Settings* menu.

The *LANSA Configuration Details* screen displays details about this LANSA system. It is frequently called 'LANSA About' as it can be obtained using the ABOUT action on the REQUEST parameter of the LANSA command.

LANSA Configuration Details

Screen 1

LANSA Details Version : 99.9 Spin Number : SPIN999 Last EPC : 999 Minor Version Level : X9 : XXXPGMLIB Database Library Program Library : XXXDTALIB Communication Library : XXXCOMLIB Owner : QOTHPRDOW Machine Details : V9R9M0 IBM i Serial Number : 999999X IBM i Version System Model Number : 999 Processor Feature : 9999 Listener Details : X99LISTJQ Job Description Job Queue : XXXLISTJD Subsystem : X99PGMLIB Web Details Web Server Type : CGI HTTP Server Type : IBM-Apache : X99WRKLIB CGI Library : X99COMLIB Work Library Authentication Library : N/A **HTTP** Instance : X99PGMLIB Integrator Library Name : X99PGMLIB **Open System Details** : X99OSULIB Press Enter for Details Library Name

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages Fnn=Change

Screen 2 contains IFS Details:

- Communication Directory
- Image Directory
- Apache Directory
- Integrator Directory

By option, you may capture this data on a save file to send to LANSA support. The save file is DCXSUPDTA and it will be saved in the LANSA_XXXPGMLIB/SUPPORT directory.

This LANSA IBM information is also available from the LANSA ABOUT command.

6.3 Remove LANSA User

To display the *Remove LANSA user* screen, select *Administration* from the Main System Menu (Advanced), choose *Remove LANSA User* from the *Administration Menu*.

The *Remove LANSA User* facility enables you to remove references to a user from all partitions in this LANSA system. You must enter the specific user to be removed.

References to the user will be removed from

- **security file** This will remove the user's authority to all LANSA objects in this LANSA system and in all partition**s**
- task definitions

Any reference to the user in task definitions will be removed. If the user was the only Authorized User to the task, the user will be replaced by the current partition security officer.

• PC user

Any PC user definition which has the user as the "AS/400 User Id" will be removed.

To use this option, you must be QSECOFR, part of the QSECOFR group, the Partition Security Officer or part of the Partition Security Officer group.

You will be asked to confirm the removal request. The process will be run interactively.

A related facility exists from the Housekeeping menu, the Re-organizing the LANSA Internal Database. An option in this reorgranize process allows you to remove any user on the LANSA security file who no longer exists as an IBM i user profile.

6.4 Authorize Visual LANSA Developer

To display the *Authorize Visual LANSA Developer* screen, select *Administration* from the Main System Menu (Advanced), choose *Authorize Visual LANSA Developer* from the *Administration Menu*.

The *Authorize Visual LANSA Developer* facility enables you to authorize a Visual LANSA developer to create fields, files, processes and so on in a partition.

To use this option, you must be logged on as QSECOFR, or be part of the QSECOFR group, the Partition Security Officer or part of the Partition Security Officer group.

The Visual LANSA developer must exist as an IBM i user profile.

You maintain this user profile using a related facility, the Review user access to LANSA system which is available from the *Housekeeping* menu.

7. Documentor

The Documentor facility in LANSA is provided to enable you to generate technical and user documentation for LANSA based applications.

To document a LANSA application, you first prepare a list of the objects and associated subjects to be documented and then specify what is to happen to the generated documentation.

Every document you create has a name and remains in the LANSA system until you delete it.

Objects (e.g. fields, files, processes) which are referenced in a document will be automatically removed from the document if the object is deleted from your LANSA system.

7.1 Work with Documents

The steps involved in working with documents can be visualized like this:



7.2 Work with Objects to Be Documented

When you select the option Work with System Documentation option from the Main System Menu, the first screen presented looks something like this:

DC@P840101 Work with Documents

Type options and press Enter

2=Review/Change definition 4=Delete 5=Review/Change Selected Objects 7=Produce Document 17=Last Action Details

Opt	Document Name	Description
	DOCPSLSYS	Document describing the Personnel System
	DOC0000001	Technical Document only
	DOC000002	Document all panel layouts for Personnel

F1=Help F3=Exit F6=Create F12=Cancel F14=Msgs

Working from the "Work With Documents" screen you can, by entering one of these options beside a list entry, elect to:

- 2=Review/Change Definition: indicates that the definition of a document is to be reviewed or changed.
 When option 2 for review/change is used, a screen showing the detailed definition of the document will be displayed. This screen is described in more detail later.
- **4=Delete Document:** deletes an existing document definition. A confirmation of delete screen will be displayed. Then the above screen will be re-displayed with the deleted document omitted.

• **5=Review/Change Selected Objects:** indicates that the list of objects to be documented is to be reviewed and possibly altered. When the REVIEW option is used a menu headed "Review/Change Selected Objects" will be displayed.

By working from this menu it is possible to review and alter the objects that are included for documentation. This menu is described in more detail later.

- **7=Produce Document:** indicates that you wish to generate a document for all the objects nominated in the list. The generated document can either be a spooled file or copied to a PC document.
- **17=Last Action Details**: indicates that you wish to review the last action details recorded for the document. Enter this option against any existing document shown on the display. An additional screen will appear showing the last action details. This screen is described in detail previously.
- Create: Use the ADD function key to indicate that a new document is to be defined. Once this has been done a screen will be presented that allows specification of the details of the new document. When all the new document details are valid, the document definition will be created. Next a menu headed "Review/Change Selected Objects" will be displayed. By working from this menu it is possible to review and alter the objects that are included for documentation. This menu is described in more detail later.

7.3 Input Options to Create, Review or Change Document Definition

When an existing document has been selected for display (and possible change) or the ADD function key has been used to indicate that a new document is to be defined, a screen similar to this will be presented:

DC@P840102 Review/Change Document Definition
Type Document details and press Enter. Document Name : Description : Cover Page : <u>N</u> Page Heading
\$\$PAGE
Page Footing
Portrait mode instruction (if applicable)
Landscape mode instruction (if applicable)
Appendices Required;Field details (by field) : \underline{N} Or Field details (summary) : \underline{N} File details : \underline{N} Message details : \underline{N}
F1=Help F3=Exit F12=Cancel F14=Msgs F21=Change

Working from this screen it is possible to:

• Input details of a new document.
- Review details of an existing document.
- Use the change function key to indicate that the document definition is to be changed. If this option is used the screen will be re-presented with some fields input capable. Make the desired changes and press enter to proceed.

The following input options apply when specifying details of a new document definition or changing details of an existing document definition:

Document Name

Specifies the name by which the new document is to be known. Must be unique.

Document Description

Specifies the description of this document. Must not be blank.

Cover Page

Mandatory, but always pre-filled to "N". Specifies if a cover page is to be specified for the document.

Allowable values are:

Y Specify cover page details for the document.

N Do not specify cover page details for the document.

If "Y" is entered, a screen will be displayed to allow entry of the cover page details.

Any text entered into the cover page lines will be centered within the length of the cover page.

Page Heading

Optional. The default for page heading line two is the page number substitution variable (\$\$PAGE). This variable will be replaced by the page number when the document is produced, and can be placed anywhere in the header and footer lines.

Page Footing

Optional. Substitution variable \$\$PAGE can be placed anywhere in the three footer lines and will be replaced by actual page number.

Portrait Instruction

Optional. The portrait instruction can be entered if a printer is being used to print the generated document which will require an instruction to place it into portrait mode.

Landscape Instruction

Optional. The landscape instruction can be entered if a printer is being used to print the generated document which will require an instruction to place it into landscape mode.

Appendices Required

Mandatory, but always pre-filled to "N". Y: Include the requested appendix into the document. N: Do not include the requested appendix into the document.

Note: It is not possible to request that both the field detail and the field summary appendices appear in the same document.

7.4 Work with Selected Objects to Be Documented

When a new document is created or an existing document is reviewed a screen like this will be presented to the user:

DC@P840106 Review/Change Selected Objects

- _ 1. Include fields in document
- 2. Include files in document
- 3. Include processes/functions in document
- 4. Include other documents
- 5. Review/delete objects already included in document
- 6. Review/change Table of Contents

F1=Help F3=Exit F12=Cancel F14=Msgs

By working from this screen the objects that are defined in the new or existing document can be manipulated. In summary, the facilities provided to manipulate the list are:

- Include new LANSA fields in the document.
- Include new LANSA files in the document.
- Include new LANSA process and/or functions in the document.
- Include other documents within this document. (Any document being included must be a final form IBM i document).
- Review all objects currently included in the document and delete those no longer required.
- Review the Table of Contents for the document and omit or reorder the

sequence of subjects within the document.

7.5 Include Fields in the Document

When the option to "Include Fields in Document" is chosen from "Work with Documents", a screen similar to this example is used to include fields in the document.

DC@P840107 Include Fields in Document

Enter full or partial name of the data dictionary field(s) to be worked With or leave blank to select from all fields in dictionary or enter *ALL to select all fields in the data dictionary<u>A</u>_____

Field

- Sel Name Description _ AB\$OPT Action Bar Option
- X ADDRESS1 Street No and Name
- X ADDRESS2 Suburb or Town
- X ADDRESS3 State and Country

F1=Help F3=Exit F12=Cancel F14=Msgs

This facility allows you to build temporary lists of fields from the LANSA data dictionary and display them on the screen. From these displayed lists fields can be chosen for inclusion into your document.

These lists can be built from all fields, only fields that have a certain generic name, or all fields in the dictionary can be chosen automatically for inclusion into your document.

A displayed list may not fit on one screen. This is indicated by a "+" sign in the lower right of the screen. In such cases use the roll up and roll down keys to

scroll backwards and forwards through the displayed list.

Fields shown in the displayed list which are already included in the document are shown with an "X" beside them. The "X" cannot be removed. Thus a field cannot be removed from the document using this facility. Use the option "Review/exclude objects in document" to do this. This option is described in detail in a later section.

When you want to choose a field from the displayed list for inclusion into the document enter a "Y" beside it in the column headed "Sel".

When you have completed using this facility, use the Cancel function key to return to the "Review/Change Selected Objects" Menu.

7.6 Include Files in the Document

When the option to "Include files in Document" is chosen from the "Review/Change Selected Objects" menu a screen similar to this example is used to include files in the document.

DC@P840108 Include Files in Document

Enter full or partial name of the file definition(s) to be worked With or leave blank to select from all file definitions

File		Include all				
Sel	Name	Library	Description	Fie	lds	
Х	DEPTAB	DC@E	DEMOLIB Dej	partment code ta	bl	
_	LCOI01	DC@DE	MOLIB Testi	ng LCOI file	$\underline{\mathbf{N}}$	
_	LCOI02	DC@DE	MOLIB Large	e LCOI test file	$\underline{\mathbf{N}}$	
_	LCOI03	DC@DE	MOLIB LCO	I Department Fil	e <u>N</u>	
_	LCOI04	DC@DE	MOLIB LCO	I Section File	$\underline{\mathbf{N}}$	
Х	PSLMST	DC@D	EMOLIB Per	sonnel master		
Х	PSLSKL	DC@D	EMOLIB Pers	sonnel skills		
Х	SECTAB	DC@D	EMOLIB Sec	tion code table		
Х	SKLTAB	DC@D	EMOLIB Ski	ll code table		

F1=Help F3=Exit F12=Cancel F14=Msgs

This facility allows you to build temporary lists of active file definitions known to LANSA and display them on the screen. From these displayed lists file definitions can be chosen for inclusion into your document.

Note the use of the words file definition. Only the name of the file definition, which is the same as the associated physical file name, is ever included into the

list of objects to be documented. When a file definition is included, all associated details such as logical views, virtual fields, etc, are automatically included for documentation.

These lists can be built from all files or only files that have a certain generic name.

A displayed list may not fit onto one screen. This is indicated by a "+" sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

Files shown in the displayed list which are already included in the document are shown with an "X" beside them. The "X" cannot be removed. Thus a file cannot be excluded from the document by using this facility. Use the option "Review/exclude objects in document" to do this. This option is described in detail in a later section.

When you want to choose a file from the displayed list for inclusion into the document enter a "Y" beside it in the column headed "Sel".

The "Include all Fields" column is used (by entering a "Y" in this column beside a chosen file) to indicate that all fields defined on this file are also to be included for documentation.

When you have completed using this facility, use the Cancel function key to return to the "Review/Change Selected Objects" Menu.

7.7 Include Processes/Functions in Document

When the option to "Include processes/functions in document" is chosen from the "Review/change Selected objects" Menu a screen similar to this example is used to include processes and functions into the document.

DC@P840109 Include Processes/Functions in Document

Enter full or partial name of the process(s) to be worked With or leave blank to select from all processes <u>P</u>

	Process /	Incl	ude all				
Sel	Function	Description	Files 1	Fields			
Х	PSLSYS	Personnel System Main N	A enu				
Х	EMPLIST	Full Employee Listing		Y	Y		
Х	ENROL	Enrol a New Employee		Y	Y		
Х	INQUIRE	Browse/Maintain Emplo	yee and S	Skill F	Files Y	Y	-
Х	PHONE	Employee Business Phon	e Numbe	r List	Y	Y	
Х	SALARY	Produce Salary Commit	ment Rep	orts	Y	Y	
Х	SEARCH	Perform General Employ	yee Searc	h	Y	Y	
Х	SNAME	Telephone Number Searc	ch	Y	Y		
Х	WINDOV	V Dept/Section/Employee	e Window	v Enqu	uiry	Y	Y
_	PSLSYS_A	B Work with Employees					
Х	EMPLOY	Employee	$\underline{\mathbf{N}}$	Y			

F1=Help F3=Exit F12=Cancel F14=Msgs

This facility allows you to build temporary lists of processes (and associated functions) known to LANSA and display them on the screen. From these displayed lists processes and/or functions can be chosen for inclusion into your document.

These lists can either be built from all processes, or only from processes that

have a certain generic name.

When any process is included into a displayed list all its associated functions are also shown for possible selection.

A displayed list may not fit on one screen. This is indicated by a "+" sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed list.

Processes and/or functions shown in the displayed list which are already included in the document are shown with an "X" beside them. The "X" cannot be removed. Thus a process or function cannot be excluded from the document by using this facility. Use the option "Review/exclude objects in document" to do this. This option is described in detail in a later section.

When you want to choose a process or function from the displayed list for inclusion into the document enter an "Y" beside it in the column headed "Sel".

An option is provided to automatically include into the document all fields that are used by a function. To do this enter a "Y" in the column headed "Include all Fields" beside the chosen function.

Additionally, an option is provided to automatically include into the document all file definitions that are used by a chosen function. To do this enter a "Y" in the column headed "Include all Files" beside the chosen function.

When you have completed using this facility, use the Cancel function key to return to the "Review/Change Selected Objects" Menu.

7.8 Include Other Documents into the Document

When the option to "Include other documents" is chosen from the "Review/change selected Objects" Menu a screen similar to this example is used to include other documents into the document being worked with.

DC@P840110 Include Other Documents

Enter details of any other documents to be included in this document:

From Document From Folder

F1=Help F3=Exit F12=Cancel F14=Msgs

This facility allows you to include into the document contents, other documents. Only final form IBM i documents can successfully be included.

+

Only new "other documents" can be added to the list via this facility. Objects that are already in the list cannot be reviewed, changed or deleted via this facility. To do this use the "Review/exclude objects in document" facility that is

described in a later section of this chapter.

To specify details of other documents, key in the document and folder name and press enter. No validation is carried out on the names entered.

If all required "other documents" will not fit onto one screen use the roll up and roll down keys to scroll backwards and forwards through the list of "other documents".

When the object details are accepted the screen will be re-displayed requesting that more "other document" details are input. Accepted details are not redisplayed and cannot be reviewed or changed via this facility. Refer to 7.9 Review/Exclude Objects in the Document described in a later section of this chapter.

7.9 Review/Exclude Objects in the Document

When the option to "Review/exclude objects in document" is chosen from the "Review/change objects in document" Menu a screen similar to this example is used to manipulate the contents of the document.

DC@P840111 Review/Exclude Objects in Document

Drop	Object/Typ	e/Library or	Process/Description
1	J J I		1

_	AB\$OPT /FIELD /	/Action Bar Option
_	ADDRESS1 /FIELD /	/Street No and Name
_	ADDRESS2 /FIELD /	/Suburb or Town
_	ADDRESS3 /FIELD /	/State and Country
_	COMMENT /FIELD /	/Comment on skills acquired
_	DATE /FIELD /	/Numeric date in installatio
_	DATEACQ /FIELD /	/Date Skill Acquired (DDMMYY
_	DEPTDESC /FIELD /	/Department Description
_	DEPTMENT /FIELD /	/Department Code
_	EMPNO /FIELD /	/Employee Number
_	FULLNAME /FIELD /	/Employee full name
_	FUNCTION /FIELD /	/Current function name
_	GIVENAME /FIELD /	/Employee Given Name(s) +

F1=Help F3=Exit F12=Cancel F14=Msgs

The screen displays details of all objects currently included in the document contents.

A displayed document object list may not fit on one screen. This is indicated by a "+" sign in the lower right of the screen. In such cases use the roll up and roll down keys to scroll backwards and forwards through the displayed document

object list.

To delete an object from the list, enter a "Y" beside the object in the column headed "Drop" and press enter. The object is dropped and the new revised list is re-displayed.

7.10 Review/Change Table of Contents

When the option to "Review/change Table of Contents" is chosen from the "Review/change objects in document" Menu a screen similar to this example is used to review, omit or reorder the sequence of subjects to be documented.

DC@P840112 Review/Change Table of Contents

Omit, or reorder subjects, then press Enter.

Order/O	mit Subject
5	End user description of process XXXXXXXXXX
_10	Technical description of process XXXXXXXXXX
<u> 15</u>	Menu layout of process XXXXXXXXXX
_20	Summary of functions used by process XXXXXXXXXXX
_25	Summary of files used by process XXXXXXXXXX
<u> 30</u>	Summary of fields used by process XXXXXXXXXXX
<u> 35</u>	End user description of function XXXXXXXXXX
_40	Technical description of function XXXXXXXXXXX
<u> 45</u>	Panel layouts used by function XXXXXXXXXX
_50	MSL Diagram for function XXXXXXXXX
<u> 55</u>	Summary of files used by function XXXXXXXXXX
<u> 60</u>	Summary of fields used by function XXXXXXXXXX
<u> 65</u>	Call cross referencing for function XXXXXXXXX +

F1=Help F3=Exit F12=Cancel F14=Msgs

This screen displays the subjects which will be included into the document for the selected processes and functions, in the order in which they will appear. It is possible to omit or reorder the subjects into the desired sequence of subjects. Any "other documents" which have been entered will also be shown in the Table of Contents. Files and fields which have been selected will not appear in the Table of Contents, as they will always appear in an Appendix in alphabetical order at the end of the document.

On the first review of the Table of Contents, the selected processes and functions will ordered into a pre-set sequence of subjects. Each process will be followed by any of its associated functions, and any other documents to be included will appear following processes and functions. The pre-set sequence of subjects for processes is:

- Technical Description of the process
- End User Description of the process
- Menu layout of the process
- Summary of functions for the process
- Summary of all files used by all functions within the process
- Summary of all fields used by all functions within the process

The pre-set sequence of subjects for functions being documented is:

- Technical Description of the function
- End User Description of the function
- Panel layouts used by the function
- MSL diagram for the function (Major Structure and Logic Diagram)
- Summary of all files used by the function
- Summary of all fields used by the function
- Call cross referencing for the function
- Report layouts for the function

To omit a subject, either blank out the order/omit field or enter "9999" into the order/omit field. To reorder a field, enter the desired sequence number. When enter is pressed, the Table of Contents will be redisplayed in the correct sequence, with omitted subjects at the end. Omitted subjects can be re-selected by entering a sequence number into the order/omit field.

If new objects are added to the list of objects to be documented following the first review of the Table of Contents, they will appear at the beginning of the Table of Contents so that they can be reordered into the correct sequence if desired.

7.11 Submit a Document Production Run to Batch

When option 7 to Produce a Document has been entered beside an existing document a prompt screen like this will result;

DC@P840104 Submit Production Run to Batch					
Document: XXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXX					
Submit this job to batch YES Using Job name LANSADOC Job description QGPL/QDFTJOBD Job queue *JOBD Output queue *JOBD Copy to PC or RFT Folder/Document name					
OR Spool document for printing YES					
Length - lines per page					
Lines per inch $\dots \dots \dots$					
Last detail line number					
Formtype					
Hold spooled file					
F1=Help F3=Exit F12=Cancel F14=Msgs					

This document production run confirmation screen requests that details of the actual document production job (which will run in batch) be provided. The details required to successfully run the document production job are:

Submit This Job To Batch Using Job Name Job Description Job Queue Output Queue

Specify Copy to PC or RFT Document

Use this field to specify if the document should be copied into a PC document, or an RFTDCA (Revisable Form Text Document Content Architecture) document. If the document is to be spooled, leave this field blank. Specifying RFT will allow the document to be edited via OfficeVision/400.

Copy to PC or RFT Folder/Document Name

Use this field when you want the generated document to be copied into a PC document or an RFTDCA document. It is not possible to both spool the document for printing and copy the generated document to a PC or RFTDCA document. Specify the folder name and the document name to receive the generated document, otherwise leave blank. Note - for copy to PC documents if an existing document name is specified, the contents will be replaced by the generated document. If the folder name entered is not valid, the copy will fail. For copy to an RFTDCA document the document name entered must not exist.

OR Spool Document for Printing

YES Spool the generated document for printing.

NO Do not spool the generated document for printing.

Printing Options

The defaults for the documentor printing options will be set in the documentor default printing options definition data area, DC@A06. They can be changed on submitting a document production run.

Refer to 7.12 Documentor Default Printing Options -DC@A06 Configuration later in this chapter for more information on how to define the printing options in the printing options definition data area DC@A06.

Page Length

Mandatory. Number of lines per page to be used by the device file. Must be in range 40 to 255.

Page Width

Mandatory. Positions per line to be used by the device file. Must be in range 1 to 378.

Lines per Inch

Mandatory. Specifies the line spacing setting on the printer, in lines per inch, to be used by the device file. The possible values are: 4, 6, 8 or 9

Characters per Inch

Mandatory. Specifies the printer character density in characters per inch. The possible values are: 10, 12, 15 or 16.7

Last Detail Line Number

Mandatory. Specifies the line number on the page on which the last detail line will be printed prior to any specified footers and headers. The value specified must not exceed the page length value, and must be at least five less than the page length value to accommodate the footer lines.

Formtype

Specifies the type of forms used in the printer.

Number of Copies

Mandatory. Specifies the number of copies to print of the spooled file. Must be in range 1 to 255.

Hold Spooled File

Mandatory. Specifies whether the spooled file is held or not. The possible values are: YES - The spooled output file is held until it is released. NO - The spooled output file is not held by the output queue.

7.12 Documentor Default Printing Options -DC@A06 Configuration

The default printing options data area DC@A06 has been created specifically for documentor services and is broken down as follows:

From Byte	To Byte	Byte Length	Туре	Description
1	3	3	А	Spool document for printing YES or NO
4	6	3	S	Default printer page length (40 - 255)
7	9	3	S	Default printer page width (1 - 378)
10	10	1	S	Default lines per inch value (4 6 8 9)
11	14	4	А	Default characters per inch ("10" "12" "15" "16.7")
15	17	3	S	Default overflow line number (1 - 255) (Last detail line number will be this value minus one)
18	27	10	А	Default form type
28	30	3	S	Default number of copies (1 - 255)
31	33	3	А	Hold spooled file YES or NO

7.13 Questions and Answers

Q: How is LANSA Documentor turned on or off?

Documentor is turned off or on at the LANSA partition level.

It is recommended that you only enable Documentor in development partitions and do not enable it in other partitions.

Q: Will using Documentor use up more disk space?

Yes. Once documentor is enabled every compile will build up and store more information (e.g.: screen panel layouts). The storage of this information of course requires more disk space. However you should find that the storage of, for example, screen panel layouts does not require a lot more disk space than the storage of the screen panel layouts in some other facility (e.g.: OFFICE/400 folders).

Q: Can I estimate how much more disk space will be used?

This is very difficult because it is so application dependent. For instance 100 online functions have many screen panels, and thus use up more space than 100 batch functions.

As an example, consider the PSLSYS demonstration process shipped with LANSA. It consists of 8 functions which are heavily biased to online transactions and have many screen panels. The storage of the compiled objects (i.e. RPG programs) resulting from this system requires around 800K of disk space (in non-GUI mode). The storage of Documentor details for this system requires about 300K of disk space. So in this type of environment you could generalize that we need around 100K for each program object and around 40K to store complete documentation, so using documentor increases the storage required for program objects by about 40%.

Note that this increase is for program objects only. So if you allow 20Mb to store all the programs in your application, you should allow 28Mb if full documentation details are also required. This of course is a simple generalization and may not be applicable to all sites, particularly those that have a higher mix of batch functions, thus significantly reducing the figure down from 40%.

The use of documentor has no impact on the size required to store data objects, which is where, in most applications, the majority of disk space is utilized.

Q: Will using Documentor increase compile times?

Yes, but only by very slight amounts. Even this affect can be removed by submitting non-final compiles with a request that Documentor details should not be updated.

Note that when a compile is submitted like this all existing documentor details for the function are deleted, but not replaced with the new details. This prevents you from producing documents with details that are out of date.

Q: How should I approach using Documentor for the first time?

This is a recommended way of approaching Documentor for the first time:

Choose (or create) a very small application consisting of less than five database files and three RDML functions. Enable Documentor and then (re)compile the entire application to ensure that all details are stored.

- Create User and Technical help text for the processes and functions involved. Note: help text enhancements such as reverse image/underline/blinking etc will not be carried into documentor due to the difficulty in consistently representing these features when the document may be a spool file, a PC document or an RFTDCA document.
- Create 2 documents. One that is to document a "test case" for end users and the other for technical people. Initially include all subjects in the Table of Contents for both documents.

Produce one of the documents and examine it. Do not worry if a very large amount of documentation is produced. This is the approach of Documentor very large amounts of information may be produced - but you can be selective about what you actually want to produce.

- Put aside time to experiment with adding and removing subjects, page sizes, page layouts, output formats, etc, etc, as well as the effect of how details input into LANSA such as RDML functions, User Help Text, Developer Help Text, etc have on the final output document.
- When all experiments are complete, define a site standard for what should be included in various types of documents (e.g.: *User Guide* and *Technical Reference Guide*) and for the layout and format of the guides. This site standard may go as far as copying the completed document into a word processing or document handling system on a PC for final "polishing" into a format that makes the best use of the level of publishing facilities that your site has available.

The site standard may even include "post production" manipulation with specific site developed software to (re)format into a site specific format, replace special site "triggers" in the document, etc, etc.

Q: How and when are screen and report layouts captured?

Process Menu or Action Bar Layouts are captured and stored at the time that a process is compiled.

RDML Function panel layouts are captured and stored at the time that a function is compiled.

Report layouts, by their variable nature, cannot be created and stored automatically to produce a "sensible" image. You will find that every time that you exit from the report painter you will be given an option to save the image as a Documentor "sample" of the report layout. So to store a report layout for use by Documentor you should use the report painter to paint a sensible "sample" of the report, and then when exiting, request that the "sample" be saved as a report image for use with Documentor.

Q: What are some guidelines for getting the best from Documentor?

These guidelines are recommended for getting the best results from Documentor:

- It is a good idea to try to include all the objects to be documented prior to reviewing the Table of Contents (which always must be done at least once). The included objects will then appear with their subjects in the standard preset order of subjects the first time that the Table of Contents is reviewed. Once the Table of Contents has been reviewed, any further objects being included into the document will be added to the start of the Table of Contents.
- Use modular design techniques to create small RDML functions of less than 200 lines. Whenever possible use "smart" solutions such as virtual fields or system variables in preference to RDML function defined logic.
- Define all fields in the data dictionary whenever possible. Where this rule cannot be followed, function level field definitions should always carry a DESC parameter. For example:

DEFINE FIELD(#LPRODUCT) REFFLD(#PRODUCT) DESC('Last Product Number Processed')

Q: What are "MSL diagrams"?

MSL Diagrams are "Major Structure and Logic" Diagrams that are meant to represent the major/significant flow of an RDML function in a style that does not require specific RDML skills or knowledge.

For instance RDML skills usually require some knowledge of the English language (e.g.: the words SELECT, DISPLAY or CHANGE), but an MSL

diagram can represent RDML logic in other languages.

MSL diagrams work best in highly modular systems that make use of "smart" facilities like virtual fields and system variables, rather than in systems that heavily rely on RDML function level definitions of complex logic.

Q: What should MSL Diagrams be used for?

MSL diagrams are intended to demonstrate the "intent" of a function to someone working at the analyst or designer level, rather than the "detail" of a function to an application builder.

For example, a system analyst/designer may use an MSL diagram to work out how a whole new category of products should be introduced into an existing order entry system. This is the correct level of detail at which MSL diagrams should be used because the designer is trying to understand (for extension) the "intent" of the function at a business or end user level, rather than the detail of the function at the RDML level.

Q: What should I do if I want Documentor changed or enhanced?

More than any other thing, the production of documentation varies enormously from organization to organization. The variation is most likely to be found in the way that the "final form" of the document is produced, and the printing hardware that is being used to produce it.

Should you feel that a change to what Documentor does could be beneficial to your organization please forward your comments and suggestions to your product vendor.

Q: What is the difference between User and Developer Help Text?

The delivery of Documentor coincides with an extension to the existing help text facilities. Formerly only help text for display to "end users" could be defined for fields, processes or functions. However, the help text facilities also now support help text for technical people (i.e. developers).

This new help text facility is designed to allow help or descriptive text for technical people to be associated with fields, processes or functions. This help text is only displayed to users who enter LANSA with the DEVELOPER(*YES) option.

However, its intended use goes beyond displaying it to technical people. It is intended to form the basis upon which technical "free format" documentation for fields, processes and functions is produced by Documentor.

Note: Help text enhancements, such as reverse image/underline/ blinking etc., will not be carried into documentor due to the difficulty in consistently

representing these features when the document may be a spool file, a PC document or an RFTDCA document.

Q: How are User and Developer Help text input?

Both types of help text are input by using the existing help text facilities. They are simply separated from one another by the special \$\$TECH keyword. This allows the User and Technical help text details to be reviewed and changed together.

Note: Help text enhancements, such as reverse image/underline/ blinking etc., will not be carried into documentor due to the difficulty in consistently representing these features when the document may be a spool file, a PC document or an RFTDCA document.

Q: When I export information, are the Documentor details also exported?

Only if you request that they should be.

Q: Are Documentor Details imported even if the target system does not have documentor turned on?

Yes, if the exporting system requested that they should be exported.

Q: Can I export LANSA Document definitions?

No, the current version of Documentor does not allow the definition of a document to be exported.

8. LANSA PC Development

These topics are described in this chapter:

8.1 Define Personal Computers to LANSA

8.2 Repository Synchronization

8.3 Work with Repository Groups

8.4 Work with Work Groups

8.5 Propagating Objects from the IBM i

8.1 Define Personal Computers to LANSA

This section deals with the way in which Personal Computers running LANSA are defined to the LANSA system on the IBM i.

Note that on the PC side, when installing LANSA on the PC, the PC will be given a (permanent) node name. This name is used as the identifier of the PC on the IBM i. Therefore, this name must be a valid IBM i name.

Also note that data queues used for Visual LANSA are deleted and then recreated during LANSA Reorgs. This is done to free the space previously occupied by data on the data queues (clearing a data queue does not free the space used up by data on it). Please make sure that no data waiting to be processed is on the data queues when submitting a reorganization of the LANSA database.

Note that the LANSA REORG job addresses only data queues that are currently on file. Data queues deleted from the LANSA file due to a change of name or explicit delete of the PC definition still exist and should be deleted manually from the Data Queue In and Data Queue Out Libraries. These libraries should be deleted if they are no longer required as they may take up variable amounts of space even if they contain no data.

If you are working with multilingual systems, please review Handling Multilingual Text in the LANSA Multilingual Application Design Guide.

8.1.1 Maintain the PC Definition File

To be able to run Visual LANSA, it first has to be configured on the IBM i so that communication can be established between the PC and the LANSA system on the IBM i.

To maintain PC definitions on the IBM i, enter the command

LANSA REQUEST(PCMAINT)

The Work with PC Definitions screen is displayed:

DC@P642101 Work with PC Definitions

Type options, Press Enter. 2=Change 3=Copy 4=Delete 5=Display 17=Last Action Details

Opt PC Name	Description
JOHNSPC	PC definition for John
FREDSPC	PC definition for Fred
JACKSPC	PC for JACK

F1=Exit F3=Exit F6=Add/Create F8=Export Def F10=PC Users F12=Cancel F14=Messages F16=Repo Grps F18=Work Grps

Working from the Work with PC Definitions screen it is possible to:

- Export System Definitions. To do this, use the Export Def function key to get to the request screen for exporting definitions to PC.
- Work with PC Users. To do this, use the PC Users function key to get to the Work with PC Users screen.
- Work with Repository Groups. To do this, use the Repo Grps function key to

get to the Work with Repository Groups screen.

- Work with Work Groups. To do this, use the Work Grps function key to get to the Work with Work Groups screen.
- Add new PC Definitions. Use the Add/Create function key to get to the entry screen for new PC definitions.
- 2=Change: Allows changes to be made to an existing PC definition. If this option is entered against any of the PC Definitions shown on the screen, the Change PC Definitions screen is displayed next.
- 3=Copy: This option can be used only in conjunction with the create option. To create a new PC Definition by copying an existing PC definition, specify option 3 (to copy) beside the existing list, and press the Add function key.
- 4=Delete: Deletes an existing PC definition. A confirmation screen for the delete request is displayed. Enter 'YES' on this screen to confirm the delete request. After pressing Enter, the PC definition will be deleted immediately. **Note** that no security exists over PC definitions. Any user authorized to use this facility can delete any PC definition.
- 5=Display: Allows the display of a PC definition. If this option is entered against any of the PC definitions shown on the screen, the Display PC Definition screen is displayed.
- 17=Last Action Details: indicates that you wish to review the last action details recorded for the PC definition.

Enter this option against any existing PC definition shown on the display. An additional screen will appear showing the last action details. Refer to Last Action Details in the *LANSA Overview* for information about the details displayed.

8.1.2 Create PC Definitions

To create new PC definitions, use the Add/Create function key on the Work with PC Definitions screen.

This will cause the Create PC Definition screen to be displayed.

DC@P642103	Create PC Definition
PC Name PC Description	····
Data Queue In	
Data Queue Out	•••••
Monitor Program	
Job Description	<u>*LIBL/QBATCH</u>
Job Queue	<u>*LIBL/QINTER</u>
Output Queue	<u>*LIBL/QPRINT</u>
Translation	
Table In	<u>QSYS/ANSEBC1140</u>
Table Out	<u>QSYS/QASCII</u>
Submitted Jobs	
Job Description	<u>*LIBL/QBATCH</u>
Job Queue	<u>*LIBL/QBATCH</u>
Output Queue	<u>*LIBL/QPRINT</u> More
F1=Help F3=Exit F14=	=Messages

The details on the Create PC Definition screen should be entered or modified according to these guidelines:

PC NameMust be a valid IBM i name. This is the node name that you gave your PC when Visual LANSA was installed.

PC Description

A brief description for the PC.

Data Queue In & Out

The name you specify for "Data Queue In" or "Data Queue Out" is used to form the name of an IBM i library. The name is formed by appending the letters _DQL to the name you specify. The resulting name is then truncated to form at most a 10 character IBM i library name.

Thus, if you specify TESTIN, the resulting library name would be TESTIN_DQL. If you specify name TESTWORK then the resulting name would be TESTWORK_D. If you specify name TESTWORKIN then the resulting library name would be TESTWORKIN.

The portion of any queue library name that you specify:

- Must be different to any other input or output queue name already specified for any other workstation (including the one that you are currently defining).
- Must be a valid IBM i object name.

The library name (and the required objects within it) will be automatically created and initialized when the workstation starts a Host Monitor session or when you next do a LANSA database REORG operation.

Generally the libraries do not need to be backed up.

If the queues within the library become unusable you can rectify the problem by:

• Running a LANSA database REORG

or

• Shutting down the Host Monitor of the workstation involved, clearing or deleting the libraries and then restarting the Host Monitor.

Either of these solutions will of course lose any information currently held in the queues within the libraries.

Monitor Program

The Job Description, Job Queue and Output Queue entered here will be the ones used for the LANSA Host Monitor Program to run. They must be entered in IBM i format.

Translation Table In

This is the table used by LANSA to translate data received from the PC (ASCII) to the format used on the IBM i (EBCDIC).

If you are creating multilingual applications please read the Note for All Users regarding this entry.

Translation Table Out

This is the table used by LANSA to translate data from the format used on the IBM i (EBCDIC) to the format used on the PC (ASCII).

If you are creating multilingual applications please read the Note for All Users regarding this entry.

Submitted Jobs

The Job Description, Job Queue and Output Queue entered here will be the ones used by the LANSA Host Monitor Program to submit jobs. They must be entered in IBM i format.

After entering the details, 'page down' should be pressed to display the second part of the screen.

DC@P642103	Create PC Definition
Allowable Partitions .	
F1=Help F3=Exit F14	=Messages

Allowable Partitions

This screen allows the specification of partitions allowed for the PC to sign on

to. Enter up to 50 unique identifiers to allow access to these partitions. If the PC tries to sign on to a partition that is not specified here, the LANSA Host Monitor job on the IBM i will terminate.

After pressing Enter, the data will be stored on file. At the same time, the inbound and outbound libraries of data queues are created. Also, a data area with the same name as the inbound data queue library is created. These three sets of objects are required for running the LANSA Host Monitor.

8.1.3 Change PC Definitions

To change existing PC definitions, enter option 2 against a PC definition displayed on the 'Work with PC Definitions' screen.

This will cause the Change PC Definition screen to be displayed.

DC@P642102	Change PC Definition	
PC Name	<u>FREDSPC</u>	
PC Description	<u>PC definition for Fred</u>	_
Data Queue In	<u>FREDSQIN</u>	
Data Queue Out	<u>FREDSQOUT</u>	
Monitor Program		
Job Description	<u>*LIBL/QBATCH</u>	_
Job Queue	<u>*LIBL/QINTER</u>	
Output Queue	<u>*LIBL/QPRINT</u>	-
Translation		
Table In	QSYS/ANSEBC1140	
Table Out	<u>QSYS/QASCII</u>	
Submitted Jobs		
Job Description	<u>*LIBL/QBATCH</u>	_
Job Queue	<u>*LIBL/QBATCH</u>	
Output Queue	<u>*LIBL/QPRINT</u>	More

F1=Help F3=Exit F14=Messages

The details on the Change PC Definition screen can now be changed to meet the new requirements. The same conditions apply as for the Create option. To maintain the allowable partitions for the PC page down, as for the Create option. After pressing Enter, the data will be stored on file. At the same time, if they do not exist already, the inbound and outbound data queues are created in library QGPL, as well as a data area with the same name as the inbound data queue.

Data queues that are no longer required should be deleted manually from library QGPL, as they may take up variable amounts of space even if they are empty of data. Clearing a data queue does not free the space used by data on it.

Note for All Users

When exporting from LANSA on the IBM i to Visual LANSA on Windows, the export program looks first in PCMAINT for a PC definition called WIN_DFT. If it does not find it, it looks sequentially through the PC definitions until it finds the first definition that does not use the QASCII/QEBCDIC translation tables. Note that it will read from the definitions file, so the order searched may not match the alphabetical order displayed in the PCMAINT screen, unless a REORG has been performed since the last PC definition addition.

If no translation table can be found, then it will use the default translations tables EBC1140ANS and ANSEBC1140. These are supplied by LANSA.

The Visual LANSA install lists a number of translation tables for most major languages. All these translation tables are pre-installed on the IBM i when LANSA is installed. The list is constantly being added to.

If the shipped translation tables do not meet your needs, you can modify existing PC definitions in PCMAINT to use your own tables for the Host Monitor or modify/create the WIN_DFT PC definition.

Multilingual Users

In V11 SP5, the tables QEBCDIC/QASCII are replaced by these new default translation tables: ANSEBC1140/EBC1140ANS.

CCSID 1140 is identical to CCSID 037 except that the Euro is now included.

CCSID 1140 is identical to CCSID 037 except that the Euro (€) replaces ¤.

The entire Windows 1252 character set is supported, except for the following characters:

Characters	Hex Range	Explanation
'f"†‡^ ‰Š‹OEŽ"""* ~(tm)š›oežŸ	0x82 to 0xA0 inclusive	These characters do not exist in the IBM i 1140 CCSID, nor 037 CCSID
¤	0xA4	The Euro replaces this character

The full set of characters supported is:
€!#\$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ [\]^_`abcdefghijkImnopqrstuvwxyz{|}~j¢£¥¦§¨©ª≪¬®¯°±²³´μ¶·,¹°»¼½¾¿ ÀÁÂÃĂÅÆÇÈÉÊËÌÍĨÌĐÑÒÓÔŐŎרÙÚÛÜÝÞßàáâããåæçèéêëìíîìðñòóôőö÷øùúûüýþÿ

8.1.4 Export System Definitions

To export System Definitions, use the Export Def function key on the 'Work with PC Definitions' screen.

This will cause the Export definitions to PC screen to be displayed.

DC@P642105	Export definitions to PC
Export to shared	folder named DEF_SYS
Perform Export of Global LANSA Current Partition RDML Comman Application Tem Built-In Function Enrolled PC Nar Enrolled PC Use Current Task Lis	f : System Definitions . Y Definition Y ds Y plates Y ns Y nes Y rs Y t Y
Message Files	Name *LIBL Lang Int. Name DC@M01 ENG DC@M01
F1=Help F3=Exit	F12=Cancel F14=Messages F17=System LAD F18=Partn.

The details on the Export definitions to PC screen should be changed to meet the requirements. These rules apply:

Export To Shared Folder Named

Must be a valid IBM i name. This name defaults to DEF_xxx where xxx is the

identifier of the partition specified on the LANSA REQUEST(PCMAINT) PARTITION(...) command. This name identifies the shared folder to which the data selected below will be sent. If it does not exist, it will be created by the export job. Also, the submitted export job will have this name.

Global LANSA System Definition

Export the information contained in the LANSA system definition data area DC@A01, DC@A07 and DC@OSVEROP to the PC

Current Partition Definition

Export the definition of the LANSA partition that was specified on the LANSA REQUEST(PCMAINT) PARTITION(...) command. If no partition was specified, the default of SYS is used.

Frameworks and groups are included in the partition details which are exported. If flag *NOL4WCOMP is in system data area DC@OSVEROP, they are not exported but a message will printed in the export report.

Refer to Version Dependency Data Area DC@OSVEROP for more information.

RDML Commands

Export the LANSA RDML command definition tables. These are required for the editing and full function checking of LANSA functions on the PC. Normally these tables need to be exported only when Visual LANSA is first set up or when a new release is installed.

Application Templates

Export the Application Templates of the IBM i LANSA system to the PC.

Built-In Functions

Export the Built-In Functions of the IBM i LANSA system to the PC.

Enrolled PC Names

Export the PC definitions that are maintained on the previous screen. This information needs to be exported when Visual LANSA is first installed and whenever a PC definition's data queue names are changed.

Enrolled PC Users

Export Users to the PC. This exports the table that relates PC User names to IBM i user names, group names and their passwords. For information about this table, refer to 8.1.5 Work with PC Users.

Current Task Lists

Export all tasks currently defined in LANSA for i to the PC.

Message Files

Up to 10 message files can be specified for export from the IBM i to the PC. The details should be entered as follows:

Message File Name

The name of the message file as it is 'known' to the IBM i. The program checks that this message file exists in the library specified when it is entered here. The first message file name is prefilled with 'DC@M01'.

Message File Library

The name of the library that the message file resides in on the IBM i. It is always *LIBL. This is set to *LIBL, to indicate that the library list should be searched to locate the message file.

Language

The language identifier entry fields are shown only if the current partition is a multilingual partition. The language that the specified message file is associated with should be entered here. The language must be valid for the current partition. The language for the first message file is pre-filled with the current language if the current partition is multilingual.

Message File Internal Name

The message file internal name entry fields are shown only if the current partition is a multilingual partition. The message file name as it is used in LANSA applications should be entered here. The program does NOT check the validity of this name, so care should be taken to avoid spelling errors. The first message file internal name is pre-filled with 'DC@M01' if the current partition is multilingual.

Example for an export to the PC in a multilingual partition:

Name Library Lang Int. Name DC@M01 (English) DC@M01 *LIBL ENG DC@M01FRA *LIBL FRA DC@M01 (French) DC@M01ITL <u>*LIBL</u> ITL DC@M01 (Italian) USERMSG1 (English) USERMSG1 *LIBL ENG USERMSG1 (Italian) USERMSG1ITL *LIBL ITL USERMSG1DEU *LIBL USERMSG1 (German) DEU

In this example, DC@M01 will be exported to the folder as DC@M01, the messages being associated with the English language.

DC@M01FRA will be exported to the folder as DC@M01, the messages being associated with the French language.

DC@M01ITL will be exported to the folder as DC@M01, the messages being associated with the Italian language.

USERMSG1 will be exported to the folder as USERMSG1, the messages being associated with the English language.

USERMSG1ITL will be exported to the folder as USERMSG1, the messages being associated with the Italian language.

USERMSG1DEU will be exported to the folder as USERMSG1, the messages being associated with the German language

8.1.5 Work with PC Users

The "Work with PC users" facility allows the IBM i User Profile and Password corresponding to a Visual LANSA user to be specified.

To reach the Work with PC users screen, use the Users function key on the 'Work with PC Definitions' screen.

This will cause the Work with PC users screen to be displayed.

DC@P642107 Work with PC users

Type options, Press Enter. 2=Change 4=Delete 5=Display 17=Last Action Details System Name APPN.SYDAS400 Opt PC User ID AS/400 User ID AS/400 Group ID Kerberos User _____ JOHN WAYNEJOHN QPGMR ____ FRED ASTAIRFRED QOTHPRDOWN YES _____ JACK NICHOLJACK QOTHPRDOWN

F1=Help F3=Exit F6=Add/Create F12=Cancel F14=Messages

Working from the Work with PC users screen it is possible to:

- Add new PC Users. Use the Add/Create function key to get to the entry screen for new PC Users.
- 2=Change: Allows changes to be made to an existing PC User. The Change PC Users' screen is displayed when this option is entered against any of the PC Users shown on the screen.
- 4=Delete: Deletes an existing PC User. You will be asked to confirm your delete request. Enter 'YES' to confirm the delete request. After pressing

Enter, the PC User entry will be deleted immediately. **Note** that no security exists over PC Users. Any user authorized to use this facility can delete any PC User entry.

- 5=Display: Displays information about an existing PC user. The Display PC Users screen is displayed when this option is entered against any of the PC Users shown on the screen.
- 17=Last Action Details: indicates that you wish to review the last action details recorded for the PC user.
- The Display Last Action Details screen is displayed when this option is entered against any PC user shown in the display. Refer to Last Action Details in the *LANSA Overview* for information about the details displayed.

8.1.6 Add PC Users

To add new PC users, use the Add/Create function key on the 'Work with PC Users' screen.

This will cause the Create PC User screen to be displayed.

DC@P642109 Create PC User

System NameAPPN.DEVAS400PC User ID______AS/400 User ID_______

AS/400 Group ID: AS/400 Password Supplemental Group Profiles: GRP01 GRP02 GRP03 GRP45 GRPZZZZZZ3 GRPXXXXXX

F1=Help F3=Exit F14=Messages

System Name

This network attribute and is displayed here for information only. It is an outputonly field.

PC User ID

The ID that the user will use when signing on to the PC.

AS/400 User ID

The ID that will be used by a session on the PC to sign on to the IBM i. It must exist on the IBM i. It is stored encrypted.

AS/400 Group ID

The group ID that corresponds to the AS/400 User ID above. It will be obtained by LANSA from operating system details entered when the AS/400 User ID was recorded. It is stored encrypted.

AS/400 Password

The password that will be used by a session on the PC to sign on to the IBM i. It must be correct for the IBM i User ID. The password is not displayed. It is stored in an encrypted way.

After pressing Enter, the data will be stored on file. To be usable, it must be exported to the PC using the 8.1.4 Export System Definitions facility.

Note: If the IBM i password is configured to have long passwords, the maximum length is 128 and it is also case sensitive.

Supplemental Group Profiles

The supplemental group profiles that correspond to the User ID above. These will be obtained by LANSA from operating system details entered when the AS/400 User ID was recorded. There may be up to 15 supplemental group profiles.

Kerberos User (Windows credentials user)

Yes in the Kerberos User column indicates that this AS/400 user is set up to use the Single Sign On facility using the Kerboros technology. If not used, it will be blank.

For further information, refer to Single Sign-On (SSO) in the *Visual LANSA Administrator Guide*.

8.1.7 Change PC Users

To change existing PC Users, enter option 2 against a PC User on the Work with PC Users screen.

This will cause this screen to be displayed.

DC@P642108 Change PC User

System NameAPPN.DEVAS400PC User IDFREDAS/400 User IDASTAIRFRED

AS/400 Group ID: QOTHPRDOWN AS/400 Password Supplemental Group Profiles: GRP01 GRP02 GRP03 GRP45 GRPZZZZZZ3 GRPXXXXXX

F1=Help F3=Exit F14=Messages

System Name

A network attribute displayed here for information only. It is an output-only field.

PC User ID

The ID that the user will use to sign on to the PC.

AS/400 User ID

The ID that will be used by a session on the PC to sign on to the IBM i. Change

this User ID as required. It will be stored encrypted.

AS/400 Group ID

The group ID that corresponds to the AS/400 User ID above. It will be obtained by LANSA from operating system details entered when the AS/400 User ID was recorded

AS/400 Password

This password will be used by a session on the PC to sign on to the IBM i. It must be correct for the IBM i User ID. The password is not displayed. It will be stored encrypted.

After pressing Enter, the data will be changed on file. For the change to be effective, the data must be exported to the PC using the 8.1.4 Export System Definitions facility.

Supplemental Group Profiles

The supplemental group profiles that correspond to the User ID above. These will be obtained by LANSA from operating system details entered when the AS/400 User ID was recorded. There may be up to 15 supplemental group profiles.

8.2 Repository Synchronization

Repository groups are set up on the IBM i to reflect how non-IBM i repositories are connected to the IBM i repository, so that information can be automatically propagated from the IBM i to these repositories. The propagated information consists of changes to task tracking and partitions on the IBM i, updated information checked in to the IBM i by a member of a work group and user initiated requests from the main "work with" panels on the IBM i.

Note that changes to objects made by a user on the IBM i are not propagated. For example, changes to a function are not propagated.

A change is propagated by automatically sending a check out for read-only transaction to a PC. Note that if an object is checked out for update on the non-IBM i repository, the propagation still overwrites this object. Thus, it is recommended that only one user updates an object at any one time. Refer to Task Tracking for more information. A user on this non-IBM i repository would need to check the object out for update again in order to be able to modify it.

It should be remembered that checking out is always partition-specific, in other words objects are only checked out to the allowed partitions defined in the PC definition of the member. Please refer to the Visual LANSA online help for more detail on checking objects in and out of an IBM i.

8.2.1 Repository Groups

Configuration of repository synchronization requires repository groups to be created. A repository group is a list of all the PCs that use a non-IBM i repository. Thus a repository group is synonymous with the users of a non-IBM i repository.

The non-IBM i repositories can be either on a stand-alone workstation or on a server in a network configuration.

Because the repository groups reflect the actual configuration of the repositories, each PC can only be in one repository group.

How you set up repository groups depends on how the PCs and the repositories are connected to the IBM i repository:

- In the simplest case, you create a repository group for a single PC which has a repository (so called stand-alone workstation configuration).
- In a configuration with a network server containing a repository and several client PCs you create one repository group to include both the server and the PCs.

In a repository group, one PC must be identified as the repository gateway. This PC uses Host Monitor to receive the propagations and store them in the repository.

In a stand-alone configuration, the single PC is the gateway as there is only one PC in the repository group. In a network configuration, the PC chosen as the gateway must have the host monitor installed, but it does not have to be the PC with the repository. In this configuration, the gateway is often the server.

If the PC identified as the repository gateway is deleted without another PC being identified as the new repository gateway, then the first PC in the repository group is considered to be the repository gateway for the repository group.

A PC must be in a repository group in order for its repository to receive any propagations.

The Host Monitor must be running on the repository gateway for changes to the current partition to be received. Propagations are queued on the IBM i until the Host Monitor is started in the appropriate partition.

A fundamental part of repository synchronization is the IBM i. Without an IBM i, no propagations can occur. If Visual LANSA is configured in standalone mode - disconnected from any IBM i - it is also disconnected from all other

non-IBM i repositories.

In order to configure repository groups you need to answer at least the following questions:

• What non-IBM i repositories are there that need to take part in repository synchronization?

This gives you the repository groups, one for each repository.

- What PC will be the gateway for each repository? The gateway is the PC that will be sent the propagations for a particular repository. This is the first member of the repository group.
- What other PCs use each repository? This gives you the other members of each repository group.

Armed with this information you can now create your repository groups as described in 8.3.1 Add Repository Groups.

Provided you only need to add repository groups and not create PC definitions, a refresh of PC definitions is not required. The next Check In made from a PC in a repository group will be propagated to the other repositories.

A further refinement can be added to restrict which changes go to which repositories. This is referred to as a work group. When no work groups are defined, all propagations go to all repositories.

8.2.2 Work Groups

Configuration of repository synchronization does not require work groups to be created in order for changes to be propagated. When no work groups are defined, all propagations go to all repositories. A work group is a list of all the PCs that share the same work. The members of a work group can be in the same or different repository groups.

When a PC in a work group checks in objects to the IBM i using the host monitor, copies of all these objects are checked out to the repositories of the other members of the group. Where a work group contains two or more PCs defined in the same repository group, the propagations are sent only once to the repository gateway.

For example, the PCs of a group of developers working on financial applications for a company could form one work group so that they share the latest versions of objects.

A PC can be in many work groups or in no work groups at all. It does not have to belong to a repository group. A work group must contain at least two PCs.

If one work group exists, and the PC originating a change is not in a work group, the change is not propagated.

In order to configure work groups you just need to know which PCs share the same work and want to automatically receive changes that a co-worker makes.

If repository groups are set up but there are no work groups, all PCs are considered to be in one universal work group.

8.2.3 Options on PC

Options are available within the *Partition Maintenance* dialog on the PC to control the action of propagated deletes. These are:

- Ignore propagated deletes
- Perform checks before propagating.

Selecting the *Ignore propagated deletes* option will cause propagated deletes to be ignored.

Selecting the *Perform checks before propagating* option will cause propagated deletes to perform referential integrity checks before actioning the delete. This will prevent, for example, the deletion of a field that has been used on a local file.

8.3 Work with Repository Groups

The "Work with Repository Groups" facility allows PCs that share a common repository to be defined to LANSA for i, and a repository gateway for that common repository to be specified.

To reach the Work with Repository Groups screen, use the Repo Grps function key on the 'Work with PC Definitions' screen.

This will cause the Work with Repository Groups to be displayed.

DC@P642112 Work with Repository Groups

Type options, Press Enter.

2=Change 4=Delete 5=Display 17=Last Action Details

Opt Group Name Total PCs

 JOHNSPC	1
 SERVER1	5
 SERVER2	2

Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Cancel Fnn=Messages

Working from the Work with Repository Groups screen it is possible to:

- Add new Repository Groups. To do this use the Add/Create function key to get to the entry screen for new Repository Groups
- 2=Change: Allows changes to be made to an existing Repository Group. If this option is entered against any of the Repository Groups shown on the screen, the 'Change Repository Group' screen will be displayed next.
- 4=Delete: Deletes an existing Repository Group. A confirmation screen for the delete request will be displayed. Enter 'YES' on this screen to confirm the

delete request. After pressing Enter, the Repository Group will be deleted immediately. **Note** that no security exists over Repository Groups. Any user authorized to use this facility can delete any Repository Group.

- 5=Display: Allows the display of information about an existing Repository Group. If this option is entered against any of the Repository Groups shown on the screen, the 'Display Repository Group' screen will be displayed next.
- 17=Last Action Details: indicates that you wish to review the last action details recorded for the Repository Group. If this option is entered against any of the Repository Groups shown on the screen, the 'Review Last Action Details' screen will be displayed next. This screen has been described in the topic Last Action Details.

8.3.1 Add Repository Groups

To add new Repository Groups, use the Add/Create function key on the Work with Repository Groups screen.

____ _____

_____ ____+

This will cause the Create Repository Group screen to be displayed.

DC@P642114 Create Repository Group

Group Name

PC Names/Gateway Flag. . . .

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Group Name

This is the name of the Repository Group.

PC Names / Gateway Flag

These are the names of all the PCs that are in the Repository Group. Only one of the PCs can have the gateway flag set to Y. All other PCs must have the gateway flag set to blank or N.

8.3.2 Change Repository Groups

To change existing Repository Groups, enter option 2 against a Repository Group on the Work with Repository Groups screen.

This will cause the Change Repository Group screen to be displayed.

DC@P642116	Change Repository Group
Group Name	SERVER1
PC Names / Gateway	Flag
SERVER1 Y JACKPC	

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Group Name

This is the name of the Repository Group.

PC Names / Gateway Flag

These are the names of all the PCs that are in the Repository Group. Only one of the PCs can have the gateway flag set to Y. All other PCs must have the gateway flag set to blank or N.

+

8.4 Work with Work Groups

The Work with Work Groups facility allows PCs that share common work to be defined to LANSA for i.

To reach the Work with Work Groups screen, use the Work Grps function key on the Work with PC Definitions screen.

This will cause the Work with Work Groups screen to be displayed.

DC@P642113 Work with Work Groups Type options, Press Enter. 2=Change 4=Delete 5=Display 17=Last Action Details Opt Group Name Total PCs _____ PROJ1 2 _____ PROJ2 5 _____ PROJ3 2

Fnn=Help Fnn=Exit Fnn=Add/Create Fnn=Cancel Fnn=Messages

Working from the Work with Work Groups screen you can:

- Add new Work Groups. Use the Add/Create function key to display the entry screen for new Work Groups
- 2=Change: Allows changes to be made to an existing Work Group. The Change Work Group screen is displayed when this option is entered against any of the Work Groups shown on the screen.
- 4=Delete: Deletes an existing Work Group. You will be asked to confirm your delete request. Enter 'YES' to confirm the delete request. After pressing Enter, the Work Group will be deleted immediately. **Note** that no security exists over Work Groups. Any user authorized to use this facility can delete any Work Group.
- 5=Display: Displays information about an existing Work Group. The Display Work Group screen is displayed when this option is entered against any of the Work Groups shown on the screen.

• 17=Last Action Details: Indicates that you wish to review the last action details recorded for the Work Group. The Review Last Action Details screen is displayed when this option is entered against any of the Work Groups shown on the screen. Refer to the topic Last Action Details for information about the details displayed.

8.4.1 Add Work Groups

To add new Work Groups, use the Add/Create function key on the Work with Work Groups screen.

This will cause the Create Work Group screen to be displayed.

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Group Name

This is the name of the Work Group.

PC Names

These are the names of all the PCs that are in the Work Group.

8.4.2 Change Work Groups

To change existing Work Groups, enter option 2 against a Work Group on the Work with Work Groups screen.

This will cause the Change Work Group screen to be displayed.

DC@P642117	Change Work Group
Group Name	SERVER1
PC Names	•••
SERVER1 JACKPC	

Fnn=Help Fnn=Exit Fnn=Cancel Fnn=Messages

Group Name

This is the name of the Work Group.

PC Names

These are the names of all the PCs that are in the Work Group.

8.5 Propagating Objects from the IBM i

Option 33=Propagate is available on all of these advanced "work with " panels:

- Work with Fields
- Work with System Variables
- Work with Multilingual Variables
- Work with Files
- Work with Processes
- Work with Functions
- Work with Templates

Note that this option will only be visible if repository synchronization is enabled, that is, at least one repository group is defined.

When option 33=Propagate has been selected for an object, a panel similar to this one will be displayed:

DC_P644801 Propagate Object

Partition: XXX Object:XXXXXXXXX Extension: XXXXXXXXX Type:

Fnn=Help Fnn=Exit Fnn=Prompt Fnn=Cancel Fnn=Messages

The Propagate Object panel enables PC repositories to be updated with the latest definition of an object from the LANSA for i development environment. Typically this request to propagate would be done after changes have been completed. The next time that the host monitor is run for the PC repository or if it is currently running, the object will be checked out for read only.

The Partition, Object, Extension and Type define which LANSA object for which propagation has been requested.

Propagate to Work Group

To propagate the latest definition of the object to all Work Groups, or if there are no Work Groups defined, specify *ALL.

To propagate the latest definition of the object to a single work group, specify the name of the Work Group.

Prompt with blank or partial Work Group to select from a list of matching Work Groups.

9. Developer Services

This chapter details the Developer Services components available for your LANSA system and how each component can be accessed.

9.1 Developer Services Components

Access to the Developer Services components of LANSA always starts from the Developer Services menu. The Developer Services menu can be accessed from anywhere within LANSA simply by pressing the Messages function key.



Access to the "Work with Developer Messages" component of the Developer Services menu is dependent on the value of the ALLOW_MSGS parameter ("*YES" or "*NO") in the LANSA command.

If the value for the ALLOW_MSGS parameter is "*YES" then the "Work with Developer Messages" option will be available from the Developer Services menu:

LANSA ALLOW_MSGS(*YES)

If the value for the ALLOW_MSGS parameter is "*NO" then the "Work with Developer Messages" option will not be available from the Developer Services menu:

LANSA ALLOW_MSGS(*NO)

9.2 Developer Services Menu

The Developer Services facility can be invoked by pressing the Messages function key, the user will then be prompted with the Developer Services "Action Box" from which any of these options may be selected:

:	Developer Services	:		
:	Use cursor to select an option	:		
:	Display LANSA/Workstation M	Messa	ages	:
:	Work with Developer Message	S	:	
:	Execute an OS/400 command		:	
:	User defined exit program 01	:		
:	User defined exit program 02	:		
:	User defined exit program 03	:		
:	Return to previous screen	:		
:	Fnn=Help Fnn=Cancel	:		
:.	:			

Working from this screen you can select an option to:

- Display LANSA/Workstation messages allows you to display all the messages that have been issued by LANSA and any other messages that are currently in the message queue for the workstation that you are currently working.
- Work with developer messages allows you to create, change, review, delete, print and send developer messages. This option will only be available from the Developer Services menu if the ALLOW_MSGS parameter of the LANSA command is set to "*YES".
- Executing an OS/400 command allows you to execute a command via an IBM i command line. This option will only be available from the Developer Services menu if the DEVELOPER parameter of the LANSA command is set to "*YES" and the IBM i operating system is at Release 1.0 Version 3.0 (V1R3) or higher.

• User defined exit program 01 - 03 allows you to call up to 3 user defined developer services programs. This option will only be available from the Developer Services menu if the DEVELOPER parameter of the LANSA command is set to "*YES" and user defined developer programs have been specified in the extended information data area DC@A07.

9.3 Display LANSA/Workstation Messages

Display LANSA/Workstation messages allows you to display all the messages that have been issued by LANSA and any other messages that are currently in the message queue for the workstation that you are currently working.

The LANSA/Workstation messages will be displayed at line 22 of your workstation.

9.4 Work with Developer Messages (LANSA/Messages)

LANSA/MESSAGES is a stand alone facility which will be primarily available to system developers and system testers as the "Work with Developer Messages" option from the "Developer Services" menu.

LANSA/MESSAGES can be used to:

- Associate messages to any LANSA object, either temporarily or permanently.
- Exchange (Send & Receive) messages between developers and end users.
- Act as an electronic mail/notice board for LANSA objects that can be accessed by Developers and End Users.
- Notify users of existing messages when accessing an object.
- Replace the blizzard of memos, letters and scraps of paper exchanged during the application development cycle.

LANSA/MESSAGES can be run in multi-partition and multi-machine environments.

9.4.1 Work with Developer Messages Components

Access to the "Work with Developer Messages" components of LANSA always starts from the "Work with Developer Messages" menu which in turn can be accessed from the "Developer Services" menu.





Refer to 9.11 Developer Services Configuration for more information on the user defined "User Profile Validation" and "Send Developer Message" programs.

9.4.2 Build List of Developer Messages

Build a list of Developer Messages allows you to specify the type of developer messages you wish to appear on the "Work with Developer Messages" menu, whether it be all the developer messages relating to a user (Sent to or Sent from), a specific object (Field, File, Function or Process) or "SYSTEM" developer messages.

The "Build List of Developer Messages" facility will be invoked when the "Work with Developer Messages" option has been selected from the "Developer Services" menu and when the "Build list" Function key has been pressed on the "Work with Developer Messages" menu.

The example below shows what may result when the "Build List of Developer Messages" facility has been invoked:

Build List of Developer Messages Specify search criteria and press Enter Sent by User . . . _____ name, *ALLUSR OR Sent to User . . . ______ name, *ALLUSR OR For FIELD + OR For FILE _ + OR For PROCESS ... + AND/OR For FUNCTION . . . _ _ + OR For SYSTEM YES YES, NO Fnn=Help Fnn=Prompt Fnn=Create Fnn=Cancel

Only one search criterion may be specified, upon which a list of all developer messages matching the specified search criteria will be generated for use in the "Work with Developer Messages" menu.

Working from this screen you can build a developer message list by specifying one of the following:

- Sent by User must be a valid user/group profile name or "*ALLUSR".
- Sent to User must be a valid user/group profile name or "*ALLUSR".
- The "Sent by User" and the "Sent to User" values will be validated by the user defined "User Profile validation" program specified in the extended information data area DC@A07.
- Refer to 9.11 Developer Services Configuration in this chapter for more information on how to define a user defined "User Profile Validation" program in the extended definition data area DC@A07.
- For FIELD must be a valid field name that exists within the LANSA data dictionary. If you wish to perform a search for a field name then press the Prompt function key, a pop-up window will then be displayed providing you with the option to search for and select the required field name. If the current/last object being worked with is a/was a field, then the "For FIELD" field will be defaulted to the name of that object (field name).
- For FILE must be a valid file name that exists within LANSA. If you wish to perform a search for a file name then press the Prompt function key, a pop-up window will then be displayed providing you with the option to search for and select the required file name. If the current/last object being worked with is a/was a file, then the "For FILE" field will be defaulted to the name of that object (file name).
- For PROCESS must be a valid process name that exists within LANSA. If you wish to perform a search for a process name then press the Prompt function key, a pop-up window will then be displayed providing you with the option to search for and select the required process name. If the current/last object being worked with is a/was a process, then the "For PROCESS" field will be defaulted to the name of that object (process name).
- For FUNCTION must be a valid function name that exists within LANSA, if a Process name has also been specified then the function name must exist within the specified process. If you wish to perform a search for a function name then press the Prompt function key, a pop-up window will then be displayed providing you with the option to search for and select the required function name. If the current/last object being worked with is a/was a function, then the "For FUNCTION" field will be defaulted to the name of that object (function name) and the "For PROCESS" field will be defaulted to the extended name of that object (process name).
• For SYSTEM must be either "YES" or "NO". If no objects have been selected to work with in this session of LANSA, then the "For SYSTEM" field will be defaulted to "YES", if objects have been worked with the "For SYSTEM" field will be defaulted to "NO".

9.4.3 Work with Developer Messages Menu

Work with Developer Messages allows you to create, change, review, delete, print and send developer messages.

The "Work with Developer Messages" menu will give you access to all the developer messages that have met the selection criteria specified from the "Build list of Developer Messages" facility.

The example below shows what may result when the "Work with Developer Messages" option has been chosen from the Developer Services action box:

Work with Developer Messages Type options and press Enter 2=Review/Change 4=Delete 6=Print 7=Send Opt Subject Type Date Category

- _ Customer Number FIELD 27/07/91 Problem
- _ Sales Order Header FILE 27/07/91 Problem
- _ Order Entry FUNCTION 25/07/91 Question
- _ Release Announcement SYSTEM 18/07/91 Information

Bottom Fnn=Help Fnn=Exit Fnn=Build list Fnn=Create Fnn=Cancel

Working from the Work with Developer Messages screen you can:

- Create a new developer message.
- Review a specific developer message by making an entry of "2" against the developer message you wish to work with.
- Change a specific developer message by making an entry of "2" against the developer message you wish to work with.
- Delete a specific developer message by making an entry of "4" against the developer message you wish to work with.
- Print a specific developer message details by making an entry of "6" against the developer message you wish to work with.

- Send a specific developer message to the user(s) who has been designated to receive the message by making an entry of "7" against the developer message you wish to work with. A message will be sent via a user defined "Send Developer Message" program.
- Build list of developer messages that you wish to work with.

9.4.4 Create a Developer Message

Create a Developer Message allows you to create a new developer message. This example shows what may result when the "Create a Developer Message" option has been chosen from the "Work with Developer Messages" screen:

Create a Developer Message Message ID : 012345670000001DC@PGMLIB
From user : QOTHPRDOWN Date/Time : 27/07/91 12:22:33
To user name, *ALLUSR
Category P T
Type Type FIELD, FILE, FUNCTION, PROCESS, SYSTEM Object Name +
Extension +
More
Fnn=Help Fnn=Prompt Fnn=Cancel

Default Values when Creating a New Developer Message

These default values are automatically set by the system when a new developer message has been created:

Message Identifier

Is the unique identifier of the developer message. The message identifier is 25 characters containing 3 items, as in this example:

012345670000001DC@PGMLIB

where

• the first 8 characters represent the IBM i system value QSRLNBR of the

machine where the message originated.

- the next 7 characters contain the next message number to be allocated from the DC@A09 data area.
- the last 10 characters contain the LANSA program library name.

From User

Is the user profile name of the current user who has created the new developer message.

Date/Time

Is the date and time at which the new developer message was created.

Input Options When Creating a New Developer Message

These input options apply when creating a new developer message:

Subject

Is the brief description which relates to the developer message.

To User

Is the user profile name of the user to which the developer message should be "sent to" or marked as "for the attention of". Valid entries are as follows:

- A user profile name
- A group profile name
- "*ALLUSR" all users

The "To User" value will be validated by the user defined "User Profile validation" program which determines whether the user or group profile is valid.

Refer to 9.11 Developer Services Configuration later in this chapter for more information on how to define a user defined "User Profile Validation" program in the extended definition data area DC@A07.

Category

Is the category of the developer message. The category must be a valid category code from the category code table defined within the extended information data area DC@A07.

Refer to 9.11 Developer Services Configuration later in this chapter for more information on how to define a user defined "Category Code" table in the extended definition data area DC@A07.

If the category code's "Send Message Flag" is set to "Y" then the message will be sent to the user nominated in "To User" field by the user defined "Send Developer Message" program which is also specified in the extended information data area DC@A07.

If the category code's "Notify User Flag" is set to "Y" then the user will be notified of the developer message each time the object is accessed.

Perm/Temp

Is the flag that indicates whether the developer message is permanent or temporary. Valid entries are:

"P"= Permanent developer message

"T"= Temporary developer message

If a developer message is flagged as permanent, then it will remain on the system until it has been selected for deletion.

If a developer message is flagged as temporary, it will remain on the system until it has been purged, unless the developer message is changed to a permanent developer message.

Туре

Is the type of message or object for which the developer message relates to. Valid entries are:

FIELD	Message relates to a specific Field
FILE	Message relates to a specific File
FUNCTION	Message relates to a specific Function
PROCESS	Message relates to a specific Process
SYSTEM	Message is a System-wide developer message which is intended to be sent from one user to another user/s.

This field will be defaulted to the object type of the last object that was worked with.

Object Name

Is the name of the object for which the developer message relates to. If you wish to perform a search for an object then ensure the "Type" field contains a valid object type (FIELD, FILE, FUNCTION or PROCESS) and then press the Prompt function key, a pop-up window will then be displayed providing you with the option to search for and select the required object.

This field must be blank if the message type is set to "SYSTEM".

This field will be defaulted to the object name of the last object that was worked with.

Extension

Is the object extension name of the object for which the developer message relates to.

This field must contain an entry if the message type is set to "FILE" (file library name) or "FUNCTION" (process name in which the function resides).

This field must be blank if the message type is set to "SYSTEM", "FIELD" or "PROCESS".

This field will be defaulted to the object extension of the last object that was worked with.

Message Text

Is the message text for the developer message. Up to 200 lines of message text may be entered.

When you have completed entering the developer message details, you will be returned to the "Work with Developer Messages" screen.

9.4.5 Review/Change a Developer Message

Review/Change a Developer Message allows you to review or change an existing developer message.

The example below shows what may result when the "Review/Change a Developer Message" option has been chosen from the "Work with Developer Messages" screen:

Review a Developer Message Message ID : 01234567000001DC@PGMLIB From user : QOTHPRDOWN Date/Time : 27/07/91 12:22:33 Subject . . . : Release Announcement To user . . . : *ALLUSR Category . . . : INF Information Perm/Temp . . . : P Permanent Type : SYSTEM Object Name . . : Extension . . : Message Text Attention all users - the Order Entry will go LIVE on 25/07/91.

Bottom Fnn=Help Fnn=Cancel Fnn=Change Fnn=Delete

Working from the Review a Developer Message screen you can:

- **Review** the developer message details.
- **Change** the developer message details. To do this, use the CHANGE function key to place the screen into change mode and make any desired changes.
- **Delete** the developer message details. To do this, use the DELETE function key to place the screen into delete mode and confirm the deletion.

9.4.6 Change a Developer Message

Change a Developer Message allows you to change an existing developer message.

The example below shows what may result when the change key has been enabled from the "Review a Message" screen:

Make the required changes and press Enter.

Input Options when Changing a Developer Message

These input options apply when changing an existing developer message:

Subject

Is the brief description which relates to the developer message.

To User

Is the user profile name of the user to which the developer message should be "sent to" or marked as "for the attention of". Valid entries are as follows:

- A user profile name
- A group profile name
- "*ALLUSR" all users

The "To User" value will be validated by the user defined "User Profile validation" program which determines whether the user or group profile is valid.

Refer to 9.11 Developer Services Configuration later in this chapter for more information on how to define a user defined "User Profile Validation" program in the extended definition data area DC@A07.

Category

Is the category of the developer message. The category must be a valid category code from the category code table defined within the extended information data area DC@A07.

Refer to 9.11 Developer Services Configuration later in this chapter for more information on how to define a user defined "Category Code" table in the extended information data area DC@A07.

If the category code for this developer message has been changed to a category code that's "Send Message Flag" is set to "Y" then the message will be sent to the user nominated in "To User" field by the user defined "Send Developer Message" program which is also specified in the extended information data area DC@A07.

If the category code's "Notify User Flag" is set to "Y" then the user will be notified of the developer message each time the object is accessed.

Perm/Temp

Is the flag that indicates whether the developer message is permanent or temporary. Valid entries are:

"P" = Permanent developer message

"T" = Temporary developer message

If a developer message is flagged as permanent, then it will remain on the system until it has been selected for deletion.

If a developer message is flagged as temporary, it will remain on the system until it has been purged, unless the developer message is changed to a permanent developer message.

Туре

Is the type of message or object for which the developer message relates to. Valid entries are:

FIELD	Message relates to a specific Field
FILE	Message relates to a specific File
FUNCTION	Message relates to a specific Function
PROCESS	Message relates to a specific Process
SYSTEM	Message is a System-wide developer message which is intended to be sent from one user to another user/s.

Object Name

Is the name of the object for which the developer message relates to. If you wish to perform a search for an object then ensure the "Type" field contains a valid object type (FIELD, FILE, FUNCTION or PROCESS) and then press the Prompt function key, a pop-up window will then be displayed providing you with the option to search for and select the required object.

This field must be blank if the message type is set to "SYSTEM".

Extension

Is the object extension name of the object for which the developer message relates to.

This field must contain an entry if the message type is set to "FILE" (file library name) or "FUNCTION" (process name in which the function resides).

This field must be blank if the message type is set to "SYSTEM", "FIELD" or "PROCESS".

Message Text

Is the message text for the developer message. Up to 200 lines of message text may be entered.

When you have finished reviewing/changing the developer message details, you will be returned to the "Work with Developer Messages" screen.

9.4.7 Delete a Developer Message

Delete a Developer Message allows you to delete an existing developer message.

The example below shows what may result when the "Delete a Developer Message" option has been chosen from the "Work with Developer Messages" screen:

Delete a Developer Message Message ID : 012345670000001DC@PGMLIB From user : QOTHPRDOWN Date/Time : 27/07/91 12:22:33 Subject . . . : Release Announcement To user . . . : *ALLUSR Category . . . : INF Information Perm/Temp . . . : P Permanent Type : SYSTEM Object Name . . : Extension . . : Message Text <u>DELETE THIS MESSAGE</u> . . _ YES, NO Attention all users - the Order Entry will go LIVE on 25/07/91.

Bottom

Fnn=Help Fnn=Cancel

Working from the Delete a Developer Message screen you can:

• Delete the developer message details. To do this specify "YES" in the "Delete this Message" prompt to confirm the deletion.

Input Options when Deleting A Developer Message

These input options apply when deleting an existing developer message:

Delete This Message

A "YES" or "NO" entry is required. "YES" will delete the developer message, "NO" will prevent the developer message from being deleted.

When you have answered the "Delete this Message" question, you will be returned to the "Work with Developer Messages" screen.

9.4.8 Send a Developer Message

This option allows you to send a developer message.

A message will be sent via the user defined "Send Developer Message" program specified in the extended information data area DC@A07.

Refer to 9.11 Developer Services Configuration later in this chapter for more information on how to define a user defined "Send Developer Message program" in the extended information data area DC@A07.

9.5 Execute an IBM i Command

This option will only be available from the Developer Services menu if the DEVELOPER parameter of the LANSA command is set to "*YES".

When a command line has been requested, a screen similar to the one below will be displayed:

Type command, press Enter ===>

Fnn=Help Fnn=Exit Fnn=Prompt Fnn=Retrieve Fnn=Cancel

Using this display you can execute an IBM i command.

9.6 User Defined Developer Services Exit Programs

The user can specify up to 3 user defined developer services programs that can be selected as options from the "Developer Services" action box. These options will only be available from the Developer Services menu if the DEVELOPER parameter of the LANSA command is set to "*YES" and user defined developer programs have been specified in the extended information data area DC@A07.

Refer to 9.11 Developer Services Configuration later in this chapter for more information on how to define a user defined "Developer Services Exit program" in the extended information data area DC@A07.

9.7 Notifying a User of Developer Messages

A user will be notified of any developer messages that have been marked for his/her attention (sent to) if the ALLOW_MSGS parameter has been specified on the LANSA command with a value of "*YES" when one of these events occur:

- Invoking LANSA for the first time and "SYSTEM" type developer messages exist which require the user to be notified of their existence.
- Accessing an object (field, file, function or process) in LANSA and developer messages exist which require the user to be notified of their existence.

Note: The notify user flag of the category code for which the developer message applies to determines whether the user should be notified of the developer's message existence.

If developer messages exist which have been sent to the user and require him/her to be notified of their existence, then the user will be prompted with the Developer Messages "Action Box" from which either of these options may be selected:

Developer Messages :
Developer Messages exist which have :
been marked for your attention :
Use cursor to select option :
Work with Developer Messages :
Proceed to next screen :
Fnn=Help Fnn=Cancel :

Working from this screen you can select an option to:

• Work with developer messages will present the user with a list of all the developer messages sent to him/her that relate to the object currently being accessed and that have a category code with a "notify user" flag set to "Y" that requires the user to be notified of their existence. For more information, refer to 9.4 Work with Developer Messages (LANSA/Messages).

- Proceed to next screen ignores the developer messages which have been sent to the user and proceeds to the next screen.
- "Cancel" function key ignores the developer messages and proceeds to the next screen.

9.8 Export Developer Messages

The Export Developer Messages facility (LANSA EXPORTMSGS) allows you to export developer messages from one LANSA system to another.

9.8.1 Steps Required to Export Developer Messages

The basic requirements to ensure an EXPORTMSGS job completes normally is that the library name specified in PARM03 exists and the export date specified in PARM04 is a valid date in CCYYMMDD format, the two export files specified in PARM01 and PARM02 are (re)created each time an EXPORTMSGS job is executed.

To Export Developer Messages perform the following command:

```
LANSA REQUEST(EXPORTMSGS) PARM01(AAAAAAAAAA)
PARM02(BBBBBBBBB) PARM03(CCCCCCCCC)
PARM04(CCYYMMDD)
```

replacing:

- 'AAAAAAAAAA' in PARM01 with the name of the file to receive the export developer message header details. Use UPPERCASE characters only.
- 'BBBBBBBBBB' in PARM02 with the name of the file to receive the export developer message text details. Use UPPERCASE characters only.
- 'CCCCCCCCC' in PARM03 with the name of the library where the exported developer message files will be created. Use UPPERCASE characters only.
- 'CCYYMMDD' in PARM04 with the export start date, the export start date must be in CCYYMMDD format Century, Year, Month and Day). All developer messages that were first created/subsequently changed on or after the export start date will be exported.

For example, this command:

```
LANSA REQUEST(EXPORTMSGS) PARM01(MSGFIL1)
PARM02(MSGFIL2) PARM03(EXPLIB)
PARM04(19911231)
```

will result in all the developer messages that have a create date greater than or equal to "19911231" (31st of December 1991) being exported to the export developer message files "MSGFIL1" and "MSGFIL2" in library "EXPLIB".

9.8.2 Steps to Perform after Exporting Developer Messages

These examples provide some guidelines on what to do with the export developer messages files when the EXPORTMSGS job has completed normally:

Example 1

If the export developer message files are to be saved on a tape or diskette device, perform the following IBM i Control Language command:

SAVOBJ OBJ(AAAAAAAAA BBBBBBBBBB LIB(CCCCCCCCC) DEV(DDDDDDDDD) OBJTYPE(*FILE)

replacing:

'AAAAAAAAAA' with the name of the file which contains the exported developer message header details.

'BBBBBBBBBB' with the name of the file which contains the exported developer message text details.

'CCCCCCCCC' with the name of the library where the exported developer message files have been created.

'DDDDDDDDD' with the name of the device where the exported developer message files will be saved.

Example 2

If the export developer message files are to be sent to another system using SNADS, perform the following IBM i Control Language command:

SNDNETF FILE(BBBBBBBBBBBBBAAAAAAAAAA) TOUSRID((CCCCCCCC DDDDDDD))

replacing:

'AAAAAAAAAA' with the name of the file which contains the exported developer message header details.

'BBBBBBBBBB' with the name of the library where the exported developer message header file is resident.

'CCCCCCCC' and 'DDDDDDDD' with the two-part user ID of one or more users to whom the exported developer message header file is being sent. then:

SNDNETF FILE(BBBBBBBBBBB/AAAAAAAAA)

TOUSRID((CCCCCCC DDDDDDD))

replacing:

'AAAAAAAAAA' with the name of the file which contains the exported developer message text details.

'BBBBBBBBBB' with the name of the library where the exported developer message text file is resident.

'CCCCCCCC' and 'DDDDDDDD' with the two-part user ID of one or more users to whom the exported developer message text file is being sent.

The complete EXPORTMSGS procedure can be performed manually, but it is recommended that this procedure be executed within an IBM i Control Language program.

If LANSA/MESSAGES is being run in a multi-machine environment it is recommended that the EXPORTMSGS job should be executed on a daily basis as part of a start of day/day end procedure.

9.9 Import Developer Messages

The Import Developer Messages facility (LANSA IMPORTMSGS) allows you to import developer messages from another LANSA system.

9.9.1 Steps Required to Import Developer Messages

The basic requirements to ensure an IMPORTMSGS job completes normally is that the two import developer message files that contain the developer message header and text details are resident in the import library on the system where the IMPORTMSGS job will execute.

These examples provide some examples on how to ensure the import developer messages files are resident in the import library before the IMPORTMSGS is executed:

Example 1

If the import developer message files are to be restored using a tape or diskette device, perform the following IBM i Control Language command:

replacing:

'AAAAAAAAA' with the name of the file which contains developer message header details were saved.

'BBBBBBBBBBB' with the name of the file which contains developer message text details were saved.

'CCCCCCCCC' with the name of the library where the developer message files were saved.

'DDDDDDDDDD' with the name of the device where the developer message files will be restored from.

Example 2

If the developer message files have been sent from another system using SNADS, you will need to perform the following IBM i Control Language commands so that the developer message files can be received ready for import:

Before the developer message files can be imported a duplicate of two internal LANSA files must be made before the files can be received:

CRTDUPOBJ OBJ(DC@F83) FROMLIB(AAAAAAAAA) OBJTYPE(*FILE) TOLIB(BBBBBBBBB) NEWOBJ(CCCCCCCCC)

replacing:

'AAAAAAAAAA' with the name of the LANSA system data library where DC@F83 is resident. DC@F83 is an internal file within LANSA which stores developer message header LOG records.

'BBBBBBBBB' with the name of the library where the duplicate of DC@F83 is to be created. The library name specified here should be the name of the import library where the developer message files will reside.

'CCCCCCCCC' with the name of the file which will contain the imported developer message header detail records to be used in the IMPORTMSGS job.

CRTDUPOBJ OBJ(DC@F84) FROMLIB(AAAAAAAAA) OBJTYPE(*FILE) TOLIB(BBBBBBBBB) NEWOBJ(CCCCCCCCC)

Replace:

'AAAAAAAAAA' with the name of the LANSA system data library where DC@F84 is resident. DC@F84 is an internal file within LANSA which stores developer message text LOG records.

'BBBBBBBBB' with the name of the library where the duplicate of DC@F84 is to be created. The library name specified here should be the name of the import library where the developer message files will reside.

'CCCCCCCCC' with the name of the file which will contain the imported developer message text detail records to be used in the IMPORTMSGS job. When duplicates of DC@F83 and DC@F84 have been created, the developer

message files can be received:

RCVNETF FROMFILE(AAAAAAAAA) TOFILE(CCCCCCCC/BBBBBBBBBBB MBROPT(*REPLACE)

replace:

'AAAAAAAAA' with the name of the file which has been sent from an external system containing the developer message header detail records to be imported.

'BBBBBBBBB' with the name of the file which will receive the developer message header detail records to be imported. The file name specified here should be the name of the import file that will be used in the IMPORTMSGS job containing the developer message header detail records.

'CCCCCCCCC' with the name of the library where the developer message

header file to be used in the IMPORTMSGS job will be resident. then:

RCVNETF FROMFILE(AAAAAAAAA) TOFILE(CCCCCCCCC/BBBBBBBBBBB MBROPT(*REPLACE)

replace:

'AAAAAAAAAA' with the name of the file which has been sent from an external system containing the developer message text detail records to be imported.

'BBBBBBBBBB' with the name of the file which will receive the developer message text detail records to be imported. The file name specified here should be the name of the import file that will be used in the IMPORTMSGS job containing the developer message text detail records.

'CCCCCCCCC' with the name of the library where the developer message text file to be used in the IMPORTMSGS job will be resident.

To Import Developer Messages perform the following command:

LANSA REQUEST(IMPORTMSGS) PARM01(AAAAAAAAA) PARM02(BBBBBBBBBB) PARM03(CCCCCCCCC)

Replacing:

'AAAAAAAAAA' in PARM01 with the name of the file that contains the developer message header details to be imported. Use UPPERCASE characters only.

'BBBBBBBBB' in PARM02 with the name of the file that contains the developer message text details to be imported. Use UPPERCASE characters only.

'CCCCCCCCC' in PARM03 with the name of the library where the developer message files to be imported are resident. Use UPPERCASE characters only.

For example, the following command:

LANSA REQUEST(IMPORTMSGS) PARM01(MSGFIL1) PARM02(MSGFIL2) PARM03(IMPLIB)

will result in all the developer messages within the two import developer message files "MSGFIL1" and "MSGFIL2" in library "IMPLIB" being

imported/merged into the "real" developer message files in the current LANSA system where the IMPORTMSGS job has taken place.

If a new developer message has been imported/merged into the "real" developer message files and the send message flag for the category code that applies to the developer message is set to "Y", then the developer message will be sent to the user who the message is intended for by the user defined "send developer message" program.

The complete IMPORTMSGS procedure can be performed manually, but it is recommended that this procedure be executed within an IBM i Control Language program.

If LANSA/MESSAGES is being run in a multi-machine environment it is recommended that the IMPORTMSGS job should be executed on a daily basis as part of a start of day/day end procedure.

9.10 Purge Developer Messages

The Purge Developer Messages job is invoked as part of the LANSA REORG job. Refer to Submit the Job to Re-Organize the Database for more information on the LANSA REORG job.

When the Purge Developer Messages program is invoked these developer messages will be purged:

- All non-"SYSTEM" type Developer Messages that are unattached. That is, the object to which the developer message is attached no longer exists within the LANSA system across all partitions.
- All Temporary Developer Messages that are less than (older), or equal to, the purge date specified in the LANSA REORG job.

Note: All developer messages that were first sent/created before or on the specified purge date will be purged.

9.11 Developer Services Configuration

If you wish to specify your own user defined programs and user defined tables, there are a number of initial steps that should be performed before developer services is used for the first time.

The option to define your own user defined programs and tables has been introduced to provide you with a greater amount of flexibility to the developer services facility. Please use this option, it has been introduced for your benefit. The user defined programs and tables are:

Developer Services exit programs

- User profile validation program
- Send a developer message program
- Developer messages category codes table

The user defined programs and tables are stored within the extended information data area DC@A07 and contain default values when LANSA is first installed, but these defaults can be changed to your own specific requirements.

9.11.1 User Defined Developer Services Exit Programs

You can have up to 3 user defined developer services exit programs available for use on the Developer Services Menu, simply by defining their names and descriptions in the extended information data area DC@A07.

To enable the first user defined developer services exit program to be invoked these steps must be taken:

Step 1

Change the extended information data area DC@A07 by inserting the name and description of the user defined developer services program into positions 21 to 60, using the following command:

Replace:

'AAAAAAAAA' with the user defined developer services exit program name. Use UPPERCASE characters only.

Step 2

The user defined developer services exit program will have no parameters passed to it.

To enable the second and third user defined developer services exit program to be invoked, repeat the steps above ensuring the start position on the CHGDTAARA command described in step 1 is changed to:

Developer Services Exit program 02:

CHGDTAARA DTAARA(DC@A07 (61 40)) VALUE('AAAAAAAAABBBBBBBBBBBBBBBB...')

and Developer Services Exit program 03:

CHGDTAARA DTAARA(DC@A07 (101 40)) VALUE('AAAAAAAAABBBBBBBBBBBBBBBB...')

9.11.2 User Defined User Profile Validation Program

The user defined "user profile validation" program is used to validate user profile names specified in the "To User" field within the "Work with Developer Messages" facility.

The default program provided with the LANSA system to perform user profile validation is defined in the extended information data area DC@A07 and is called "DC@P9101". However, the user has the option to change the default program to their "user profile validation" program name.

To enable a user defined user profile validation program to be invoked these steps must be taken:

Step 1

Change the extended information data area DC@A07 by inserting the name of the user defined user profile validation program into positions 141 to 150, using the following command:

CHGDTAARA DTAARA(DC@A07 (141 10)) VALUE('AAAAAAAAA')

Replace 'AAAAAAAAAA' with the user defined user profile validation program name. Use UPPERCASE characters only.

Step 2

The user defined user profile validation program must be set up to receive 2 parameters which are:

Parameter Type Length Description Number

1	А	10	User/Group profile name - contains the name of the user/group profile to whom a developer message will be sent to.
2	А	1	Return code. Values "Y" = Valid profile "N" = Invalid profile

The default "user profile validation" program "DC@P9101" simply receives a user profile name and performs a check to ensure the user profile exists and returns a valid/invalid return code, however it is possible to achieve a similar result as "DC@P9101" by designing and implementing your own "user profile validation" program to perform a much more specific purpose such as a program to present a list of user profiles and return the selected user profile to the Developer Message program.

9.11.3 User Defined Send a Developer Message Program

The user defined "send a developer message" program is used to send developer messages from within the "Work with Developer Messages" facility and the "Import Developer Messages" job.

The default program provided with the LANSA system to send developer messages is defined in the extended information data area DC@A07 and is called "DC@P9102", however the user has the option to change the default program to the name of his/her own "send a developer message" program.

To enable a user defined send developer message program to be invoked these steps must be taken:

Step 1

Change the extended information data area DC@A07 by inserting the name of the user defined "send developer message" program into positions 151 to 160, using the following command:

CHGDTAARA DTAARA(DC@A07 (151 10)) VALUE('AAAAAAAAA')

Replace 'AAAAAAAAAA' with the user defined send a developer message program name. Use UPPERCASE characters only.

Step 2

The user defined send developer message program must be set up to receive 19 parameters which are:

Parameter	Type	Length	Description
Number			

1	А	25	Message Identifier - is the unique identifier of the developer message.
2	А	10	Created by User - is the user/group profile name of the user who created the developer message.
3	А	8	Created on date - is the date when the developer message was created.
			*Note: "created on date" will be in system date format and separated by the system date separator character.

4	А	6	Created at time - is the time when the developer message was created.
5	А	10	Created by Job Name - the name of the job where the developer message was created.
6	А	6	Created by Job Number - the number of job where the developer message was created.
7	А	20	Subject - is the brief description which relates to the developer message.
8	А	10	User to receive message - is the user/ group profile name of the user who the developer message should be "sent to".
9	А	3	Category code - is the category of this developer message.
10	А	15	Category code description - is the description of the category for the developer message.
11	A	1	Category code send flag - indicates whether the developer message should be sent on creation. Values "Y" = Send developer message "N" = Do not send developer message
12	A	1	Category code notify flag - indicates whether the user should be notified of the developer message when accessing the object assigned to the message. Values "Y" = Notify user "N" = Do not notify user
13	A	1	Perm/Temp flag - indicates whether the developer message is permanent or temporary. Values "P" = Permanent developer message "T" = Temporary developer message
14	А	20	Perm/Temp Narrative - is a brief description relating to the value of the Perm/Temp flag.

15	A	8	Object/Message type - is the type of message or object for which the developer message relates to.
16	А	10	Object name - is the name of the object for which the developer message relates to.
17	А	10	Extension - is the object extension name of the object for which the developer message relates to.
18	А	varies	Message Text - contains the complete message text for the developer message.
19	A	1	Return code - indicates whether the developer message was sent successfully.
			Values "Y" = Send message successful "N" = Send message failed

Tip

The default "send a developer message" program "DC@P9102" receives all the message details and message text for a developer message and sends a standard format message to the user ("To user") the message is intended for, however it is possible to achieve a similar result as "DC@P9102" by designing and implementing your own "send a developer message" program to perform a much more specific purpose such as:

- A program to send a developer message in your own format.
- A program to send network messages using the developer message information.
- A program to capture the developer message information and store its details within an external database file.

9.11.4 User Defined Developer Message Category Codes Table

The user defined developer message category codes, descriptions, send message and notify user flags will be used from within the "Work with Developer Messages" facility to categorize, describe and control the sending/notification of developer messages.

A default table of category codes, descriptions, send message and notify flags will be provided with the LANSA system, but the user has the option to change the default table to specify his/her own table of category codes, descriptions, send message and notify user flags.

The default category codes table is:

Category Code	Description	Send Message	Notify User
ANC	Announcement	Y	Ν
FIN	Finished	Ν	Ν
INF	Information	Ν	Y
NTE	Note	Ν	Ν
QUS	Question	Y	Ν
PRB	Problem	Y	Y
RPY	Reply	Y	Ν

The default table of user defined category codes, descriptions, send message and notify user flags was set up as follows:

• Change the extended information data area DC@A07 by inserting the user defined category codes into positions 161 - 220, using the following command:

CHGDTAARA DTAARA(DC@A07 (161 60)) VALUE('ANCFININFNTEQUSPRBRPY')

Up to 20 category codes can be specified, each code is 3 characters long.

• Change the extended information data area DC@A07 by inserting the user defined category code descriptions into positions 221 to 520 using the
following command:

CHGDTAARA DTAARA(DC@A07 (221 300)) VALUE('Announcement Finished Information Note Question Problem Reply')

Up to 20 descriptions can be specified, each description is 15 characters long. Ensure each category code description relates to the correct category code.

• Change the extended information data area DC@A07 by inserting the user defined send developer message flags into positions 521 to 540, using the following command:

CHGDTAARA DTAARA(DC@A07 (521 20)) VALUE('YNNNYYY')

Up to 20 send developer message flags can be specified, each flag is 1 character long and its value can only be either a "Y", "N" or " ". The value of this flag determines whether a message should be sent or not when a message is first created. Ensure each send developer message flag relates to the correct category code.

• Change the extended information data area DC@A07 by inserting the user defined notify user flags into positions 541 to 560, using this command:

CHGDTAARA DTAARA(DC@A07 (541 20)) VALUE('NNYNNYN')

Up to 20 notify user flags can be specified, each flag is 1 character long and its value can only be either a "Y", "N" or " ". The value of this flag determines whether the user should be notified of the message each time the object associated with the message is accessed. Ensure each notify user flag relates to the correct category code.

Use the above steps to specify your own table of category codes, descriptions, send developer and notify user flags.

9.11.5 Extended Information Data Area - DC@A07 Configuration

Positions 21 through to position 560 of the Extended Information Data Area DC@A07 has been exclusively reserved for developer services user defined programs and tables.

To review and amend the details stored in these data areas, use the System Settings facility available from the Administration menu and described in Review System Settings.

This area is broken down as:

From Byte	To Byte	Byte Length	Туре	Description
21	30	10	A	Name of the user defined Developer Services Exit program 01.
31	60	30	A	Description of the user defined Developer Services Exit program 01.
61	70	10	A	Name of the user defined Developer Services Exit program 02.
71	100	30	А	Description of the user defined Developer Services Exit program 02.
101	110	10	А	Name of the user defined Developer Services Exit program 03.
111	140	30	А	Description of the user defined Developer Services Exit program 03.
141	150	10	А	Name of the user defined "User Profile Validation" program.
151	160	10	А	Name of the user defined "Send Developer Message" program.
161	220	60(20x3)	А	Developer Message Category codes. Up to 20 codes can be specified, each code is 3 characters long.
221	520	300(20x15)	А	Developer Message Category Descriptions. Up to 20 descriptions can be specified, each

description is 15 characters long. Ensure the category description relates to the correct category code.

user should be notified of the message each time the object associated with the message is accessed. Ensure the category notify user flag relates to the correct category code.

521	540	20(20x1)	A	 Developer Message Category Send Message Flag. Up to 20 send message flags can be specified, each flag is 1 character long and its value can be either a "Y", "N" or " ". The value of this flag determines whether a message should be sent or not when a message is first created. Ensure the category send message flag relates to the correct category code.
541	560	20(20x1)	A	Developer Message Category Notify User Flag. Up to 20 notify user flags can be specified, each flag is 1 character long and its value can be either a "Y", "N" or " ". The value of this flag determines whether the

9.11.6 Next Developer Message Number - DC@A09 Configuration

The Next Developer Message Number data area DC@A09 has been created specifically for developer services.

WARNING - DO NOT attempt to change the contents of this data area as unpredictable results could occur when creating new Developer messages.

From Byte	To Byte	Byte Length	Туре	Dec Pos	Description
1	7	7	Ν	0	Next Developer message number for this LANSA system.

This data area will be automatically maintained by the LANSA system. The value for this data area will be incremented every time the "Create a Developer Message" facility has been invoked ensuring a unique message number is allocated to each new developer message identifier.

10. Configuration

Configuration items are LANSA system definition values that allow you to customize your LANSA environment.

10.1 Work with Configuration Items

10.2 LANSA Communications Extensions Configuration Items (COMMS_EXTENSIONS)

10.3 LANSA 3rd Party ILE Extensions

10.1 Work with Configuration Items

Some system definition values are accessed directly by changing the IBM i objects in which they are stored.

Some other system definition values are stored in a format which is not easily accessible or which may change in time. For these, LANSA provides program interfaces by which you can display or change the values without having to be concerned about the physical implementation of the configuration item.

To work with configuration items, enter the command:

LANSA REQUEST(CONFIGURE)

The Work with Configuration Items screen is displayed.

DC@P401601 Work with Configuration Items

Current position . : /

Select configuration item to be reviewed or changed:

SelConfiguration ItemDescription_COMMS_EXTENSIONSLANSA communications extensions_ILE_EXTLANSA ILE Extension

F1=Help F3=Exit F12=Cancel F14=Messages

Also see

10.2 LANSA Communications Extensions Configuration Items (COMMS_EXTENSIONS)

10.3 LANSA 3rd Party ILE Extensions

From the Work with Configuration Items screen you can select the

configuration item to be displayed or changed.

Configuration items are logically arranged in a folder/document like structure to group related configuration items together. Selecting a group item will display the items housed under it. Once the configuration item being searched for is found, selecting it will bring the review/change screen for that particular configuration item.

The "Current position" field displays the sequence of all configuration items (each item separated by a "/") you have selected from the top level up to the current configuration item. The top level is where you are positioned when you first invoke the Work with Configuration Items screen.

Displayed in the lower part are the configuration items found in the current position.

Select the configuration item that is to be reviewed or changed from the list displayed by entering any non-blank character beside it in the column headed "Sel" (select). If you select a configuration item that is a group item, a list of the configuration items grouped under it will be displayed. Otherwise, the details of the configuration item will be displayed.

10.2 LANSA Communications Extensions Configuration Items (COMMS_EXTENSIONS)

Group item COMMS_EXTENSIONS groups all configuration items related to LANSA Communications Extensions. Select this item from the 10.1 Work with Configuration Items screen.

When configuration item COMMS_EXTENSIONS is selected, a screen containing all items related to LANSA Communications Extensions opens.

DC@P401601 Work with Configuration Items Current position . : /COMMS_EXTENSIONS

Select configuration item to be reviewed or changed:

Sel Configuration Item Description

- _ COMMS_DEFINITIONS LANSA communications definitions
- _ COMMS_EIM_USER EIM Authorized user
- _ COMMS_FACILITY_TRACE Communications facility tracing records
- _ COMMS_LISTENER_RECS Communications listener records
- _ COMMS_ROUTING_RECS Communications routing records

F1=Help F3=Exit F12=Cancel F14=Messages

These are:

10.2.1 LANSA Communications Definitions (COMMS_DEFINITIONS)
10.2.2 EIM Authorized User (COMMS_EIM_USER)
10.2.3 LANSA Communications Extensions Facility Tracing Records (COMMS_FACILITY_TRACE)
10.2.4 LANSA Communications Extensions Listener Records (COMMS_LISTENER RECS)

10.2.5 LANSA Communications Routing Records(COMMS_ROUTING_RECS)

10.2.1 LANSA Communications Definitions (COMMS_DEFINITIONS)

Configuration item COMMS_DEFINITIONS stores the basic definitions used by LANSA Communications Extensions for the listener and session jobs. Select configuration item COMMS_DEFINITIONS to review LANSA communications definitions:

LCOADMP401 Display Communications Definitions

Communications routing table .../etc/dc_pgmlib/lroute.dat

Trace file name /tmp/dc_pgmlib/lroute.trc

Maximum size of trace file (bytes) 640000 User queue for session jobs . . . QUSRSYS/LCOTPJOBS Job description for session jobs . *LISTENER Prefix for session jobs' name . . . TP

Line turnaround wait & retry: Wait time (milliseconds) 300 Number of retries 300

Object LCOA01 in library DC@PGMLIB now locked by this job. F1=Help F12=Cancel F14=Messages F21=Change

This screen displays the basic definitions required to use LANSA Communications Extensions. LANSA provides default values which can be left unchanged. Using this display you can:

- Review the details of LANSA communications definitions.
- Use the Change function key to change details of LANSA communications definitions.

Communications Routing Table

This is the Integrated File System (IFS) path to the stream file where the listener and facility tracing definition records are stored.

- It must be an absolute IFS path. That is, you must enter the path starting from the root directory ("/").
- If the IFS directory entered doesn't exist, it is created. If the routing table itself doesn't exist, it is created and initialized with default contents. If you will be creating a new directory, it is strongly recommended that you sign on using the LANSA system's owner user profile.

Trace File Name

This is the Integrated File System (IFS) path to the stream file where trace data is recorded when tracing is enabled.

- It must be an absolute IFS path. That is, you must enter the path starting from the root directory ("/").
- If the IFS directory entered doesn't exist, it is created. If the trace file itself doesn't exist, it is created during runtime when tracing is enabled. If you will be creating a new directory, it is strongly recommended that you signon using the LANSA system's owner user profile.

If left blank, the default value of /tmp/subdir/lroute.trc is used, where subdir is based on the name of the communications configuration object library.

Maximum Size of Trace File

Used to limit the amount of tracing data written to the tracing file. Once this limit is reached, no more tracing data is written to the file. The default size is 0, which means the size limit is the maximum size for the file allowed by the operating system.

User Queue for Session Jobs

The qualified name (library/user queue) of the IBM i user queue used by the prestarted session jobs to make themselves available to the listener job. It must satisfy these requirements:

• The nominated library must allow for user domain objects to be created in it (review the IBM i system value QALWUSRDMN).

If left blank the default value of QUSRSYS/LCOTPJOBS is returned.

Job Description for Session Jobs

The qualified name (library/job description) of the IBM i job description used by the listener job to submit session jobs. The job description named here allows the user to specify the job queue (and therefore the subsystem) where session jobs will run. It must satisfy these requirements:

- The job description must exist before it is named here.
- The library list parameter in the job description must include the libraries where the LANSA Communications Extensions programs reside.
- The special value *LISTENER may be used. This special value specifies that sessions jobs will be submitted to the same job queue and with the same job description and library list used to submit the listener job.

If left blank the default value of *LISTENER is returned.

Prefix for Session Jobs' Names

The prefix used to start session jobs' names. Its purpose is to make it easier to identify session jobs in the system. The session job name is completed by appending a numeric sequential number (for example, TP00000001). The prefix must satisfy these requirements by being:

- 1 to 4 characters long.
- A valid IBM i name.

If left blank the default value of "TP" is returned.

Line Turnaround Wait & Retry

LANSA communication jobs are conversation oriented, in which the local and remote programs take turns to send and receive data. Before one side can send data, the other side must "turn the line around". If one side attempts to send data when it is in "receive state", it will get a state-check exception. This may also occur if the receiving side has already received all the data it expected and is ready to send, but it hasn't got yet the line turnaround signal, due to a slow communications link or due to one side,(either the remote or local program) being comparatively slower than the other side.

Use these "wait & retry" values to reduce the number of state check exceptions due to delays in the line turnaround signal.

- Wait time the time to wait (in milliseconds) before checking if the line has been turned around.
- Number of retries The number of times it will check for the line turnaround

signal.

If left blanks the default values of 300 milliseconds and 300 retries are returned.

10.2.2 EIM Authorized User (COMMS_EIM_USER)

To add or update EIM (Enterprise Indentity Mapping) User Details, select COMMS_EIM_USR from the 10.2 LANSA Communications Extensions Configuration Items (COMMS_EXTENSIONS) screen.

To use this facility, you must be logged on as QSECOFR, or be part of the QSECOFR group, the Partition Security Officer or part of the Partition Security Officer group.

Add EIM User

This facility enables you to specify the User Name and Password that LANSA Communications will use when querying the Enterprise Indentity Mapping system during Kerberos authentication.

Only one User Name and Password is recorded and this is stored in a validation list. This Validation list (LCOVLDL *VLDL) is created by the LANSA IBM i Install/Upgrade process and placed in the LANSA Communication Library.

When changing either field on this screen, please re-enter the required values for both fields.

DC@P813101 Add EIM - User Details

Distinguished name Password

F1=Help F3=Exit F12=Cancel F14=Messages

Add/Update the name and the password as described:

Distinguished name

Enter the distinguished name of an LDAP user authorized to query the EIM. A typical example is:

cn=Administrator

Password

Enter the password for the LDAP user authorized to query the EIM.

10.2.3 LANSA Communications Extensions Facility Tracing Records (COMMS_FACILITY_TRACE)

Configuration item COMMS_FACILITY_TRACE allows the creation, change and deletion of LANSA Communications Extensions Facility Tracing Records. These records are used to set tracing and logging options for each facility or module. When the listener job starts, it enables the tracing options specified by these records or those specified on the listener record.

Select configuration item COMMS_FACILITY_TRACE from the COMMS_EXTENSIONS group to review Communications Facility Tracing Records. The screen presented looks something like this:

г

LCOADMP405	Work with Comms Facil	ity Tracing Records
Type options, press 2=Review/Change	s Enter. 4=Delete	
Sel Facility	Trac	cing
CPIC MANA	AGER	Partial
IPC MODUL	LE	Partial
Object /etc/dc_pgn	nlib/lroute.dat now locked	by this job.
F1=Help F6=Add/	/Create F12=Cancel F14=	=Messages

This screen lists all the facilities (modules) for which tracing records have been created. To the right of the facility name is a summary value of the tracing flag settings.

Note: Facility tracing records are entirely optional. If they are defined, the normal settings should be with all flags unchecked (that is, no tracing and no

logging). Tracing should only be set on when asked by your product vendor to assist in troubleshooting.

- Full: Means all tracing and logging options have been selected for this facility or module.
- Partial: Means that some, but not all tracing and logging options have been selected for this facility or module.
- None: Means that the facility tracing record exists, but none of the tracing or logging options have been selected.

Working from this screen you can, by entering one of the options beside a list entry, perform these actions:

2=Review/Change: Display the tracing and logging details for the facility tracing record. From the screen displayed, press the Change function key to enable the fields to be amended.

4=Delete: Delete the facility tracing record on which this option is entered.

Pressing the Add/Create function key will present the Add Facility Tracing screen which allows the addition of a new facility tracing record.

LCOADMP414	Add Facility Tracing Record		
Facility (Module)	· · · · ·		
Tracing Options Errors External IPC calls IPC data Debug listener Calls to CPIC Internal IPC calls CPIC data System info Cryptographic info			
Logging Options Connect in Connect out			
F1=Help F12=Cancel	F14=Messages		

These input options are required on the Add Facility Tracing Record screen:

Facility (Module)

The facility or module for which the tracing and logging options are specified. It must be one of these values:

- CPIC MANAGER: The CPIC Manager service program.
- IPC MODULES: The Inter Process Communications (IPC) module. For example, the TCP/IP sockets module.
- LISTENER: The listener program.
- TP STUB: The Transaction Program (TP) entry program.
- ADMIN UTILITY: The LANSA Communications Extensions program to maintain configuration items.

Tracing Options

Check (enter any non-blank character) beside the events or actions which require tracing:

Errors	Traces error information. This consists of errors which may be considered "normal" during communications as well as any "fatal" errors encountered. "Fatal" error messages are also written to the IBM i job log.
System Info	Traces basic information at the system software level such as version negotiation.
Debug Listener	Traces the events in the listener job.
CPIC Data	Trace data sent and received at the point where LANSA Communications Extensions APIs are called.
IPC Data	Trace data sent and received at the point where the Inter Process Communications (IPC) interface (for example, TCP/IP sockets) is called.
External IPC Calls	Trace calls to the IBM i communications interface (for example, TCP/IP sockets APIs).
Internal IPC	Trace calls made to the Inter Process Communications (IPC)

Calls	interface (for example, the Sockets IPC Module).
Calls to CPIC	Trace API (CPIC) calls to the LANSA Communications Extensions CPIC Manager.
Cryptographic Info	Trace cryptographic information

Logging Options

Enables the recording of normal events which need tracking. On the IBM i, messages are sent to the job's external message queue and will be listed in the job log. Check (enter any non-blank character) beside the event(s) which require logging:

Connect In Record details of incoming connection requests.

Connect Out Reserved for future use.

10.2.4 LANSA Communications Extensions Listener Records (COMMS_LISTENER_RECS)

Configuration item COMMS_LISTENER_RECS allows the creation, change and deletion of LANSA Communications Extensions Listener Records. These records define the parameters used by the listener job when employing the listener record's communications method.

Select configuration item COMMS_LISTENER_RECS from the COMMS_EXTENSIONS group to review these records.

The screen presented looks something like this:

This screen lists all the listener records stored in the LANSA Communications Extensions routing table.

Working from this screen you can, by entering one of the options beside a list entry, perform these actions:

2=Review/Change: Display the details for the listener record. From the screen

displayed, press the Change function key to enable the fields to be amended. 4=Delete: Delete the listener record on which this option is entered. Pressing the Add/Create function key will present the Add Communications Listener Record screen which allows the addition of a new listener record.



These input options are required on the Add Listener Record screen:

Communications Method

Is the communications method used by the Listener. The communications methods supported on the IBM i by LANSA Communications Extensions are:

• SOCKET: Use TCP/IP sockets.

Number of Session Jobs to Prestart

Enter the number of session jobs to be prestarted by the listener job. Session jobs are prestarted to reduce the connection time from requesting clients by avoiding the time required by the operating system to start a job.

It must be a positive number in the range 1 to 9,999.

Note: The subsystem where the session jobs run (and the job queue where they are submitted to) must have a "maximum number of jobs active at the same time" value of at least the number entered in the "Number of Session Jobs to Prestart" field, otherwise some session jobs will stay on the job queue.

Connection Identifier

The meaning and purpose of this field is specific to each communications method:

• For SOCKET (TCP/IP) this is the port number at which the listener job will listen for connection requests. If left blank, the default value of 4545 is used. The workstations must be configured to connect to the port number nominated here. If left blank they will use the default value of 4545.

Cryptographic Algorithm

The cryptographic algorithm used to encrypt transmitted data:

- *NONE No encryption. This is compatible with prior releases of LANSA Communication extension.
- SSL Secure Sockets Layer. SSL and its successor TLS are today the most widely used protocols providing end-to-end encryption of network communications. LANSA's SSL implementation uses anonymous Diffie-Hellman keys generated anew for each session, used for symmetric encryption with SHA-1 based ciphers. It does not require certificates for server authentication.
- DES Data Encryption Standard algorithm. This standard was adopted by ANSI in 1981. DES is a block cipher, it encrypts data in 64 bit blocks. The key length is 64 bits.
- TWF Twofish cryptographic algorithm. This was one of the five finalists for the new encryption standard known as Advanced Encryption Standard(AES). This standard was introduced by NIST(National Institute of Standards and Technology). Twofish (copyright Counterpane Internet Security Inc.) is a 128 bit block cipher. The key length can be 128, 192 or

256 bits.

The default value is *NONE. That is, the data is not encrypted.

The use of encryption and/or compression algorithms cause more CPU cycles to be consumed during communication sessions. However this overhead is quite small when compared to other communication overheads found in most normal operational situations.

Note: If a cryptographic algorithm is selected, the listener job will only accept connection requests from clients that support the selected cryptographic algorithm.

If you have clients that are on an earlier version of LANSA Communications Extensions (versions 1.x) which doesn't support encryption, then you can only accept connection requests from them if you select *NONE.

Press the prompt key on this field to select from the list of valid cryptographic algorithms.

Cryptographic Algorithm Selection

From this panel, you can select a cryptographic algorithm by typing its name in the input field or by positioning the cursor on the required cryptographic algorithm and pressing the Enter/record advance key.

LCOADMP410	Change C	Communications Li	istener Record	
Communications me	thod	SOCKET		
Number of	•••••	•••••		
Connection : LCOAL	OMP415	Select Cryptogr	aphic Algorithm	:
Cryptograp :		:		
Data compr :	Cryptogra	phic algorithm	*NONE :	
Login time :		:		
TCP_NODELA : Al	gorithm	Description	:	
Enable Ipv : *NONE	No e	ncryption	:	
TCP KeepAl : SSL	Secur	e Sockets Layer	:	
: DES Da	ata Encry	otion Standard (DE	ES) :	
: TWF T	wofish cry	ptographic algorit	hm :	
Tracing Op :	0	:		
Errors . :		:		

```
Debug li : Bottom : lls

CPIC dat : F1=Help F12=Cancel F14=Msgs : nfo

:......

Logging Options

Connect in . . . . Connect out . . . .

F1=Help F4=Prompt F12=Cancel F14=Msgs F16=SelBasic F17=Select
```

Data Compression

F18=Clear All

Whether the transmitted data should be compressed or not:

- YES Allow LANSA Communications Extensions to compress the data before it is transmitted. This is the default value.
- NO Don't compress the data. You would normally choose this option only when enabling tracing for troubleshooting.

Login Timeout

How long to wait (in seconds) for the client's login packet before timing out (Enter a value from 1 to 32 seconds).

Enable IPv6

Whether to enable support for Internet Protocol version 6.

- YES The listener accepts client connections using IPv6 in addition to IPv4.
- NO The listener only accepts client connections using IPv4. This is the default value.

TCP_NODELAY TCP/IP Socket Option

Indicates whether the TCP_NODELAY TCP/IP socket option should be enabled or not. This option is displayed only if the communications method is SOCKET (TCP/IP). Valid values are:

- YES Disable the Nagle algorithm to allow small data packets to be delivered to the remote system without delay. This is the default value.
- NO Don't disable the Nagle algorithm.
- The Nagle algorithm coalesces a small data buffer from multiple send calls and delays sending it until an acknowledgement for the previous data packet

sent is received from the remote system. It is enabled by default in TCP/IP sockets.

• As the LANSA Communications Extensions implements its own data buffering, the Nagle algorithm is not needed.

TCP_KEEPALIVE TCP/IP Socket Option

Select this option to enable the TCP/IP socket option *KeepAlive*.

An application can request a TCP/IP provider to enable the use of keep alive packets on TCP connections by turning on the TCP_KEEPALIVE socket option. If a connection is dropped, as the result of the KEEPALIVE option, the error code 10052 is returned to any calls in progress on the socket. Any subsequent calls will fail with error code 10057. Refer to Communications Error Codes in the *LANSA Communications Setup Guide* for further details of these error codes.

Tracing Options

Check which events or actions require tracing when the listener job is running. Refer to Tracing Options in the *LANSA Communications Extensions Facility*

Tracing Records for the details of each option.

Logging Options

Enables the recording of normal events which need tracking when the listener job is running. On the IBM i messages are sent to the job's external message queue and will be listed in the job log.

Refer to 10.2.3 LANSA Communications Extensions Facility Tracing Records (COMMS_FACILITY_TRACE) for the details of each option.

10.2.5 LANSA Communications Routing Records(COMMS_ROUTING_RECS)

The routing information you enter on this screen is used to set up the link between the client and the server. It is stored in the communications routing table /<LANSA_adminlib>/connect/lroute.dat where <LANSA_adminlib> is your LANSA administration library.

LCOADMP407 Change Communications Routing Record Partner LU Name (HOST) *HOST Fully Qualified Name Local Communications method Socket Communications module 0 Connection identifier 4545 Packet size 00000 TCP NODELAY TCP/IP socket option . . YES Enable IPv6 NO Tracing Options External IPC calls . IPC data Errors Debug listener . . . Calls to CPIC . . . Internal IPC calls . System info Cryptographic info . CPIC data Trace Host Logging Options Connect in Connect out . . . Fn=Help Fnn=Cancel Fnn=Msgs Fnn=SelBasic Fnn=Select All Fnn=Clea

Enter the following information:

Partner LU name (Host)

This is the name for the Data/Application Server. (For example, the name might appear as S102ABCD.) This name is simply used to represent the fully qualified name.

Fully qualified name

For an IP connection, the fully qualified host name can be the IP address (for example 123.123.123.123) or it can be the assigned TCP Host Name (for example S102ABCD.APPN.SNA.IBM.COM) for the Application/Data Server.

Communications method

Always SOCKET.

Communications Module

Leave blank.

Connection Identifier

For SOCKET (TCP/IP), this is the port number at which the listener job will listen for connection requests. The workstations must be configured to connect to the port number nominated here. The default is 4545.

Packet size

Leave as default. The size is determined internally from the size of the send buffer. The default is 1024 plus the size of a header. This parameter should not be changed, unless the change is made by a networking specialist configuring for special circumstances.

TCP_NODELAY TCP/IP Socket Option

Indicates whether the TCP_NODELAY TCP/IP socket option should be enabled or not. This option is displayed only if the communications method is SOCKET (TCP/IP). Valid values are:

- YES Disable the Nagle algorithm to allow small data packets to be delivered to the remote system without delay. This is the default value.
- NO Don't disable the Nagle algorithm.
- The Nagle algorithm coalesces a small data buffer from multiple send calls and delays sending it until an acknowledgement for the previous data packet sent is received from the remote system. It is enabled by default in TCP/IP sockets.
- As the LANSA Communications Extensions implements its own data buffering, the Nagle algorithm is not needed.

Enable IPv6

Whether to enable support for Internet Protocol version 6.

- YES The host accepts client connections using IPv6 in addition to IPv4.
- NO The host only accepts client connections using IPv4. This is the default value.

Tracing Options

Select (enter any non-blank character) for each option to indicate the events or actions which require tracing or Press the relevant function key to either *Select the Basic* options, *Select All* options or *Clear All* options:

Traces error information. This consists of errors which may be considered "normal" during communications as well as any "fatal" errors encountered. "Fatal" error messages are also written to the IBM i job log.
Traces basic information at the system software level such as version negotiation.
Traces the events in the listener job.
Trace data sent and received at the point where the Inter Process Communications (IPC) interface (for example, TCP/IP sockets) is called.
Trace data sent and received at the point where LANSA Communications Extensions APIs are called.
Trace calls made to the Inter Process Communications (IPC) interface (for example, the Sockets IPC Module).
Trace calls to the IBM i communications interface (for example, TCP/IP sockets APIs).
Trace cryptographic information
Trace API (CPIC) calls to the LANSA Communications Extensions CPIC Manager.
Trace flags will be sent to the host during the connection of this route record. The host will use the trace flag for this communication session.

Logging Options

Enables the recording of normal events which need tracking when the listener job is running. On the IBM i messages are sent to the job's external message queue and will be listed in the job log.

Connect In Record details of incoming connection requests.

Connect Out Reserved for future use.

Press *Enter* when you have completed these entries.

10.3 LANSA 3rd Party ILE Extensions

LANSA ILE Extensions (Configuration Item: ILE_EXT) give you the ability to use 3rd party ILE service programs.

When using this feature, please note:

- It supports bound calls to within ILE service programs only via the RPGIV CALLB operation code.
- The creation of the program that uses 3rd party ILE service programs requires the service programs to be specified in binding directories.
- Both the binding directories and service programs must be in the library list at compile time.
- The service programs must be in the library list when the programs that use it are called.
- There is a maximum of 50 binding directories for each owner (Other Vendor and User Defined).
- The ILE module name can be no more than ten characters.
- When calling an ILE module, the library name should not be included.

10.3.1 Distribute the ILE Integration

- 1. Set-up the ILE integration on the source LANSA system.
- 2. Copy the relevant records from the ILE modules/service programs file in the source LANSA system data library.
- 3. Duplicate the relevant data area, either DC@ABNDUD or DC@ABNDOV, from the source LANSA system data library.
- 4. Copy the relevant records to the ILE modules/service programs file in the target LANSA system data library.
- 5. Update the relevant data area if it already has existing information, either DC@ABNDUD or DC@ABNDOV, from the target LANSA data library.
- 6. Duplicate the relevant data area, either DC@ABNDUD or DC@ABNDOV, to the source LANSA system data library.

10.3.2 Set Up the Integration

- 1. Use LANSA Configure to name the ILE Modules that are to be used.
- 2. Create one or more binding directory if no binding directory yet exists.
- 3. Use LANSA Configure to name the binding directories that are to be used.
- 4. Insert the RDML CALL or CALLCHECK commands, complex logic validation rules, etc ...as required, to call the ILE Service program module.
- 5. Include "*RPGIV" and "*ILE" in DC@OSVEROP data area if not already there.
- 6. Compile the functions and I/O modules.

10.3.3 Work with ILE Extensions

Group item LANSA ILE Extension (ILE_EXT) groups all the configuration items related to LANSA ILE Extensions.

When you select configuration item ILE_EXT from the main menu of the Work with Configuration Items screen, a list of all configuration items related to ILE extensions is displayed.

DC@P401601 Work wit	th Configuration Items				
Current position . : /ILE_EXT					
Select configuration item	to be reviewed or changed:				
Sel Configuration Item ILE_EXT_OV ILE_EXT_UD	Description Other Vendor ILE Extensions User Defined ILE Extensions				
F1=Help F3=Exit F12=C	Cancel F14=Msgs				

When you select the particular ILE Extensions (ILE_EXT) that you want to work with, the next Work with Configuration Items screen is displayed. Other vendor ILE Extensions (ILE_EXT_OV) has been selected in this example. A similar screen is displayed when ILE_EXT_UD is selected.

DC@P401601 Work with Configuration Items

Current position . : /ILE_EXT/ILE_EXT_OV

Select configuration item to be reviewed or changed:

Sel Configuration Item Description

 ILE_EXT_OV_BND
 ILE_EXT_OV_MOD

Other Vendor ILE Binding Directories Other Vendor ILE Modules

F1=Help F3=Exit F12=Cancel F14=Msgs

10.3.4 Binding Directories

When you select ILE Binding Directories from the Work with Configuration Items screen, you are presented with the Display Binding Directories screen. In this example, Other Vendor Binding Directories (ILE_EXT_OV_BND) was selected. A similar screen is displayed if you select User Defined ILE Binding Directories (ILE_EXT_UD_BND).

DC@P402101 Display Binding Directories - Other Vendor

MD@XXXX

Number of Entries 1

F1=Help F3=Exit F12=Cancel F14=Msgs F21=Change

The Display Binding Directories screen displays all the available Binding Directory for each owner.

Using this display you can:

- Review all binding directories.
- Use the Change function key to update the binding directory entries.

10.3.5 Change Binding Directories

When you press the change key on the Display Binding Directories screen, you are presented with the Change Binding Directories screen. This screen allows you to add or delete a binding directories entries.

In this example, Other Vendor Binding Directories (ILE_EXT_OV_BND) was selected. A similar screen is displayed if you select User Defined ILE Binding Directories (ILE_EXT_UD_BND).

DC@40212A Change Binding Directories - Other Vendors

MD@XXX______

Number of Entries.... 1

F1=Help F3=Exit F12=Cancel F14=Msgs

Enter, Change or Delete the new Binding Directory names as required.
10.3.6 Work with ILE Modules

When you select ILE Modules from the Work with Configuration Items screen, the Work with ILE Modules screen is displayed.

In this example Other Vendor ILE Modules (ILE_EXT_OV_MOD) was selected. A similar screen is displayed if you select User Defined ILE Modules (ILE_EXT_UD_MOD).

DC@P40201 Work with ILE Modules - Other Vendor

Type options, press Enter. 4=Delete

Sel	ILE Module	Service Program	
	MD@P001	SP@SXXX	
	MD@P002	SP@SXXX	
	MD@P003	SP@SXXX	
	MD@P004	SP@SXXX	
	MD@P005	SP@SXXX	
	MD@P006	SP@SXXX	
	MD@P007	SP@SXXX	
	MD@P008	SP@SXXX	

F1=Help F3=Exit F6=Create F12=Cancel F14=Msgs

This screen lists all the ILE Modules/Service Programs for the selected owner type (i.e. User Defined or Other vendor).

+

Using this display it is possible to:

- Delete an ILE Module/Service Program.
- Add an ILE Module/Service Program.

10.3.7 Add ILE Modules

When you press the Create function key on the Work With ILE Modules screen, the Add ILE Modules screen is displayed.

Add ILE Module

Press Enter to confirm, F12 to cancel

ILE Module _____

Service Program

F1=Help F12=Cancel F14=Msgs

Enter the ILE Module and associated Service Program and press Enter.

10.3.8 Delete ILE Modules

When option 4 is entered against an entry on the Work with ILE Modules, the Delete ILE Module screen is displayed.

Delete ILE Module

Press Enter to confirm, F12 to cancel

ILE Module MDXX@P001 Service Program SP@SXXX

F1=Help F12=Cancel F14=Msgs

Press the Enter key to permanently delete the displayed ILE Module from the file.

Press F12 to cancel the deletion process.

Appendix A. Virtual Fields

- What Is a Virtual Field?
- Virtual Field Applications
- Defining Virtual Fields & Code to LANSA
- Writing Virtual Field Derivation Code
- The VC_USING Command
- The VC_COPY Command
- Examples of Virtual Fields & Derivation Code

What Is a Virtual Field?

LANSA file definitions can contain three types of fields:

- Traditional or "real" fields.
- "Virtual" fields.
- "Predetermined join" fields.

For details about these three types of fields and how to enter them, refer to Virtual Fields in this guide.

Virtual Field Applications

The capabilities of virtual fields are only limited by your imagination. Here are just some examples. How you might use these examples is described in more detail in Examples of Virtual Field Applications.

- Sub-stringing and concatenation
- Access to files with no external description
- Simplification and standardization of common data manipulations
- Type and length conversions
- Aggregation and accumulation
- Date conversion
- Date to age conversion
- Dynamic data conversion

Defining Virtual Fields & Code to LANSA

The actual procedures for defining virtual fields in a file definition and for editing the associated RPG derivation code are described in detail in Steps Involved in Defining, Reviewing or Changing Virtual Fields.

Writing Virtual Field Derivation Code

Before attempting to write any virtual field derivation code several things should be understood:

• The code written is not a complete program. It will be automatically included at pre-defined points into the RPG file I/O module generated by LANSA when the file definition is made operational.

• The code written must adhere to certain restrictions if it is to work correctly.

When virtual field derivation code is to be input it must be input into one (or more) of the following "sections" of the file I/O module:

I/O Module Code Section	Allowable RPG Type
File specifications	"F" specs
Array specifications	"E" specs
External record format or field renames	"I" specs
Data structure specifications	"I" specs
Calculations after input from file	"C" specs
Calculations before output to file	"C" specs
Internal subroutines	"C" specs
Output specifications	"O" specs
Compile time array data	N/Applicable

In addition, certain restrictions and considerations must be followed to ensure that the virtual field derivation code will compile and work correctly:

• Work field naming: where "work" fields (as opposed to real or virtual fields in the file definition) are used in the code they should be named in such a way as to not interfere with field names generated by LANSA in the I/O module or RDML function.

Generally LANSA field names contain an "@" or a "\$" symbol to make them different. Do not use work field names that contain an "@" or a "\$"

symbol.

It is **strongly recommended** that a naming standard/convention for work fields used in virtual field derivation code be established so that their names **never** conflict with LANSA fields or other user defined fields.

• **File field names and locations**: all fields defined in the file definition (real and virtual) are renamed by LANSA within the I/O module. As such they cannot be referenced by their actual names unless a VC_USING command has been included into the code section.

In addition, all fields in the file definition are placed into a data structure by LANSA. As such it is not possible for your code to group file fields into a data structure. This is not a problem as alternative code can always be used to achieve the same effect.

Finally the "rename" name assigned to a file field and its location within the data structure is unpredictable and may change from compile to compile. As such, code written should never rely on a field having the same rename value or location in the data structure.

• LANSA fields: I/O modules contain many fields (other than file fields) that have been generated by LANSA. These fields should never be referred to or modified in user written code.

This is for two reasons. The first is that modification of the variables may cause LANSA to fail in some way. The second is that they may change type or length, or disappear altogether in subsequent LANSA releases.

- **Re-entrant code and robust logic**: LANSA I/O modules are re-entrant programs (i.e. they do not set on LR) and do not use the RPG cycle. As such all code written should be capable of being used in a re-entrant mode and be as robust as possible.
- **Indicator use**: Avoid the use of indicators whenever possible other than in "short scope" situations (i.e. within the same coding section). Never use an indicator to pass information between coding sections as LANSA may change its value between them. No restrictions about which indicators can be used exist.
- **Testing**: Like all logic, virtual field code included in an I/O module should be tested. Often substantial testing can be performed by using the file

maintenance utility. Refer to the File Maintenance Facility for more details.

- **Performance**: Remember that nothing is for free. If the virtual field derivation code uses up a lot of computer resource your applications will suffer accordingly.
- **Responsibility**: Remember that when virtual field derivation code is written its effect upon the functioning of an I/O module, an RDML function, and the LANSA product as a whole **is entirely your responsibility**.

Although your product vendor will assist you with virtual field derivation code, they are not responsible for its function nor liable in any way to correct any problems it causes.

• Using virtual field derivation code with *DBOPTIMISE: A facility called *DBOPTIMIZE (or *DBOPTIMISE) exists that allows a compiled RDML function to **directly** include all required I/O code.

This I/O code is used to perform database access directly, rather than by placing calls to I/O modules, to improve performance in some situations.

The *DBOPTIMISE facility is described in detail in an appendix at the end of this guide. Please read this appendix before attempting to use this facility.

This direct inclusion of I/O code into the RDML function can cause problems when coding virtual field derivation code.

Consider the following situation: two files, A and B, both have common date conversion requirements, so the associated virtual field derivation code is copied from one to the other.

The copied code includes a data structure ("I" Specs) and a subroutine ("C" specs).

When I/O modules for these files are created, this all works okay, because the data structure and subroutine are isolated in two different I/O modules.

However, if an RDML function, using *DBOPTIMISE, and accessing files A and B is compiled, it will fail. This is because the data structure and the subroutine would be included into the RDML function twice (once for each file).

Fortunately, a command called VC_COPY can be used to indicate that a code block should only be copied into the RDML function if it has not already been copied. Refer to The VC_COPY Command for more information about this command.

The VC_USING Command

In Writing Virtual Field Derivation Code it was mentioned that all file fields (both real and virtual) are renamed by LANSA within an I/O module.

This is done for 2 reasons:

- 1.A field name might be the same in another file (i.e. validation file, batch control file, etc). If the field was not renamed the possibility of "over-writing" the field would arise.
- 2. The field name in the file might be more than 6 characters long. Although LANSA supports 10 character field names RPG does not. They must be renamed.

Thus you are unable to reference real or virtual fields in your code by their actual names. However, this problem is overcome by the **VC_USING (virtual code using fields)** command which can be inserted into the RPG code sections as comments.

The VC_USING command tells LANSA what file fields (real or virtual) are used in the following section of code, and optionally what "name" is being used to represent the field in the following code. Thus LANSA can scan the source lines and replace any occurrences of the field name with its "rename" name when generating the I/O module.

The syntax diagram for the VC_USING command looks like this:

```
Required
VC_USING ----- FIELDS ------ field name ---- *SAME ------|
| name used |
| |
------ 100 maximum ------
```

FIELDS

Specifies a list of 1 to 100 fields that are defined in the file definition and optionally the name used in the user RPG code to represent the field.

Some points to note about using VC_USING commands are:

- If the maximum of 100 fields is exceeded the virtual fields are not calculated (derived). There is no runtime error.
- They can be inserted into any I/O module code section except "Compile time array data".

- The command and its parameters should only ever be coded between columns 8 -> 80 of the source program line. LANSA ignores columns 1 -> 7 of the line when processing a VC_USING command and its parameters.
- Where a command and its parameters will not fit on one line use a "+" sign as the last character on the line and continue the command in column 8 (or beyond) on the next line. This method is identical to that used when continuing IBM i operating system commands over multiple lines. Note that the "-" sign continuation character supported by the IBM i is **NOT** supported by LANSA in this context.
- The VC_USING command causes the RPG code that you write to be translated for compilation purposes.

Following further on in this section are detailed examples of how this is done and the effect that it has on your RPG virtual derivation code (as it is actually compiled).

One of the most **common problems** that occurs when using virtual code is that VC_USING command uses **simple scanning** to find and replace fields in the RPG code.

This can cause problems when different field names are generically identical, or when they are imbedded within one another.

For example:

C* C* VC_USING FIELDS(COMP COMPNO) C* C Z-ADDCOMP FIELD1 C Z-ADDCOMPNO FIELD2

which would be translated at compile time to:

* FIELD REPLACEMENT VALUES USED FOR THIS SECTION OF USER
*
* Field name as Field name as in Internal field name
* in file definition user RPG code used as replacment

111	me deminion	user KPG CO	ue use	u as replac
*				
*	COMP	COMP	@F(0001
*	COMPNO	COMPN	NO	@F0002
*				
С	Z-A	DD@F0001	FIELD1	
С	Z-A	DD@F0001	FIELD2	

which is incorrect because the names COMP and COMPNO have generically identical portions (i.e. "COMP"). The scanning logic always looks for "COMP" first, so it has replaced it in **both** lines of code.

The simplest solution is to make the scanning logic look for "COMPNO" first:

which would be translated correctly at compile time to:

** FIELD REPLACEMENT VALUES USED FOR THIS SECTION OF USEI * * Field name as Field name as in Internal field name *in file definition user RPG code used as replacment _____ COMPNO COMPNO @F0002 * COMP COMP @F0001 * С Z-ADD@F0001 FIELD1 С Z-ADD@F0002 FIELD2

But the best solution is to remove the generic similarities altogether like this: C*

```
C* VC_USING FIELDS((COMP XXXX) (COMPNO YYYYY))
C*
C Z-ADDXXXX FIELD1
C Z-ADDYYYYYY FIELD2
```

This variation of the command tells LANSA that field COMP is being used, but it has been referenced (and should be scanned for) as XXXX, and that field COMPNO is being used, but it is referenced (and should be scanned for) as YYYYYY.

It would produce a correct result like this:

* FIELD REPLACEMENT VALUES USED FOR THIS SECTION OF USEI

* Field name as Field name as in Internal field name

* n	* n file definition user RPG code used as replacement					
*						
*	COMP	XXXX	@F0001			
*	COMPNO	YYYYYY	@F0002			
*						
С		Z-ADD@F0001	FIELD1			
С		Z-ADD@F0002	FIELD2			

Another common variation of this problem is the imbedding of field names within one another. This example shows some other situations that would cause this problem and require corrective action:

C*		
C* VC_USIN	G FIELDS(CON	MPNO FXCO CO)
C*		
С	Z-ADDCOMPI	NO FIELD1
С	Z-ADDCO	WORKCO
С	Z-ADDFXCO	WFXCO1

which would be incorrectly translated like this:

*	Field name a	s Field name as	in Internal field name
*iı	n file definiti	on user RPG code	e used as replacement
*			
*	COMPNO	COMPNO) @F0001
*	FXCO	FXCO	@F0002
*	CO	CO (@F0003
*			
C	2	Z-ADD@F0001	FIELD1
C		Z-ADD@F0003	WORK@F0003
C		Z-ADD@F0002	W@F0002

VC_USING Examples

Some correct examples of VC_USING commands follow.

Example 1: Accumulate real fields TOT01 -> TOT05 into virtual field TOTAL:

C*

C* VC_USING FIELDS(TOT01 TOT02 TOT03 TOT04 TOT05 TOTAL) C*

С	Z-ADDTOT01	TOTAL
С	ADD TOT02	TOTAL
С	ADD TOT03	TOTAL
С	ADD TOT04	TOTAL
С	ADD TOT05	TOTAL

Some points to note about this example are:

- The VC_USING command details all fields in the code section that come from the file definition. The command is included into the source code as a comment so as not to upset the RPG syntax checker when editing and begins in column 9.
- None of the fields are declared (i.e. type and length specified). LANSA automatically declares all real and virtual file fields in a data structure.

If this example virtual field code section was actually compiled into the file I/O module it would look something like this:

* FIELD REPLACEMENT VALUES USED FOR THIS SECTION OF USE *

* Field name as Field name as in Internal field name*in file definition user RPG code used as replacement

*			
*	TOT01	TOT01	@F0001
*	TOT02	TOT02	@F0002
*	TOT03	TOT03	@F0003
*	TOT04	TOT04	@F0004
*	TOT05	TOT05	@F0005
*	TOTAL	TOTAL	@F0006
*			
С		Z-ADD@F0001	@F0006
С		ADD @F0002	@F0006

С	ADD	@F0003	@F0006
С	ADD	@F0004	@F0006
С	ADD	@F0005	@F0006

Note how the VC_USING command has been replaced by a series of comments that detail exactly what effect it has on the following code. Also note how all fields listed in the VC_USING command have been replaced by their "internal" names in the actual code section.

Example 2: Accumulate real fields TOT01 -> TOT05 into virtual field FULLTOTAL:

C*		
C* VC_	USING FIELDS(TOT	01 TOT02 TOT03
C*	TOT04 TOT05 (I	FULLTOTAL FFFFF))
C*		
С	Z-ADDTOT01	FFFFFF
С	ADD TOT02	FFFFFF
С	ADD TOT03	FFFFFF
С	ADD TOT04	FFFFFF
С	ADD TOT05	FFFFFF

Some points to note about this example are:

- The VC_USING command has been continued over 2 source lines by use of the "+" continuation character. In addition, it indicates that file field "FULLTOTAL" is actually referenced by name "FFFFFF" in the following code. This was done because field name FULLTOTAL is more than 6 characters long.
- The name FFFFFF has been used in place of name FULLTOTAL in all RPG code lines.

If this example virtual field code section was actually compiled into the file I/O module it would look something like this

FIELD REPLACEMENT VALUES USED FOR THIS SECTION OF USER *

* *i *	Field name as n file definition	Field name as in user RPG code	Internal field name used as replacement
*	тото1	тот01	@F0001
*	TOT02	TOT02	@F0002
*	TOT03	TOT03	@F0003
*	TOT04	TOT04	@F0004

*	TOT05	TOT05	@F	0005
*	FULLTOTAL	FFFF	FF	@F0006
*				
С	Z-AD	D@F0001	@F0006	
С	ADD	@F0002	@F0006	
С	ADD	@F0003	@F0006	
С	ADD	@F0004	@F0006	
С	ADD	@F0005	@F0006	

Again the VC_USING command has been replaced by a series of comments that detail exactly what effect it has had on the following code. Also note how all fields listed in the VC_USING command have been replaced by their "internal" names, particularly the name "FFFFFF".

The VC_COPY Command

In Writing Virtual Field Derivation Code, a duplication problem was described that can occur when virtual derivation code from multiple files is copied into an RDML function that uses the *DBOPTIMISE facilities.

The VC_COPY command can be used in virtual field derivation code to indicate that a block of code only be copied if it has not already been copied. The syntax diagram for the VC_COPY command looks like this:

Required

VC_COPY ------ SRCFILE ------ file name . library name ----->

>----- SRCMBR ------ source member name ------|

SRCFILE

Specifies the name and library of the IBM i source physical file containing the source member specified by the SRCMBR parameter.

*LIBL is acceptable and preferable as the library name, but of course, the correct library must be in the library list of any I/O module or RDML compile job attempting to access the member.

SRCMBR

Specifies the name of the source member in the source file specified by the SRCFILE parameter that contains the code block to be copied.

Some points to note about using VC_COPY commands are:

- The nominated source member is **not** copied if a previously processed VC_COPY command nominated the same source file, library and source member.
- The nominated source member **cannot** contain VC_COPY or VC_USING commands. If they are encountered they are totally ignored and treated like comments.
- The nominated source member **cannot** contain RPG compiler directive /COPY commands. If they are used, they will cause RDML functions to fail to compile.
- VC_COPY commands are usually only used to copy code blocks that may be common or shared between different file definitions. It would be **unusual**

to use this facility to include "Before output" or "After input" C specification code blocks.

- The command and its parameters should only ever be coded between columns 8 -> 80 of the source program line. LANSA ignores columns 1 -> 7 of the line when processing a VC_COPY command and its parameters.
- Where a command and its parameters will not fit on one line use a "+" sign as the last character on the line and continue the command in column 8 (or beyond) on the next line. This method is identical to that used when continuing IBM i operating system commands over multiple lines. Note that the "-" sign continuation character supported by the IBM i is **NOT** supported by LANSA in this context.

VC_COPY Examples

Some examples of VC_COPY commands follow.

Example 1:

To specify that a code block contained in a source member called CUSMSTVIRI in source file QRPGSRC be copied in as virtual code, the following would be input as virtual field derivation code of type "Data Structure Specifications":

```
I*
I* VC_COPY SRCFILE(QRPGSRC) SRCMBR(CUSMSTVIRI)
I*
```

Example 2:

To specify that a code block contained in a source member called DATER001 in source file QVIRSRC be copied in as virtual code, the following would be input as virtual field derivation code of type "Internal Subroutines":

C* C* VC_COPY SRCFILE(QVIRSRC) SRCMBR(DATER001)

C*

Examples of Virtual Fields & Derivation Code

This section details some specific examples of virtual field definition and the RPG derivation code required to establish the virtual fields.

Example 1: Sub-stringing a Field

Example 2: Accumulation Of Fields

Example 3: Converting a Date of Birth to an Age

Example 4: Getting Information from Another File

Example 5: Accessing an Internally Described File

Example 1: Sub-stringing a Field

Break real field CHTACC (alpha 8 - chart of accounts) down into 3 virtual fields called COMPNO (alpha 2 - company number), DIVNUM (alpha 2 - division number) and COSTCT (alpha 4 - cost center).

Step 1 - Define the Virtual Fields

Seq	Name	Description
10	COMPNO	Company number
20	DIVNUM	Division number
30	COSTCT	Cost centre

Step 2 - Input RPG Section "Calculations After Input From File" C* C* VC_USING FIELDS(CHTACC COMPNO DIVNUM COSTCT) C* C MOVELCHTACC COMPNO C MOVELCHTACC CHAR04 4 C MOVE CHAR04 DIVNUM C MOVE CHTACC COSTCT

Note the use of "work" field CHAR04. It is not referred to in the VC_USING command as it not part of the file definition.

Example 2: Accumulation Of Fields

A sales file contains monthly sales figures in real fields SAL01 to SAL12.

Generate a virtual field called SALYR to contain the total yearly sales and a virtual field called SALAV to contain the average monthly sales.

Also generate 4 virtual fields called SALQ1 -> SALQ4 to contain total quarterly sales.

Step 1 - Define the Virtual Fields

Seq	Name	Description		
10	SALYR	Total sales for year		
20	SALAV	Average monthly sales		
30	SALQ1	Total first quarter sales		
30	SALQ2	Total second quarter sales		
30	SALQ3	Total third quarter sales		
30	SALQ4	Total fourth quarter sales		

Step 2 - Input RPG Section "Calculations After Input from File"

>C* C* VC_USING FIELDS(SAL01 SAL02 SAL03 SAL04 SAL05 SAL06 SAL0 С* SAL08 SAL09 SAL10 SAL11 SAL12 SALYR) C* C* Calculate yearly sales C* С Z-ADDSAL01 SALYR С ADD SAL02 SALYR С ADD SAL03 SALYR

С ADD SAL04 SALYR С ADD SAL05 SALYR С ADD SAL06 SALYR С ADD SAL07 SALYR С ADD SAL08 SALYR С ADD SAL09 SALYR С ADD SAL10 **SALYR** С ADD SAL11 SALYR С ADD SAL12 **SALYR** C* C* VC_USING FIELDS(SALYR SALAV) C^* C* Calculate average monthly sales C^* С DIV 12 SALYR **SALAV** С* C* VC_USING FIELDS(SAL01 SAL02 SAL03 SALQ1) С* C* Calculate first quarter sales C* С Z-ADDSAL01 SALQ1 С ADD SAL02 SALQ1 С SALQ1 ADD SAL03 C* C* VC_USING FIELDS(SAL04 SAL05 SAL06 SALQ2) C* C* Calculate second quarter sales C^* С Z-ADDSAL04 SALQ2 С ADD SAL05 SALQ2 С ADD SAL06 SALQ2 C* C* VC_USING FIELDS(SAL07 SAL08 SAL09 SALQ3) C* C* Calculate third quarter sales C^* С Z-ADDSAL07 SALQ3 SALQ3 ADD SAL08 С С ADD SAL09 SALQ3

C* C* VC_USING FIELDS(SAL10 SAL11 SAL12 SALQ4) C* C* Calculate fourth quarter sales C* C Z-ADDSAL10 SALQ4 C ADD SAL11 SALQ4 C ADD SAL12 SALQ4

Note the use of multiple VC_USING commands in the source code. A VC_USING command remains in effect until another VC_USING is encountered or the end of the code section is encountered.

Example 3: Converting a Date of Birth to an Age

An employee master file contains field EMPDOB (date of birth). Using existing program GETAGE (convert date to age) derive virtual fields AGEYER (age in years) and AGEMON (age in months).

Step 1 - Define the Virtual Fields

Seq Name_____ Description_____

- 10 AGEYER Age in years
- 20 AGEMTH Age in months

Step 2 - Input RPG Section "Calculations After Input from File"C*C*C* VC_USING FIELDS(EMPDOB AGEYER AGEMTH)C*CCCALL 'GETAGE'CPARMEMPDOBCPARMAGEYERCPARMAGEMTH

Example 4: Getting Information from Another File

An invoice header file contains a customer number (CUSTNO). Add the customer name (CUSNAM) to this file as a virtual field.

The customer name must be extracted from the customer master file which is called CUSTMST.

Step 1 - Define the Virtual Fields

Seq Name_____ Description_____

10 CUSNAM Customer name

Step 2 - Input RPG Section "File Specifications"

FCUSTMST IF E K DISK

Step 3 - Input RPG Section "External Record Format or Field Renames"

```
I*
I* VC_USING FIELDS((CUSNAM $$$$$))
I*
IRCUSTMST
I CUSNAM $$$$$
```

Step 4 - Input RPG Section "Calculations After Input from File"

```
>
C*
C* VC_USING FIELDS(CUSTNO CUSNAM)
C*
C MOVE *BLANKS CUSNAM
C MOVEL'N/AVAIL' CUSNAM
C CUSTNO CHAINRCUSTMST 70 70=IGNORED
```

Note:

• In the external field rename section the \$\$\$\$\$ name has been used to cause the internal name of CUSNAM to be inserted at the correct point without

replacing the CUSNAM value which must remain unchanged in this context.

• A simpler solution to this example would be to create an access route from the invoice header file to the customer master file with a 1 to 1 relationship and define a predetermined join field on the access route to receive the customer name. For further information on access routes and predetermined join fields refer to What Is a Predetermined Join Field?.

Example 5: Accessing an Internally Described File

A transaction file called S3TRANS has no external description.

The file record is one 14 character field called DATA, but it is actually made up of 3 fields PRODNO (product number - alpha 7), WHOUSE (warehouse - alpha 2) and QUANTY (quantity - packed 9,0).

Define these 3 fields as virtual fields to allow **read and write** access to the file as if it were an externally described file.

Step 1 - Define the Virtual Fields

Seq	Name	Description		
10	PRODNO	Product number		
20	WHOUSE	Warehouse identifier		
30	QUANTY	Quantity supplied		

Step 2 - Input RPG Section "Data Structure Specifications"

Ι	1 7 EXT001
Ι	8 9 EXT002
Ι	P 10 14 EXT003

Step 3 - Input RPG Section "Calculations After Input from File" C*

C* VC_USING FIELDS(DATA PRODNO WHOUSE QUANTY) C* C MOVELDATA EXTDTA

0	INTO I EEDITIIIT	
С	MOVELEXT001	PRODNO
_		

- C MOVELEXT002 WHOUSE
- C Z-ADDEXT003 QUANTY

Step 4 - Input RPG Section "Calculations Before Output to File" C*

C* VC_USING FIELDS(DATA PRODNO WHOUSE QUANTY) C* C MOVEL PRODNO EXT001

C		LAIUUI
С	MOVELWHOUSE	EXT002
С	Z-ADDQUANTY	EXT003
С	MOVELEXTDTA	DATA

Note the "double shuffle" of all fields via data structure EXTDTA is required because LANSA declares all real and virtual file fields in a data structure. Thus it is not possible to directly declare the fields in another data structure or the I/O module will fail to compile.

Appendix B. System Data Areas

This appendix contains details of the main LANSA System Definition Data Areas.

- System Definition Data Area Overview
- Changes to the System Data Areas
- System Definition Data Area DC@A01
- Extended Information Data Area DC@A07
- Version Dependency Data Area DC@OSVEROP
- Import/Export Message Summary DC@RESLT

Other specific LANSA data areas are defined in the guide for the software to which they relate.

For reviewing and amending the details stored in these data areas, use the System Settings facility available from the Administration menu and described in Review System Settings.

System Definition Data Area Overview

The system definition data areas are used to tailor a number of values and defaults used by LANSA for your installation.

- The first and most important system definition data area is DC@A01.
- The second is the extended information data area DC@A07.
- The third, an optional data area for operating system and LANSA version dependencies, is called DC@OSVEROP.

A facility to review and change the values of the system settings in the data areas is available via the Administration Menu. The settings are available by category groupings, regardless of the data area in which they are recorded.

If you change these data area settings directly, you must exercise care and caution to ensure that you change the correct data area and any associated settings. Before you can change them, you must be signed on as either the operating system security officer QSECOFR or the LANSA system owner profile.

If you have installed multiple LANSA systems then you will have multiple data areas (one for each system). Ensure that the correct one is being referenced before it is changed.

It is a good idea to save or duplicate (with another name) the existing system definition data areas before changing them. Thus if the changes cause unexpected problems or the wrong part of the data area is changed it is easy to revert back to the old data area and start again.

Changes to the System Data Areas

Whenever changes are made to data areas DC@A01, DC@A07 or DC@OSVEROP settings, please ensure that these "post-change" procedures are completed:

- 1. Reflect the change(s) identically into **all** LANSA systems installed on **all** IBM is in your organization.
- 2. When Visual LANSA users are involved, use the LANSA REQUEST(PCMAINT) command to (re)export the changed system and/or partition definitions into a shared folder. Then use the "Import" facility from the "Tools" action bar item to import the changed system definition into **each and every** PC system.

The maintenance of identical system and partition configuration options across all LANSA systems on all IBM is and PC(s) is vital to the smooth running of a LANSA system.

System Definition Data Area DC@A01

For reviewing and amending the details stored in these data areas, use the System Settings facility available from the Administration menu and described in Review System Settings.

This area is formatted as follows:

From Byte	To Byte	Byte Length	Туре	Dec Pos	Description
1	10	10	А		Name of program library
11	20	10	А		Name of data library
21	30	10	А		Partition module library (execution time)
31	40	10	А		Name of system owner profile
41	45	5	А		Name of product (LANSA)
46	48	3	S	1	LANSA Version level
49	78	30	А		Name of company / organization
79	88	10	А		Partition security officer (execution time)
89	98	10	А		Spare Field
					The following six function key assignments apply to internal LANSA programs in non-SAA/CUA partitions only. In SAA/CUA partitions the SAA/CUA partition function key assignments are used.
99	100	2	S	0	EXIT/SYSTEM function key
101	102	2	S	0	MENU/CANCEL function key
103	104	2	S	0	MESSAGES function key
105	106	2	S	0	ADD function key
107	108	2	S	0	CHANGE function key

109	110	2	S	0	DELETE function key
111	111	1	А		Partition unique prefix (execution time)
112	141	30	А		Default file SIZE parameter
142	145	4	А		Default file LVLCHK parameter
146	146	1	А		Compile - source listing default (Y or N)
147	147	1	А		Compile - optimize compile default (Y or N)
148	148	1	А		Compile - decimal data error default (Y or N)
149	151	3	S	0	Default printer form width (80 - 198)
152	154	3	S	0	Default screen width is always 80 and cannot be changed.
155	157	3	S	0	Default screen length is always 24 and cannot be changed.
158	197	40	А		Default input attributes- alpha field
198	237	40	А		Default input attributes- numeric field
238	277	40	А		Default output attributes - alpha field
278	317	40	А		Default output attributes - numeric field
381	390	10	А		Partition default file library (execution time)
391	391	1	А		Default commitment control - new file definition
392	392	1	А		Spare Field
393	393	1	А		Default RDML command editor
					Values:
					"E"- edit by IBM's SEU or EDTSRC editors.
					"U"- edit by alternate user exit editor.
				other - edit by standard LANSA editor.	
-----	-----	---	---	--	
394	394	1	А	Was LANSA version number. No longer used.	
				Do not adjust this value.	
395	396	2	А	Export LANSA object authority (default = "MD")	
				Values:	
				"UD"- Use authority	
				"MD"- Modify authority	
				"DD"- Delete authority	
397	397	1	А	System uses ASPs (Auxiliary Storage Pools) other than 1 for partition module or file libraries.	
				Values:	
				"Y"- System uses ASPs (other than 1) for one or more partition libraries.	
				other - System does not use ASPs, or only ASP number 1 is used, for all partitions.	
				However, partition module or file libraries may be configured in differing ASPs. When any partition file or module library is configured into an ASP (other than ASP number 1) you must set this flag to "Y" to indicate this configuration.	
400	400	1	А	Compile - enable debug default (Y or N)	
401	401	1	А	Enable alternate editor user exits when editing help text and/or RDML commands.	
				Values:	
				"Y"- enable alternate editor exits.	
				other - do not enable alternate editor exits.	
				Alternative editor exits are enabled by standard user exit programs. Refer to User Exit Programs for more details.	

402	411	10	А	Dictionary interface program name
412	414	3	А	Default printer form length (RPG limit)
415	417	3	А	Default printer form overflow line (RPG limit)
418	418	1	А	Default IDENT_ATR parm to include high intensity (Y or N). Ignored in SAA/CUA partitions.
419	419	1	А	Default IDENT_ATR parm to include reverse image (Y or N). Ignored in SAA/CUA partitions.
420	420	1	А	Default IDENT_ATR parm to include underline (Y or N). Ignored in SAA/CUA partitions.
421	421	1	А	Default STD_HEAD parameter to be used (Y or N)
422	422	1	А	Use execution time locking on processes (Y or N)
423	424	2	А	LANSA minor version level.
425	434	10	А	Name of "HANDS-ON" program library
436	436	1	A	Enable cursor location sensitivity (Y or N) NOTE: This flag enables field prompting and field level help. Recompilation is necessary for changes to this flag to take effect.
437	438	2	А	IBM i being used (38/3X)
439	440	2	А	Execution environment if on IBM i (38/3X)
441	460	20	А	Hexadecimal codes for all 132 wide workstations Refer appropriate IBM manual for list of codes This is only used by the report painter to aid the user design

report layouts.

461	461	1	А	Compile - compile process default (Y or N)
462	462	1	А	Enable LANSA attention key handling (Y or N)
463	463	1	А	Field label fill character
				Recompilation is necessary for changes to this setting to take effect.
464	464	1	А	Field column heading underline character Recompilation is necessary for changes to this setting to take effect.
465	474	10	А	Hexadecimal codes for all color workstations
475	475	1	Α	System uses function level security. Note that using function level security increases overall system resource usage. Processes (NOT functions) compiled prior to a change to this value will ignore the change and should be recompiled.
				Values:
				"Y"- function level security is used
				other - only process level security is used
476	476	1	A	Automatically generate field level help text from dictionary definition and validation rules. Automatically generated help text is appended to any user defined help text that is applicable.
				Values:
				"Y"- automatic generation is enabled
				other - automatic generation is NOT enabled
477	477	1	А	Controls the default enabling of the PROMPT_KEY parameter on DISPLAY,

				REQUEST and POP_UP RDML
				commands.
				"Y"- prompt key is enabled unless the command specifically disables it. other - prompt key is disabled unless the command specifically enables it.
478	479	2	Α	Specifies the prompt function key number for any non-SAA/CUA applications in the system. If you intend to use the prompt key in any non-SAA/CUA application, this area must contain a valid function key number in the range "01" to "24". For SAA/CUA applications the prompt function key number defined at the partition level is used instead of any value defined here.
480	480	1	А	Controls the execution of a RCLRSC (reclaim resources) command on exit from LANSA, which closes all files opened by LANSA.
				Values:
				"Y"- execute RCLRSC on exit from LANSA
				other - do not execute RCLRSC on exit from LANSA.
481	481	1	А	Controls the execution of a RPLLIBL (replace library list) command on exit from LANSA, which resets the job's library list to exactly what it was on entry to LANSA.
				Values:
				"Y"- execute RPLLIBL on exit from LANSA
				other - do not execute RPLLIBL on exit from LANSA.

482	482	1	Α	Indicates if pre-release 2.5 LANSA I/O modules are required. These are programs in the file library prefixed by "I@" that are only required by programs compiled prior to LANSA Release 2.5.
				Values:
				"N"- pre-release 2.5 I/O modules are not required.
				other - pre-release 2.5 I/O modules are required.
483	483	1	Α	This value is no longer used to determine the TGTRLS value that LANSA compiles and saves are to use.
				Refer to positions 631-638 of the Extended Information Data Area DC@A07 for this information.
484	484	1	Α	Vertical bar character to be used on reports such as RDML listings. If this value is left as blank (X'40') it defaults to ":" (X'6A'). This may not be suitable for some national language sets and should be changed.
485	485	1	Α	Horizontal bar character to be used on reports such as RDML listings. If this value is left as blank (X'40') it defaults to "-" (X'60'). This may not be suitable for some national language sets and should be changed.
486	486	1	Α	External Security Matching. This is set to a "Y" to indicate that LANSA database security should be reflected into IBM i object level security. This option only applies to files set up and maintained by LANSA. It does not apply to "OTHER" files. Refer to the security references in Reviewing and Changing Access to

LANSA Objects for more details of this option.

487	487	1	A	Y allows exchange list capability in RPG / CL / COBOL etc programs so that values can be returned from LANSA to these programs (using EXCHANGE). Values can be put on the LANSA exchange list from RPG / CL / COBOL etc programs. Refer to the EXCHANGE command in the <i>Technical Reference Guide</i> for more details. Recompilation is necessary for changes to this setting to take effect. N is the default.
488	489	2	A	Comparison year for determining the century to be used. Six and four character dates have their year portion compared against this value so that the correct century will be used.
490	491	2	А	Century to use if the year is less than or equal to the comparison year value.
492	493	2	А	Century to use if the year is greater than the comparison year value.
494	494	1	Α	Decimal format for LANSA. This should relate to the IBM i system value QDECFMT. If QDECFMT = " " (blank) then the LANSA decimal format will = "." or If QDECFMT = "I" or "J" then the LANSA decimal format will = ",". Values:
				" "- I ANSA decimal format is a full stop
				" "- LANSA decimal format is a comma
				Other - LANSA decimal format is defaulted to a full stop.

WARNING: If the value of this field does not relate to the value of QDECFMT then UNPREDICTABLE results could occur with the use of numeric values.

495 495 1 A Controls suppression of the "Process will run in interpretive mode, compiled version not usable" message. This message is displayed as a warning when a process has been selected for use, informing the user that a "new" compiled version of a Process should be created.

Values:

496

496

1

Α

Y- do not display warning message. N - display warning message is the default.

Indicates that end user process and function level security is disabled in this system. This setting will improve system performance, because no security checking is performed before accessing end user applications. This flag has no effect on access to process or function definition details in a development environment. This setting is appropriate for installations that use an external menu system to control and secure access to LANSA applications. Additional security checking within the LANSA application is a waste of resource and may mean "double updating" of end user access rights. The setting of this flag is interpreted dynamically by all applications that have been recompiled since the installation of program change 4.0/D3.

Values:

"Y"- process and function level security is disabled.

Other - process and function level security

is enabled.

Indicates that end user file level security is А disabled in this system. This setting will improve system performance, because no file level security checking is performed in end user applications. This flag has no effect on access to file definition details in a development environment. This setting is appropriate for installations that use the approach "if the program is accessible from your menu then you can access all the files the program requires". The setting of this flag is interpreted dynamically by all applications that have been recompiled since the installation of program change 4.0/D3.

Values:

"Y"- file level security is disabled. Other - file level security is enabled.

Indicates whether a user defined prompt key message handler program should be called in preference to the LANSA prompt key message handler, when the prompt key function key has been selected. If this value has been set to "Y", a user defined prompt key message handler program name must be specified in the extended information data area DC@A07, please refer to the section on the 'extended information data area DC@A07' for more information.

Values:

"Y"- call user defined prompt key message handler program.

Other - call the LANSA prompt key message handler program.

498 498 1

А

497 497 1 A

499	499	1	А	EXCHANGE all fields from one function to another on a prompt request.
				Values:
				"Y"- EXCHANGE all other fields used by a function as will fit into the space left in the exchange list.
				Other - Do not include all other fields used by a function in the exchange list.
500	500	1	А	Spare

Extended Information Data Area DC@A07

For reviewing and amending the details stored in these data areas, use the System Settings facility available from the Administration menu and described in Review System Settings.

This area is formatted as follows:

From Byte	To Byte	Byte Length	Туре	Dec Pos	Description
1	10	10	А		Name of user defined Program/File error handler.
					When a program/file error has been detected, an attempt will be made to call a user defined program/file error handler program during the LANSA program/file error routine. The user defined program/file error handler will provide a means of updating audit/error logs external to the LANSA system.
11	20	10	A		Name of user defined Prompt Key message handler program. If the user defined prompt key message handler has been specified for use in the system definition data area DC@A01(position 498 = "Y"), then an attempt will be made to call a user defined prompt key message handler program when the prompt key function key has been selected.
21	560				Used for Developer Services Configuration. Refer to <i>Developer Services</i> for more details.
601	610	10	A		Import User Exit program name. During a standard import these bytes are tested. If non-blank they are assumed to identify a user exit program that should be invoked each time that an IBM i level object is

created or changed during the import. Such a program must receive 3 x Char(10) parameters identifying: (1) the object name, (2) the library, and, (3) the object type (in IBM i "*" style object type notation). This facility is provided only for IBM i.

Set Environment User Exit program name. After LANSA has set up its environment for the partition these bytes are tested. If non-blank they are assumed to identify a user exit program that should be invoked

each time that a LANSA partition is

entered. Such a program must receive 3 parameters identifying: (1) the LANSA program library Char(10), (2) the partition

id Char(3), and (3) the language code

611 620 10

А

621 630 10 A

User Exit program name for SUBMIT command. The first time a LANSA function is submitted these bytes are tested. If non-blank they are assumed to identify a user exit program that should be invoked each time a function is submitted. Such a program must receive one parameter Char(2000) containing the SBMJOB command about to be used. The user exit program can modify the SBMJOB command before exiting and passing back the parameter.(e.g. Add MSGQ() parameter to SBMJOB).

631 638 8 A Target OS Release. Specifies the TGTRLS value that IBM i compiles are to use. The TGTRLS parameter is appended to LANSA compile and export requests.

Char(4).

				Valid values include: "*CURRENT" "*PRV" "VxRyMz" where "VxRyMz" is a valid IBM i version. "*N" which means that the default for the command is used. New LANSA systems are shipped with this value. If left blank, the target release used is *CURRENT, not the default for the command.
639	639	1	A	Retain task history flag. This can take the values "Y" or "N". "Y" means that task history is to be retained. "N" means that the next Request Submit LANSA Re-org job will have Purge Task History starting from date pre-filled with today's date.
640	645	6	A	Task tracking category 2 character prefix. The 1st is for task for the system, 2nd is for a task for each user and the 3rd is for task for each "task". Each is optional.
646	648	3	Α	Task tracking category unlock object after Check In flags. Each flag is 1 character long and its value can be either a "Y", "N" or " ". The value of this flag determines whether an object is unlocked from the task of this category when the object is checked in. If this flag is "N" or " ", then the normal task tracking object locking applies. The 1st is for task for the system, 2nd is for a task for each user and the 3rd is for task for each "task"
649	668	20	А	User exit for Export/Import message logger

				649 to 658 must contain library name. It can be *LIBL.
				659 to 668 must contain the program name.
				The LANSA Export/Import message logger will pass the following parameters and they are as follows: DC@IDS type (1024) char Output only Message type (1) char I/O Message Id (7) char Output only Message text (132) char Output only
669	688	20	А	User exit for LANSA access check to an object.
				669 to 678 must contain library name. It can be *LIBL.
				679 to 688 must contain the program name.
				The LANSA security check programs will pass the following parameter to the user exit program: DC@IDS (1024) Output Only Object Name (10) Output Only
				Object extension (10) Output Only Object Type (2) Output Only Access (2) Output Only
				Return Code (1) Input/Output. Valid
				Values Y and N User Exit Return Code (1) Input/Output. Valid values C and N.
689	708	20	А	User exit for IJDT (DC@P8097)
				689 to 698 must contain library name. It can be *LIBL.
				699 to 708 must contain the program name
				The IJDT programs will pass the following parameters to the user exit program: DC@IDS (1024)
				DC@EDS (2500)

				Object Name (10) Object Extension (10) Object Type (2) Event code (3) Message Id (7) Message file name (10) Message data (132) Return Code (1) User Exit Return Code (1) Event Source (10) Event LockInd (1) Event SubCode (3)
709	728	20	A	User exit for CIDT (DC@P8096) 709 to 718 must contain library name. It
				can be *LIBL. 710 ± 720 must contain the average room.
				719 to 728 must contain the program name. The LANSA CIDT programs will pass the following parameter to the user exit program: DC@IDS (1024) DC@EDS(2500) Object Name (10) Object Extension (10) Object Type (2) Return Code (1) Input/Output.Valid values are Y and N. User Exit Return Code (1) Input/Output. Valid values are C and N. Type of call (4) 'CIDT' or 'VTSK'.
729	748	20	A	Default value for Export \$\$DTALIB\$\$ Substitution Variable. This value is used when export lists are created.
749	749	1	А	Y or N. Default value for <i>Prompt user when importing</i> on \$\$DTALIB\$\$ substitution variable in export list. This

value is used when export lists are created.

750 750 1 A Spare.

Version Dependency Data Area DC@OSVEROP

The version dependency data area DC@OSVEROP is optional. It may or may not exist on your system.

DC@OSVEROP is a different style of data area. Rather than setting a special byte within the data area to enable an option you simply include a pre-defined text string anywhere at all within it.

These Strings are also converted to Y/N flags or values and exported to Visual LANSA in file LX_F96 when a system export, *PLUGIN or *REFRESH is performed. They can be changed in Visual LANSA through the *Configure* menu.

DC@OSVEROP may contain one or more of the following strings:

String	Meaning / Comments
*CHECKBOX=x	x defines the character used to depict check boxes when displayed on the screen painter, or printed on a screen design report. The default character is X'9F'.
*CRUDEWARNONLY	Indicates that a function that would cause the "Crude Element Complexity Rating" to return a FATAL will cause a WARNING only. It is not recommended to use this setting as the function may subsequently fail to compile.
*DBCS_BUILD	Indicates to always build IBM i RDML objects (DDS files and programs) for DBCS, irrespective of the current build language. This value does not impact the build of RDMLX objects on IBM i or the build of Windows objects. When *DBCS_BUILD is not present in DC@OSVEROP, RDML IBM i objects will be built according to the current build language type. When *DBCS_BUILD is present in DC@OSVEROP, all RDML IBM i objects will be built as though the current build

	language is a DBCS language. For example, all DBCS keyboard shift attributes of J, E or O for fields will be generated into the DDS.
*DROPDOWN=x	x defines the character that follows a field to indicate that it is drop down capable. This character is also used when the field is displayed on the screen painter, or printed on a screen design report. The default character is 'V'.
*EXTENDED_TRIGGERS	Indicates that the normal LANSA rules that prevent database event triggers from calling other functions and using "user interface" commands should be relaxed. This use of this option is not recommended in most circumstances.
*FUNRTRLIBL	Controls the use of Function Routing table X_FUNRTR from *LIBL. If this is not set the Function Routing table in the partition module library will be used. See description of Function Routing for further details.
*HSTABEXTEND	Allows database files with record lengths up to 1988 bytes to be added to a user index for high speed lookup. WARNING: Refer to Database File Attributes
	before using this option.
	WARNING: It is strongly recommended that if option *HSTABEXTEND is added to system data area DC@OSVEROP to make the extended entry record length available, or is removed to limit entry length, that all files tagged as high speed tables, all read only functions that use these files and all other I/O modules and Dboptimized functions that use high speed tables for lookup validation rules be recompiled AFTER deleting the current user index which is DC@TBLIDX if adding

	*HSTABEXTEND, DC@TBLIDY if removing *HSTABEXTEND.
*ILE See Note 2	Activates the second level of ILE implementation. That will statically bind any GUI and multilingual program into the function program and use supplied service programs to dynamically call (CALLB) LANSA internal programs. *** Must be used in conjunction with *RPGIV ***
*IMPREFFLDNOPROP See Note 3	If present when an Import executes, it indicates that Reference Field characteristics are not propagated to fields that reference those Reference Fields. Note that if a Reference Field is changed subsequently, the changes are propagated to the fields that reference it, as usual.
*IOMBLOCKBYKEY See Note 1	Use this option to indicate that I/O modules should be compiled to support high speed record blocking in physical file key order. You must use this option when using LANSA Client. You must use this option when using the LANSA SuperServer *BLOCKBYRRNnnnn selection option or the "receive immediate" option. You must use *IOMXSERVER if you use this option.
*IOMBLOCKBYRRN #1	Use this option to indicate that I/O modules should be compiled to support high speed record blocking in relative record number order. You must use this option when using LANSA Client. You must use this option when using the LANSA/ Server *BLOCKBYRRNnnnn selection option or the "receive immediate" option. You must use *IOMXSERVER if you use this option.
*IOMNOADOPT	Indicates that all I/O modules created by LANSA are to have USEADPAUT(*NO) i.e.

	do not use program adopted authority for I/O modules.
*IOMXSERVER See Note 1	Use this option to indicate that I/O modules should be compiled to allow support of:
	LANSA Client applications.
	and/or LANSA Open applications using blocked I/O methods or the "receive immediate" option and/or Visual LANSA SuperServer applications.
	It is recommended that you set the full list of values *IOMXSERVER, *IOMBLOCKBYKEY and *IOMBLOCKBYRRN into data area DC@OSVEROP as a system default for all LANSA systems.
*LEADZERO	'1' indicates that Visual LANSA code will display numeric values containing decimals with a leading zero. For example, 0.12 or 0,12 depending on the LANSA decimal point format. This option would be suitable when QDECFMT='J'.
*LONG_USER_AUDIT	Indicates that user stamping (USRC, USRU, USRX) fields will be supplied with a user name of up to 256 characters. Refer to detailed information in Output Stamping Attributes for more information.
*MINISCREENMSGLIN=xx	xx specifies the line that IBM i messages will appear when using the *MINI_SCREEN facility. The full function checker does NOT validate that the line specified falls within the pop-up windows used.
*NOL4WCOMP	Disables Visual LANSA component-related information from being transferred between a LANSA for the IBM i system and Visual

	LANSA systems. The flag affects the export, check out and check in functions. Warning: Use of this flag should be considered carefully to avoid loss of component details.
*NOLADCOMP	Controls how Visual LANSA component- related information is exported and imported between LANSA for the IBM i systems.
	Warning : Use of this flag should be considered carefully to avoid loss of component details.
*NOWEBEXP	Disables the export of all Web details. This includes Web components as well as web details associated with fields, functions and system variables.
*NOWEBIMP	Disables the import of all Web details. This includes Web components as well as web details associated with fields, functions and system variables.
*NOXMLEXP	Disables the export of all XML details. This includes XML components as well as XML details associated with fields, functions and system variables.
*NOXMLIMP	Disables the import of all XML details. This includes XML components as well as XML details associated with fields, functions and system variables.
*ODBC #1	Use this option to indicate that I/O modules should be compiled to allow support of the LANSA Open (LANSA Open) ODBC Interface.
	You must use *IOMXSERVER if you use this option.
*OTHER_DATETIME	Indicates that the conversion option

	*DATETIME is to be used when OTHER file I/O modules are compiled. This allows date (L), time (T) and timestamp (Z) fields to be accessible in LANSA.
*OTHER_VARCHAR	Indicates that the conversion option *VARCHAR is to be used when OTHER file I/O modules are compiled. This allows variable length (VARLEN or varchar) fields to be accessible in LANSA.
	Note that this setting does not allow variable length character fields to be used as keys within LANSA. If the physical file or any logical views made known to LANSA have a varchar field as a key, the I/O module will fail to compile.
*PERMFILOVR	Allows permanent file overrides to be used. When you specify permanent file overrides you are telling LANSA that "every time I use this file, I really want to use this other file". This is useful when you want to use files with 10 character file names, files with a "." in their name, etc. Refer to The Permanent File Overrides Facility for details and examples.
*RADIOBUTTON=x	x defines the character used to depict radio buttons when displayed on the screen painter, or printed on a screen design report. The default character is X'80'.
*RPGIV See Note 2	Specifies that each program is to be compiled using RPG/IV code, then bound as a single module ILE type program.
*SELECTCHAR=x	x indicates the character used to select an option from a check box or radio button group. The default character is '/'
*SQL_BUILD	Indicates to build physical files and logical files on IBM i using SQL as much as possible

	when in an RDMLX partition. The physical file will be built as an SQL table. This keyword will have no effect on how logical files are built, and logical files sharing the access path of a corresponding SQL index no longer occurs.
	Note that there are some valid CRTPF /CHGPF parameters that will not be able to be applied to physical files created as tables using SQL, and similarly some valid CRTLF / CHGLF parameters that will not be able to be applied to logical files created as indexes. One such parameter for both is MAXMBRS with a value other than 1 or *SAME.
	RDML files built when the *SQL_BUILD option is in effect can be imported to RDML partitions provided they do not have DB triggers set on, however if they are rebuilt in the RDML partition they lose the effects of the *SQL_BUILD option.
*TTG6FUNCLOCKING	Disables the new task tracking logic that enforces that all functions are locked with the same Task Id as the parent process. This flag only applies when the 'Allow user to change tasks while working?' task tracking configuration flag is set to no.
*UIMHELP	Indicates that the IBM's Panel Groups are to be used for the presentation of user defined help text, rather than the LANSA help text display facility. *V2WINDOWS must also be selected.
*V2WAF/400	As the parameters used to call Workfolder Application Facility/400 have changed with V2 this option is used to determine which parameters to use.
*V2WINDOWS	Indicates that pop-up windows created by

LANSA should use the IBM i windowing facilities.

*WEBNUMVAL Indicates that the default for LANSA for the WEB process compile option 'Validate numerics' should be 'YES'. Refer to the section Compiling a Process from New or Amended Definition for details of this option. This also indicates that the COMPILE_PROCESS Built-In function default should be 'YES' rather than 'NO'.

*WINDOWTRIM Indicates that an existing LANSA defined pop-up window has the 2nd function key line trimmed from the display.

Only use this option in conjunction with the *V2WINDOWS option to enable existing functions that have been specifically sized onto line 24 of the display device to recompile without change.

Note 1 *IOMXSERVER, *IOMBLOCKBYKEY, *IOMBLOCKBYRRN and *ODBC increase the number of files declared in an I/O module, the amount of static (literal initialized) storage used by an I/O module and the number of subroutines in an I/O module. Note also that if one of *IOMBLOCKBYKEY, *IOMBLOCKBYRRN or *ODBC are specified, *IOMXSERVER must also be specified.

Note 2 RPGIV and ILE: Before you attempt to use any of the RPGIV and ILE related switches it is highly recommended that you first read the ILE **Implementation** section of this guide.

If you have grossly exceeded the recommended limits for the number of logical views created (or made known to LANSA) then you may find that an existing I/O module may not (re)compile when these options are used.

If you have grossly exceeded the recommended number of real or virtual fields in a physical file then you may find that an existing I/O module may not (re)compile after these options are used. In either case, temporarily remove the options from data area DC@OSVEROP while recompiling the I/O module that is experiencing the problem.

Also note that RPG/IV (V3 version of RPG from IBM) has removed the total file, total static initialized storage and total subroutine limits that may be causing such problems to occur.

Note 3 *IMPREFFLDNOPROP is also used by the host monitor and LANSA Import to decide if reference field changes should be propagated. Prior to V9.1 no updates were performed during these operations and "no update" is the initial default setting.

This flag now offers the choice of propagating changes or not.

The use of this flag is not recommended as fields may become out of synch with their nominated reference field and therefore it should be removed from DC@OSVEROP and set to 'N' in Visual LANSA.

When this flag is **NOT** set to 'N' the input and output attributes to a field which references another field are not protected. This was was the default before V9.1.

It should be remembered that when a field which is referenced by other fields is changed using the field maintenance options or the PUT_FIELD Built-In Function, all the referencing fields are also updated.

Import/Export Message Summary DC@RESLT

Some LANSA operations create a data area into library QTEMP when they complete.

The content of this data area can be retrieved and examined to test the result of the operation.

The content and layout of the data area depends upon the operation that was performed.

Operation: LANSA export job is executed.

DC@RESLT Layout when operation completes

From Byte	To Byte	Dec	Format	Description
1	10		Alpha	Contains literal 'EXPORT'.
11	17	0	Signed	Total fatal messages issued.
18	24	0	Signed	Total warning messages issued.
25	31	0	Signed	Total completion messages issued.
32	38	0	Signed	Total error messages issued. This value is updated by the user exit program otherwise it will contain zeros.

Appendix C. Technical Notes

In this appendix you will find technical details of:

- User Defined Program/File Error Handler
- User Defined Prompt Key Message Handler
- User Exit Programs
- The Permanent File Overrides Facility
- The @@UPID Field in LANSA Created Files
- Commitment Control
- Using *DBOPTIMIZE / *DBOPTIMIZE_Batch
- Trigger Functions
- User Defined Reporting Attributes
- Hindi Numerics
- GUI WIMP Constructs
- ILE Implementation
- Function Routing
- Built-In Function Notes

User Defined Program/File Error Handler

To enable program/file errors which occur within LANSA or LANSA user applications to be recorded, a User Defined program/file error handler program can be called. To invoke the specified program these steps must be taken:

- 1. Enter the name of the user defined program/file error handler program into the *Name of user defined program/file error handler* in Work with Execution and Security Settings.
- 2. The user defined program/file error handler program must be set up to receive 3 parameters which are as follows:

Parameter Type Length Description Number

1	А	10	Program name - contains the program name in which a program/file error has occurred.
2	А	1	Type of Error - contains the type of error which has occurred.
			Values: "F" = File Error occurred "P" = Program Error occurred
3	A	varies	Information. If the error was an "F" type error, this parameter will contain the data from the File Information Data Structure (for the program defined in parameter 1) at the time the file error occurred. If the error was a "P" type error, this parameter will contain the data from the Program Status Data Structure (for the program defined in parameter 1) at the time the program error occurred.

User Defined Prompt Key Message Handler

To invoke the user defined prompt key message handler program follow these steps:

- 1. Change to Y the *Call user defined prompt key/handler option* in Work with Execution and Security Settings.
- 2. Enter the name of the user defined prompt key message handler program into the *Name of user defined prompt key handler* in Work with Execution and Security Settings.
- **3**. The user defined prompt key message handler program must be set up to receive 1 parameter which is as follows:

Parameter Type Length Description Number

1 A 10 Program name - contains the name of the program requesting messages.

User Exit Programs

LANSA provides facilities for "user exit" programs at various key points within the product.

The following information describes the various user exit facilities that are available, and examples of how you might choose to use them.

Before attempting to enable and use a user exit program note the following:

If the User Exit Program uses SQL CLI

Disclaimer

User exits are provided to increase your productivity because they allow "tailor made" processing to be performed in some situations. However, if you enable a user exit program, and then use it, you are completely responsible for its operation, maintenance and future compatibility with any new releases of LANSA. You are also completely responsible for any loss of data integrity, or data corruption, that it may cause.

All examples in the following sections are provided on an "as is" basis. No warranty concerning these examples is expressed or implied. The suitability of these examples for your installation is a matter for your judgment alone.

Libraries in which user exit programs are placed

Do not place user exit programs in the standard LANSA program library. This library is cleared and reloaded during the install of a new LANSA release. Use another common, shared library such as QGPL instead.

User exit programs included:

User Exit F@BGNCMT - Start Commitment Control User Exit F@ENDCMT - End Commitment Control User Exit UEX@0001 - an Alternative RDML Command Editor User Exit UEX@0002 - an Alternative Help Text Editor User Exit LCXP9000 - User Exit for PC-Based Products User Exit - Export/Import Message logger User Exit CIDT (Can I Do This) User Exit IJDT (I Just Did This) User Exit LANSA Security Check User Exit for Environment User Exit for SUBMIT command

If the User Exit Program uses SQL CLI

As some parts of LANSA use SQL CLI, and there can only be one connection to the IBM i database at a time, LANSA provides a mechanism to obtain its connection handle.

This is by calling the program named DCXP8210 with these parameters:

Description	Туре	Length	Usage
Action	*CHAR	3	Use value 'GET'
Handle	*INT	4	If handle is returned as hex '00000000' then LANSA doesn't currently have a connection.

User Exit F@BGNCMT - Start Commitment Control

LANSA starts commitment control automatically under certain circumstances. Refer to Commitment Control for details on when this occurs.

The name of the program used to start commitment control is F@BGNCMT. Its source can be found in DC@F28. You may create your own version of F@BGNCMT based on this source.

When it is called it is passed the following parameters:

Parm Type/Len/Dec Description

1	A(1024)	Standard LANSA information data structure
2	A(2000)	Standard LANSA exchange data structure
3	A(2000)	Standard process information data structure

User Exit F@ENDCMT - End Commitment Control

LANSA ends commitment control automatically under certain circumstances. Refer to Commitment Control for details about when this occurs.

The name of the program used to end commitment control is F@ENDCMT. Its source can be found in DC@F28. You may create your own version of F@ENDCMT based on this source.

When it is called it is passed the following parameters.

Parm Type/Len/Dec Description

1	A(1024)	Standard LANSA information data structure
2	A(2000)	Standard LANSA exchange data structure
3	A(2000)	Standard process information data structure

User Exit UEX@0001 - an Alternative RDML Command Editor

A user exit program called UEX@0001 is provided to allow an alternative means of editing RDML commands.

If the option to use this alternative editor is enabled in the system definition data area DC@A01, and the user indicates that it should be used by specifying USR, USER or USREDT in the editor field while reviewing RDML commands, all edit requests will be directed to user exit program UEX@0001, instead of the standard LANSA editor, or the IBM supplied EDTSRC/SEU editor.

Set up this option Refer to Enable alternate editor user exits in Compile and Edit Settings.

Type/Len/Dec Description Parm 1 A(1024) Standard LANSA information data structure 2 A(2048) Standard LANSA exchange data structure 3 A(1) Error return area. Passed in as "N" (no errors). Return as "Y" (error occurred) if a fatal error occurs in the exit program. Update flag. Passed in as "N" (no updates). 4 A(1) Return as "Y" (updates performed) if LANSA is to update the details from the returned data. Name of work file being used by LANSA. The 5 A(10) work file is a temporary unkeyed data file with a record length of 80 bytes. 6 A(10) Library in which work file resides. Total number of work members in the file. 7 P(7,0) Array of 11 * A(10) process names. 8 A(110) Array of 11 * A(7) function names. 9 A(77) Array of 11 * A(10) work file member names. 10 A(110)

When UEX@0001 is called, it is passed the following parameters by the LANSA calling program DC@P3098:

There is always one process/function/member specified in the arrays passed to UEX@0001. There may be at most 11. The first one passed is the process/function/member that is being edited. The other 10, if present, are additional processes/functions/members that the user has indicated he/she may wish to browse/copy while editing the first member.

An example of how program UEX@0001 could be used follows. This CL (control language) program is used in conjunction with IBM's PC support to allow a function's RDML commands to be edited on a PC using any normal PC based source editor.

CL Program UEX@0001 - Edit RDML Commands on PC Under PC Support

UEX@0001: PGM (&DC@IDS &DC@EDS &ERROR &UPDATED &FILE &NM &PROLIST &FUNLIST &MBRLIST)

/* Declare all parameters and work variables (not described) */

/* Create a shared folder for this IBM i job */

RTVJOBA JOB(&JOB) CHGVAR &FOLDER (&JOB *TCAT '.RML') CRTFLR FLR(&FOLDER) TEXT('Temporary RDML work folder only') MONMSG (CPF0000 MCH0000)

/* Copy all members from LANSA work file into files in folder */

CHGVAR &I 1 CHGVAR &J 1 CHGVAR &K 1 LAB010: IF (&I *GT &NM) GOTO(LAB020) CHGVAR &FMEMBER %SST(&MBRLIST &J 10) CHGVAR &TMEMBER %SST(&FUNLIST &K 7) CPYTOPCD FROMFILE(&LIB/&FILE) TOFLR(&FOLDER) FROMMBR(&FMEMBER) TODOC(&TMEMBER) REPLACE(* CHGVAR &I (&I + 1) CHGVAR &J (&J + 10) CHGVAR &K (&K + 7) GOTO LAB010 /* Set up PC command details and execute PC editor .BAT file */

LAB020: CHGVAR &FMEMBER %SST(&MBRLIST 1 10) CHGVAR &TMEMBER %SST(&FUNLIST 1 7) CHGVAR &PCCMD ('MYEDIT' *BCAT &FOLDER *BCAT &TMEM STRPCCMD PCCMD(&PCCMD) PAUSE(*NO)

/* At end of edit, copy back into LANSA work member number 1 */ /* and then set the updated and error flags before ending */

CPYFRMPCD FROMFLR(&FOLDER) TOFILE(&LIB/&FILE) FROMDOC(&TMEMBER) TOMBR(&FMEMBER) MBROPT(*REPLACE)

CHGVAR &UPDATED 'Y' CHGVAR &ERROR 'N'

ENDPGM

Example

PC/DOS MYEDIT.BAT Batch Command File Used on the PC

These commands	do this:
FSPC ASSIGN J: %1	Assign shared folder to drive J
ED J:%2	Execute your favourite PC editor over the file containing the help text.
FSPC RELEASE J:	Release the folder from drive J
User Exit UEX@0002 - an Alternative Help Text Editor

User exit program UEX@0002 is provided to allow an alternative means of editing field, process or function help text.

If the option to use this alternative editor is enabled in the system definition data area DC@A01, and the user indicates that it should be used by selecting it in the check box that appears when beginning the help text edit, all edit requests will be directed to user exit program UEX@0002 instead of the standard IBM supplied EDTSRC/SEU editor.

Set up this option Enable alternate editor user exits in Compile and Edit Settings.

When UEX@0002 is called, it is passed the following parameters by the LANSA calling program DC@P1002:

Parm	Type/Length/Dec	Description
1	A(1024)	Standard LANSA information data structure
2	A(2048)	Standard LANSA exchange data structure
3	A(1)	Error return area. Passed in as "N" (no errors). Return as "Y" (error occurred) if a fatal error occurs in the exit program.
4	A(1)	Update flag. Passed in as "N" (no updates). Return as "Y" (updates performed) if LANSA is to update the details from the returned data.
5	A(3)	Type of help text being edited. "FLD" indicates field level help text. "PRO" indicates process level help text. "FUN" indicates function level help text.
6	A(10)	Name of field or process involved. Passed for "FLD", "PRO" and "FUN" edits.
7	A(7)	Name of function involved. Passed for "FUN" edits. Passed as blanks for others.
8	A(10)	Name of work file being used by LANSA. The work file is a temporary unkeyed data file

		with a record length of 77 bytes.
9	A(10)	Library in which work file resides.
10	A(10)	Name of member containing the help text.

An example of how program UEX@0002 could be used follows. This CL (control language) program is used in conjunction with IBM's PC support to allow help text to be edited on a PC using any normal PC editor or word processor.

CL Program UEX@0002 - Edit Help Text on PC Under PC Support

UEX@0002: PGM (&DC@IDS &DC@EDS &ERROR &UPDATED &TYPE &FUNCTION &FILE &LIB &MEMBER)

/* Declare all parameters and work variables (not described) */

/* Create a shared folder for this IBM i job */

RTVJOBA JOB(&JOB) CHGVAR &FOLDER (&JOB *TCAT '.HLP') CRTFLR FLR(&FOLDER) TEXT('Temp help work folder only') MONMSG (CPF0000 MCH0000)

/* Copy details into the file in the shared folder */

CPYTOPCD FROMFILE(&LIB/&FILE) TOFLR(&FOLDER) FROMMBR(&MEMBER) TODOC(&MEMBER) REPLACE(*YES)

/* Set up PC command details and execute PC editor .BAT file */

CHGVAR &PCCMD ('MYEDIT' *BCAT &FOLDER *BCAT &MEMBE STRPCCMD PCCMD(&PCCMD) PAUSE(*NO)

/* At end of edit, copy back into LANSA work member and */ /* then set the updated and error flags before ending */

CPYFRMPCD FROMFLR(&FOLDER) TOFILE(&LIB/&FILE) FROMDOC(&MEMBER) TOMBR(&MEMBER) MBROPT(*REPL

CHGVAR &UPDATED 'Y' CHGVAR &ERROR 'N'

ENDPGM

Example

PC/DOS MYEDIT.BAT Batch Command File Used on the PC

These commands:	do this:
FSPC ASSIGN J: %1	Assign shared folder to drive J
ED J:%2	Execute your favourite PC editor over the file containing the help text.
FSPC RELEASE J:	Release the folder from drive J

User Exit LCXP9000 - User Exit for PC-Based Products

User exit program LCXP9000 is called (when present in library QGPL) on startup of server-side jobs for LANSA SuperServer, LANSA Open, and LANSA Open .Net.

There are no parameters to LCXP9000.

User Exit - Export/Import Message logger

A user-defined program can change the message type parameter to another valid message type.

The available values for Message type are:

Message Type	Description
С	Completed
W	Warning
Е	Error
F	Fatal

Message type E is similar to the existing warning code W. Code E is not used by LANSA but a user exit program can change the LANSA message type W or C to Code E, or Codes C to W or W to C. The LANSA message type F cannot be changed, nor can another code type be changed to F.

Changing the Message types will not influence the processing of the import/export but it will influence the report and the final totals for each message type.

If this user exit is used, the report will print the category (E) with this text: nnnn - ERROR MESSAGES ISSUED DURING THIS IMPORT RUN.

The data area DC@RESLT, in positions 32-38, will be updated with the total of the Code Es encountered. If the user exit is not used, the value in this field will be set to zero. You can set positions 32-38 via the option User defined Export/Import message logging program in *Export and Import settings*.

If the return code from the user-defined program is not one of the valid codes, the Export/Import message logger (DC@P6203) will ignore it and use its original return code.

If the Partition ID is required, it can be derived from DC@IDS parameter position 910 - 912.

Export/Import message logger will pass these parameters to this user exit

program:

Description	Туре	Length	Usage
DC@IDS	*CHAR	1024	Cannot be changed.
Message Type	*CHAR	1	Cannot be changed to F.
Message ID	*CHAR	7	Cannot be changed.
Message Text	*CHAR	132	Cannot be changed.

Refer also to Data Area DC@A07 positions 649-668.

User Exit CIDT (Can I Do This)

When the CIDT user exit is called and how

A user-defined program can manipulate the two Return Code parameters from the LANSA CIDT (Can I Do This) user exit. The *User Exit Return Code* parameter will determine whether to continue with the LANSA CIDT (DC@P8096) processing or not.

The user-defined program will be executed before the CIDT check, regardless of whether Task control is active or not. (The user-defined program can determine if task control is active by checking the contents of parameter DC@EDS. If 'Y' is found in DC@EDS, parameter position 1517 - 1517, then task control is active for the partition.)

If the user exit program parameter *User Exit Return Code* returns a value of:

- C, the LANSA CIDT will continue processing and will override the user exit's *Return Code* with its own code, as required.
- N, the LANSA CIDT will not continue processing and will use its own *Return Code* value.

If the user exit program returns a *Return Code* value that is not Y or N, the LANSA CIDT will process as for a *User Exit Return Code* of C.

If the Partition ID is required, it can be derived from DC@IDS parameter position 910 - 912.

Description	Туре	Length	Usage
DC@IDS	*CHAR	1024	Cannot be changed.
DC@EDS	*CHAR	2500	Cannot be changed.
Object Name	*CHAR	10	Cannot be changed.
Object Extension	*CHAR	10	Cannot be changed.
Object type	*CHAR	2	Cannot be changed.
Return Code	*CHAR	1	Can be changed. Valid values are Y or N.
User Exit Return Code	*CHAR	1	Will accept C or N
Validate User for	*CHAR	4	Passed as 'CIDT' or 'VTSK' *

Task Tracking

PC Name *CHAR 15

Cannot be changed. Will only be filled if a result of Visual LANSA development.

Refer also to User defined CIDT program (positions 709-728) in the Execution and Security Settings.

* This value indicates from where the user exit has been called.

'CIDT' = called from the CIDT (Can I Do This) routine. The Object Name, Extension and Type will contain the details of the object being checked.

'VTSK' = called when checking the Task ID is valid and the user is allowed to use this task. This call occurs if a Task ID is required when importing. The Object Name, Extension and Type will be blank. The 'VTSK' call happens if a Task ID is required when importing and is actioned when an import is requested or an import is executed.

When the CIDT user exit is called and how

CIDT user exit for Fields, components & WAMs (object type DF) CIDT user exit for System Variables (object type SV) CIDT user exit for Multilingual Variables (object type MT) CIDT user exit for Files (object type FD) CIDT user exit for Processes (object type PD) CIDT user exit for Functions (object type PF) CIDT user exit for Templates (object type AT) CIDT user exit for Web Components (object type WC) CIDT user exit for XML Components (object type XC) CIDT user exit for Technology Services (object type AA) CIDT user exit for Weblets (object type WL)

CIDT user exit for Fields, components & WAMs (object type DF)

User Action	Comments
Create a field, copy help	
Create a field, copy all rules/triggers	
Create a field, selectively copy rules/triggers	
Create a field, copy multilingual definitions	
Load Other file	When creating a new field
Review/Change a field	
Maintain field help text	
Maintain field multilingual attributes	
Maintain field rules/triggers	
Import	When task tracking for imports is on
Delete	
Export to PC platform for maintenance	
Check out for update	Occurs twice
Check in	
Delete from host repository	
Use BIF PUT_FIELD	When field is changed, not created
Use BIF PUT_HELP	When field help

Use BIF PUT_FIELD_ML

Use BIF PUT_COND_CHECK, PUT_DATE_CHECK, PUT_PROGRAM_CHECK, PUT_RANGE_CHECK, PUT_VALUE_CHECK, PUT_TRIGGER

When field validations

Use BIF DELETE_CHECKS

When field validations

Use BIF DLT_FIELD

Whenever IJDT is called

For a field, component or WAM

CIDT user exit for System Variables (object type SV)

User Action	Comments
Review/Change	
Import	
Delete	
Export to PC platform for maintenance	
Check out for update	Occurs twice
Check in	
Delete from host repository	
Use BIF PUT_SYSTEM_VARIABLE	When system variable is changed, not created
Web administrator maintain system variable	
Whenever IJDT is called	For a system variable

CIDT user exit for Multilingual Variables (object type MT)

User Action	Comments
Review/Change	
Import	
Delete	
Export to PC platform for maintenance	
Check out for update	Occurs twice
Check in	
Delete from host repository	
Use BIF PUT_ML_VARIABLE	When multilingual variable is changed, not created
Whenever IJDT is called	For a multilingual variable

CIDT user exit for Files (object type FD)

User Action	Comments
Maintain	
Load Other file	
Request to make operational	
Make operational (separate batch job)	
Import	
Request to delete	
Delete (separate batch job)	
Export to PC platform for maintenance	
Check out for update	Occurs twice
Check in	
Check in request compile	Occurs twice
Request delete from host repository	Occurs twice
Use BIF START_FILE_EDIT	When file is changed, not created
Use BIF PUT_COND_CHECK, PUT_DATE_CHECK, PUT_PROGRAM_CHECK, PUT_RANGE_CHECK, PUT_VALUE_CHECK, PUT_TRIGGER	When file validations
Use BIF DELETE_CHECKS	When file validations
Use BIF MAKE_FILE_OPERATIONL	
Use BIF DLT_FILE	
Whenever IJDT is called	For a file

CIDT user exit for Processes (object type PD)

User Action	Comments
Maintain	
Maintain process help text	
Create a function	The process it is created in
Request to compile	If process itself is to be compiled
Compile (separate batch job)	If process itself is to be compiled
Import	
Request to delete	
Delete (separate batch job)	
Export to PC platform for maintenance	
Check out for update	Occurs twice
Check in	
Check in request compile	Occurs twice
Request delete from host repository	Occurs twice
Use BIF START_PROCESS_EDIT	When process is changed, not created
Use BIF START_FUNCTION_EDIT	When function is created
Use BIF PUT_HELP	When process help
Use BIF COMPILE_PROCESS	
Use BIF DELETE_PROCESS	
Whenever IJDT is called	For a process

CIDT user exit for Functions (object type PF)

User Action	Comments
Review/Change RDML	
Maintain function help text	
Rename	The old function name
Request to compile a process	For each function selected to compile
Compile a process (separate batch job)	For each function selected to compile
Import	
Delete	
Request to delete a process	For each function in the process
Export to PC platform for maintenance	
Check out for update	Occurs twice
Check in	
Check in request compile a process	For each function selected to compile
Delete from host repository	Occurs twice
Use BIF START_FUNCTION_EDIT	When function is changed, not created
Use BIF PUT_HELP	When function help
Use BIF COMPILE_PROCESS	For each function requested to compile
Use BIF DELETE_FUNCTION	
Use BIF DELETE_PROCESS	For each function in the process
Whenever IJDT is called	For a function

CIDT user exit for Templates (object type AT)

User Action	Comments
Review/Change definition	
Review/Change commands	
Review/Change Help panels	
Import	
Delete	
Export to PC platform for maintenance	
Check out for update	Occurs twice
Check in	
Delete from host repository	
Whenever IJDT is called	For a template

CIDT user exit for Web Components (object type WC)

User ActionCommentsImportExport to PC platform for maintenanceCheck out for update

Whenever IJDT is called

For a web component

CIDT user exit for XML Components (object type XC)

User Action

Comments

Import

Export to PC platform for maintenance

Check out for update

Whenever IJDT is called

For a XML component

CIDT user exit for Technology Services (object type AA)

User Action	Comments
Import	
Export to PC platform for maintenance	
Check out for update	
Check in	
Delete from host repository	
Whenever IJDT is called	For a technology service

CIDT user exit for Weblets (object type WL)

User Action	Comments
Import	
Export to PC platform for maintenance	
Check out for update	
Check in	
Delete from host repository	
Whenever IJDT is called	For a weblet

To validate task for import (object type blank)

User Action Comments

Partition	When task tracking for imports is on, to validate the task.
Initialize	Object name, extension and type are blank, and the 8th
Request	parameter is VTSK, not CIDT
import	
Import	

User Exit IJDT (I Just Did This)

A user-defined program can manipulate the processing of the LANSA IJDT (I Just Did This) and will return two parameters that will influence the processing. The *User Exit Return Code* parameter will determine whether to continue with the LANSA IJDT (DC@P8097) processing or not.

The user-defined program will be executed before the IJDT check, regardless of whether Task control is active or not. (The user-defined program can determine if task control is active by the contents of parameter DC@EDS. If 'Y' is found in DC@EDS parameter position 1517 - 1517, then task control is active for the partition.)

If the user exit's program parameter *User Exit Return Code* returns a value of:

- C, the LANSA IJDT will continue processing and will override the user exit's *Return Code* with its own value, as required.
- N, the LANSA IJDT will not continue processing and will return the *Return Code* value that was passed by the user exit program.

If the user exit returns a *Return Code* value that is not Y or N, the LANSA IJDT will process as for a *User Exit Return Code* of C.

If the Partition ID is required, it can be derived from DC@IDS parameter position 910 - 912.

Description	Туре	Length	Usage
DC@IDS	*CHAR	1024	Cannot be changed
DC@EDS	*CHAR	2500	Cannot be changed
Object Name	*CHAR	10	Cannot be changed
Object Extension	*CHAR	10	Cannot be changed
Object Type	*CHAR	2	Cannot be changed
Event Code	*CHAR	3	Cannot be changed
Message ID	*CHAR	7	Cannot be changed
Message File Name	*CHAR	10	Cannot be changed
Message Data	*CHAR	132	Cannot be changed
Return Code	*CHAR	1	Can be changed. Valid values are Y

		or N.
User Exit Return Code	*CHAR 1	Will accept C or N.
Event Source	*CHAR 10	Cannot be changed.
Event LockId	*CHAR 1	Cannot be changed.
Event SubType	*CHAR 3	Cannot be changed.

Refer also to Data Area DC@A07 positions 689-708.

When the IJDT user exit is called and how

IJDT user exit for Fields, components & WAMs (object type DF)
IJDT user exit for System Variables (object type SV)
IJDT user exit for Multilingual Variables (object type MT)
IJDT user exit for Files (object type FD)
IJDT user exit for Processes (object type PD)
IJDT user exit for Functions (object type PF)
IJDT user exit for Templates (object type AT)
IJDT user exit for XML Components (object type XC)
IJDT user exit for Technology Services (object type AA)
IJDT user exit for Weblets (object type WL)
IJDT user exit for Validating Task (object type \$\$)

IJDT user exit for Fields, components & WAMs (object type DF)

Action	Evt code	Sub Evt Code	Message Id	Comments
Create a field - copy help - copy all rules/triggers - selectively copy rules/triggers - copy multilingual definitions Load Other file	CRT CHG CHG CHG CHG CRT		DCM0015 DCM0989 DCM0990 DCM0991 DCM0992 DCM0015	When creating a new field
Maintain a field Maintain field help text Maintain field multilingual attributes	CHG CHG CHG		DCM0014 DCM0930 DCM0872	
Create / Maintain field rules/triggers	CHG		DCM1005 DCM1007 DCM1009 DCM1011 DCM1013 DCM1015 DCM0191	Range check Value check Table check Simple logic check Complex logic check Date check
Delete field rules/triggers	CHG		DCM1003	
Import	IMP		EIM0051	

Delete	DLT	DCM0012	
Export to PC platform for maintenance	CHG	EIM0110	
Check out for update	CHG CHG	EIM0110 L2M0010	
Check in	CHG CHI	L2M0003	
Delete from host repository	DLT	L2M0022	
Use BIF PUT_FIELD	CRT CHG	DCM0015 DCM0014	If field created If field changed
Use BIF PUT_HELP	CHG	DCM0930	When field help
Use BIF PUT_FIELD_ML	CHG	DCM0014	
Use BIF PUT_COND_CHECK, PUT_DATE_CHECK, PUT_PROGRAM_CHECK, PUT_RANGE_CHECK, PUT_VALUE_CHECK, PUT_TRIGGER	CHG	DCM1005	When field validations
Use BIF DELETE_CHECKS	CHG	DCM1003	When field validations
Use BIF DLT_FIELD	DLT	DCM0012	

IJDT user exit for System Variables (object type SV)

Action	Evt code	Sub Evt Code	Message Id	Comments
Create	CRT		DCM0041	
Maintain	CHG		DCM0040	
Import	IMP		EIM0064	
Delete	DLT		DCM0042	
Export to PC platform for maintenance	CHG		EIM0110	
Check out for update	CHG		EIM0110	
	CHG		L2M0010	
Check in	CHG	CHI	L2M0003	
Delete from host repository	DLT		L2M0022	
Use BIF PUT_SYSTEM_VARIABLE	CRT CHG		DCM0041 DCM0040	If system variable created If system variable changed
Web administrator maintain	CRT CHG DLT		DCM0041 DCM0040 DCM1611	If system variable created If system variable changed If system variable deleted

IJDT user exit for Multilingual Variables (object type MT)

Action	Evt code	Sub Evt Code	Message Id	Comments
Create	CRT		DCM1267	
Maintain	CHG		DCM1266	
Import	IMP		EIM0093	
Delete	DLT		DCM1268	
Export to PC platform for maintenance	CHG		EIM0110	
Check out for update	CHG CHG		EIM0110 L2M0010	
Check in	CHG	CHI	L2M0003	
Delete from host repository	DLT		L2M0022	
Use BIF PUT_ML_VARIABLE	CRT CHG		DCM1267 DCM1266	If multilingual variable created If multilingual variable changed

IJDT user exit for Files (object type FD)

Action	Evt code	Sub Evt Code	Message Id	Comments
Create	CRT		DCM0091	
Maintain	CRT		DCM0597	
Create / Maintain file rules/triggers	CHG		DCM1006 DCM1008 DCM1010 DCM1012 DCM1014 DCM1016 DCM0190	Range check Value check Table check Simple logic check Complex logic check Date check Trigger
Delete file rules/triggers	CHG		DCM1004	
Request to make operational	CRT		DCM0931	
Make operational (separate batch job)	CRT		DCM0269	
Import	IMP		EIM0056	
Request to delete	DLT		DCM0932	
Delete (separate batch job)	DLT		DCM0259	
Export to PC platform for maintenance	CHG		EIM0110	
Check out for update	CHG CHG		EIM0110 L2M0010	
Check in	CHG	CHI	L2M0003	

CRT CMP CRT CMP CHG CMP	DCM0931 DCM0269 L2M0200	
DLT DLT DLT	DCM0932 DCM0259 L2M0204	
CRT	DCM0091	If file created
CHG	DCM0597	If commit requested
CHG	DCM1006	When file validations
CHG	DCM1004	When file validations
CRT	DCM0931	
DLT	DCM0932	
	CRT CMP CHG CMP DLT CMP DLT CMP CRT CMP CHG CMP CHG CMP CHG CMP	CRT CRT CHGCMP CMP CMP CMP DCM0269 DCM0259 DCM0259 DCM0259 DCM0259 DCM0259DLT DLTIDCM0932 DCM0259CRTIDCM0932 DCM0597CHGIDCM0932 DCM1006CHGIDCM0932 DCM1004CHGIDCM0931 DCM0931CHGIDCM0931 DCM0931

IJDT user exit for Processes (object type PD)

Action	Evt code	Sub Evt Code	Message Id	Comments
Create	CRT		DCM0323	
Maintain process help text	CHG		DCM0930	
Maintain process multilingual attributes	CHG		DCM0941	
Maintain function control table	CHG		DCM0935	
Maintain special entries	CHG		DCM0936	
Maintain multilingual special entries	CHG		DCM0937	
Maintain other processes	CHG		DCM0938	
Maintain parameters	CHG		DCM0939	
Maintain miscellaneous details	CHG		DCM0940	
Maintain action bar control table	CHG		DCM0942	
Request to compile	CRT		DCM0943	If process itself is to be compiled
Compile (separate batch job)	CRT		PRC0034 PRC0035	If process compiled successfully
				If process itself failed to compile
Import	IMP		EIM0063	
Request to delete	DLT		DCM0944	
Delete (separate batch job)	DLT		PRC0076	
Export to PC platform for	CHG		EIM0110	

maintenance			
Check out for update	CHG	EIM0110	
	CHG	L2M0010	
Check in	CHG CHI	L2M0003	
Check in compile a process	CRTCMPCRTCMPCRTCMPCHGCMP	DCM0943 PRC0034 PRC0035 L2M0202	If process itself to be compiled If process compiled successfully If process itself failed to compile
Delete from host repository	DLT DLT DLT	DCM0904 PRC0076 L2M0206	
Use BIF START_PROCESS_EDIT	CRT	DCM0323	If process created
Use BIF PUT_HELP	CHG	DCM0930	When process help
Use BIF PUT_PROCESS_ML, PUT_FUNCTION_ML	CHG	DCM0941	
Use BIF PUT_PROCESS_ACTIONS	CHG	DCM0942	
Use BIF PUT_PROCESS_ATTACH, DELETE_PROCESS_ATTACH	CHG	DCM0938	
Use BIF PUT_PROCESS_ATTR	CHG	DCM0940	
Use BIF PUT_FUNCTION_ATTR	CHG	DCM0935	

Use BIF COMPILE_PROCESS	CRT	DCM0943	If process itself to be compiled
Use BIF DELETE_PROCESS	DLT	DCM0944	

IJDT user exit for Functions (object type PF)

Action	Evt code	Sub Evt Code	Message Id	Comments
Create	CRT		DCM0933	
Rename	CHG CRT		DCM1324 DCM1368	Old function name New function name
Maintain function help text	CHG		DCM0930	
Maintain function RDML via SEU etc	CHG		DCM0934	
Maintain function RDML via RDML Editor	CHG		DCM0691	
Save RDML from report painter	CHG		DCM0691	
Save RDML from screen painter	CHG		DCM0691	
Request to compile a process	CRT		DCM0945	For each function requested to compile
Compile a process (separate batch job)	CRT		PRC0183 PRC0184	For each function compiled successfully
				For each function failing to compile
Import	IMP		EIM0062	
Delete	DLT		DCM0517	
Export to PC platform for maintenance	CHG		EIM0110	
Check out for update	CHG		EIM0110	

	CHG		L2M0010	
Check in	CHG	CHI	L2M0003	
Check in compile a process	CRT CRT CRT	CMP CMP CMP	DCM0945 PRC0183 PRC0184	For each function requested For each function compiled successfully For each function failing to compile
Delete from host repository	DLT DLT		DCM0518 L2M0208	
Use BIF START_FUNCTION_EDIT	CRT		DCM0933	If function created
Use BIF PUT_HELP	CHG		DCM0930	When function help
Use BIF PUT_FUNCTION_RDML	CHG		DCM0691	
Use BIF COMPILE_PROCESS	CRT		DCM0945	For each function requested to compile
Use BIF DELETE_FUNCTION	DLT		DCM0517	
IJDT user exit for Templates (object type AT)

Action	Evt code	Sub Evt Code	Message Id	Comments
Create	CRT		DCM0807	
Maintain definition	CHG		DCM0808	
Maintain commands	CHG		DCM0960	
Create help panel	CRT		DCM0821	
Maintain help panel	CHG		DCM0824	
Delete help panel	DLT		DCM0825	
Import	IMP		EIM0087	
Delete	DLT		DCM0809	
Export to PC platform for maintenance	CHG		EIM0110	
Check out for update	CHG		EIM0110	
	CHG		L2M0010	
Check in	CHG	CHI	L2M0003	
Delete from host repository	DLT		L2M0022	

IJDT user exit for Web Components (object type WC)

Action	Evt code	Sub Evt Code	Message Id	Comments
Import	IMP		EIM0236	
Export to PC platform for maintenance	CHG		EIM0110	
Check out for update	CHG		EIM0110	
Use BIF WEB_BUILD_COMPONENT	CHG		DCM1609	If web component
Web administrator maintain	CRT CHG DLT		DCM1610 DCM1609 DCM1611	If web component created If web component changed If web component deleted

IJDT user exit for XML Components (object type XC)

Action	Evt code	Sub Evt Code	Message Id	Comments
Import	IMP		EIM0236	
Export to PC platform for maintenance	CHG		EIM0110	
Check out for update	CHG		EIM0110	
Use BIF WEB_BUILD_COMPONENT	CHG		DCM1609	If XML component
Web administrator maintain	CRT CHG DLT		DCM1610 DCM1609 DCM1611	If XML component created If XML component changed If XML component deleted

IJDT user exit for Technology Services (object type AA)

Action	Evt code	Sub Evt Code	Message Id	Comments
Import	IMP		EIM0056	
Export to PC platform for maintenance	CHG		EIM0110	
Check out for update	CHG CHG		EIM0110 L2M0010	
Check in	CHG	CHI	L2M0003	
Delete from host repository	DLT		L2M0022	

IJDT user exit for Weblets (object type WL)

Action	Evt code	Sub Evt Code	Message Id	Comments
Import	IMP		EIM0056	
Export to PC platform for maintenance	CHG		EIM0110	
Check out for update	CHG CHG		EIM0110 L2M0010	
Check in	CHG	CHI	L2M0003	
Delete from host repository	DLT		L2M0022	

IJDT user exit for Validating Task (object type \$\$)

Action	Evt code	Sub Evt Code	Message Id	Comments
Change task	CRT			Only from interactive job

User Exit LANSA Security Check

A user-defined program can manipulate the processing of the LANSA access check for an object and can return two parameters that will influence further processing. The *User Exit Return Code* parameter will determine whether to continue with the LANSA access check for an object (DC@P8001) processing or not.

If the user-defined program parameter *User Exit Return Code* returns a value of:

- C, the LANSA access check for an object routine (DC@P8001) will continue processing and, if required, will override with its own value the *Return Code* value that was passed by the user exit.
- N, the LANSA access check for an object routine (DC@P8001) will not continue processing, and will return the *Return Code* that was passed by the user exit program

The valid values for *Return Code* are N or Y

If the user exit returns a Return Code value that is not Y or N, the LANSA access check for an object (DC@P8001) routine will ignore it and process as for *User Exit Return Code* value C.

If the Partition ID is required, it can be derived from DC@IDS parameter position 910 - 912.

The LANSA Security Check programs will pass these parameters to the user exit program:

Description	Туре	Length	Usage
DC@IDS	*CHAR	1024	Cannot be changed.
Object Name	*CHAR	10	Cannot be changed.
Object Extension	*CHAR	10	Cannot be changed.
Object type	*CHAR	2	Cannot be changed.
Access	*CHAR	2	Cannot be changed.
Return Code	*CHAR	1	Can be changed. Valid values are Y or N
User exit return code	*CHAR	1	Will be accepted C or N

Refer also to Data Area DC@A07 positions 669-688.

User Exit for Environment

The Environment User Exit program name is stored in positions 611-620 of Data Area DC@A07.

After LANSA has set up its environment for the partition bytes 611-620 of DC@A07 are tested. If non-blank, they are assumed to identify a user exit program that should be invoked each time that a LANSA partition is entered.

Description	Туре	Length
LANSA program library	CHAR	10
Partition ID	CHAR	3
Language Code	CHAR	4

User Exit for SUBMIT command

The User Exit program name for the SUBMIT command is stored in positions 621-630 of Data Area DC@A07.

These bytes are tested the first time a LANSA function is submitted. If nonblank, they are assumed to identify a user exit program that should be invoked each time a function is submitted. Such a program must receive one parameter containing the SBMJOB command about to be used. The user exit program can modify the SBMJOB command before exiting and passing back the parameter. (e.g. Add MSGQ() parameter to SBMJOB).

Description Type Length

User Exit program parameter CHAR 2000

The Permanent File Overrides Facility

LANSA's Permanent File Overrides facility allows you to specify that when a specific file is used, LANSA is to always use a different one. This override facility can be used for files with 10 character file names or files with a "." in their name as shown in the Examples.

The Permanent File Overrides facility is activated by specifying Y in the *Allow permanent file overrides* option of the System Settings Work with Execution and Security Settings. (This option inserts the string "*PERMFILOVR" into LANSA data area DC@OSVEROP.)

First you need to create a source physical file called PERMFILOVR using this command:.

CRTSRCPF PERMFILOVR

You can create this anywhere. LANSA looks for this file in the *LIBL at runtime. If the file is found, LANSA reads the first member to find file override information. To make this override information partition specific, put this file into the partition data or module libraries.

You can edit file PERMFILOVR using SEU (STRSEU) or edit file (EDTF).

- Records in the first member must be either comments or file override details.
- A semi-colon (;) in the first position indicates a comment line.
- Any line that doesn't start with a semi-colon is considered to contain file override details.
- There must be a space between each name.
- All file and library names must be in upper case.

Comments can be used to indicate the layout as shown in this example:

;From File To File To Library ;234567890 1234567890 1234567890 CUSTMAST CUSTMASTER MYLIBRARY

Notice the space left between each name.

You can specify two files on one line:

;From File To File To Library From File To File To Library ;234567890 1234567890 1234567890 1234567890 1234567890 1234567890 ITEMMAST ITEMMASTER DATALIB ORDERH ORDER.HEAD DAT You can have just one file on each line with a comment on the right hand side. Use a semi-colon to indicate the start of the comment, as in this example:

;From File To File To Library Comment ;234567890 1234567890 1234567890 1 ITEMMAST ITEMMASTER DATALIB ; Item master file CUSTMAST CUSTMASTER MYLIBRARY ; Customer Master

If you do not want the library name to be overridden, leave it blank. You can also use the special values *LIBL and *CURLIB as this example:

;From File To File To Library Comment ;234567890 1234567890 1 ITEMMAST ITEMMASTER ; Item master CUSTMAST CUSTMASTER *CURLIB ; Customer master ORDERH ORDER.HEAD DATALIB ; Order Header ORDERD ORDER.DET *LIBL ; Order Detail

You must specify logical views as well. LANSA will not override the logical views unless you tell it to as shown here:

;From File To	File To	Library Comn	nent
;234567890 1	23456789	0 1234567890) 1
ITEMMAST	ITEMM	ASTER *LIBL	L ; Item master
ITEMMAL1	ITEMM	ASTL1 *LIBL	; Item master by Item
ITEMMAL2	ITEMM	ASTL2 *LIBL	; Item master by Class
ITEMMAL3	ITEMM	ASTL3 *LIBL	; Item master by Catagory
ITEMMAL4	ITEMM	ASTL4 *LIBL	; Item master by Status

Comments and Warnings

- The maximum number of file overrides is 2000. If you specify more than 2000 overrides then those over 2000 will be ignored.
- LANSA will not check the file overrides you specify. If you do not put the file overrides into the correct columns (From File, To File, etc) then you will get errors at run-time. Also, using invalid object names will cause errors at run-time.
- The first I/O module will call program M@PEROVR which reads file PERMFILOVR and loads the file override details into memory. As file

PERMFILOVR is only read once, if you change it, for your changes to take effect you must exit LANSA and RCLRSC or sign off and start LANSA again. The same applies to LANSA Client or LANSA Open, the sessions must be ended and started again if file PERMFILOVR is changed.

- As well as specifying Y for *Allow Permanent File* overrides, I/O modules must be rebuilt to enable use of the details in the PERMFILOVR file.
- For better memory (PAG) management you can call M@PEROVR before starting LANSA. This will load M@PEROVR early in the PAG so memory is potentially saved. Use this call to load M@PEROVR:

CALL M@PEROVR X'00'

The Hex 0 parameter tells M@PEROVR to do nothing but load and stay active in memory. For LANSA Open and LANSA Client, this call can be put into a program called LCXP9000. You create this program, which LANSA knows is a "User Exit" program, that must be called when LANSA Client and LANSA Open start. If LCXP9000 does not exist, LANSA traps the error on the call and LANSA Open or LANSA Client will continue normally.

Examples

Override files with a 10 character name (FILE10NAME)

Override files with a "." in the name (FIL.X)

Override files with a 10 character name (FILE10NAME)

- 1. Create a special library for dummy files. This library will just be used for file definitions (**no data**). In this example, the library is called DUMMYLIB.
- 2. Use CRTDUPOBJ to copy FILE10NAME to DUMMYLIB. Also copy the logicals attached to FILE10NAME into DUMMYLIB (FILE10NAM1 and FILE10NAM2). As you only need the file definitions, you do not need to copy the data.
- 3. Use DSPDBR to make sure that logicals FILE10NAM1 and FILE10NAM2 in DUMMYLIB are attached to file FILE10NAME in DUMMYLIB. This is important for when the file is loaded into LANSA. (Remember that your file & library names must be in upper case.)
- 4. Rename FILE10NAME in DUMMYLIB and its logicals so that the files can be loaded into LANSA.

In this example FILE10NAME becomes FILE10NA FILE10NAM1 becomes FILE10N1

FILE10NAM2 becomes FILE10N2

It is important to rename the files after they have been copied to DUMMYLIB. If the physical file does not have the same name as the one in the original library then the logicals will not attach to it when they are copied. They would remain attached to the original file.

- 5. The files can now be loaded into LANSA.
 - Define the files in LANSA as OTHER files.
 - Load their external definitions and make the files operational so that I/O modules are created.
 - Note that you must set the *Allow permanent file overrides* option to Y in the *Work with Execution and Security Settings* before the files are made operational.
- 6. Create/maintain file PERMFILOVR. The following entries would be inserted:

FILE10NA FILE10NAME *LIBL FILE10N1 FILE10NAM1 *LIBL FILE10N2 FILE10NAM2 *LIBL

A specific library could be specified instead of *LIBL.

7. When the files have been made operational, you can use them throughout LANSA. You use the file names that have been defined to LANSA. At runtime, LANSA will handle the overrides to the correct files because file PERMFILOVR is in the run-time library list (*LIBL).

Override files with a "." in the name (FIL.X)

- 1. Create a special library for dummy files. This library will just be used for file definitions (**no data**). In this example, the library is called DUMMYLIB.
- 2. Use CRTDUPOBJ to copy FIL.X to DUMMYLIB. Also copy the logical views attached to FIL.X into DUMMYLIB (FIL.X1 and FIL.X2). As you only need the file definitions, you do not need to copy the data.
- 3. Use DSPDBR to make sure that logicals FIL.X1 and FIL.X2 in DUMMYLIB are attached to file FIL.X in DUMMYLIB. This is important for when the file is loaded into LANSA.
- 4. Rename FIL.X in DUMMYLIB and its logicals so that the files can be loaded into LANSA. (Remember that your file & library names must be in

upper case.)

In this example FIL.X becomes FILEX FIL.X1 becomes FILEX1 FIL.X2 becomes FILEX2

It is important to rename the files after they have been copied to DUMMYLIB. If the physical file does not have the same name as the one in the original library then the logicals will not attach to it when they are copied. They would remain attached to the original file.

- 5. The files can now be loaded into LANSA.
 - Define the files in LANSA as OTHER files.
 - Load their external definitions and make the files operational so that I/O modules are created.
 - Note that you must set the *Allow permanent file overrides* option to Y in the *Work with Execution and Security Settings* before the files are made operational.
- 6. Create/maintain file PERMFILOVR. The following entries would be inserted...

FILEX FIL.X *LIBL FILEX1 FIL.X1 *LIBL FILEX2 FIL.X2 *LIBL

A specific library could be specified instead of *LIBL.

7. When the files have been made operational, you can use them throughout LANSA. You use the file names that have been defined to LANSA. At runtime, LANSA will handle the overrides to the correct files because you have file PERMFILOVR in the run-time library list (*LIBL).

Important note for the above examples

LANSA uses I/O module programs to access files. The I/O module programs must be in the *LIBL at run-time. There is one I/O module program for each physical file and it has the same object name as the physical file. By default, LANSA will put the I/O module into the same library as the file (DUMMYLIB). This will mean that DUMMYLIB will have to be in the *LIBL at run-time.

Some alternatives:

- Change LANSA's database file attributes for the file so that the I/O module will be put into the partition module library. This alternative is only valid if the partition's module library is not DUMMYLIB.
- Copy the I/O module to another library which will be in the*LIBL at runtime. Use this alternative only if you remember to do this every time you recompile the I/O module.

The @@UPID Field in LANSA Created Files

Whenever a physical file is created and maintained by LANSA (rather than some OTHER system) it has an additional field placed into it. The field is called @@UPID and is defined as packed (7,0). It is always the last field in the file.

Field @@UPID is used by LANSA to automatically check for "crossed updates". The logic to do this is very simple:

- Read the record and save the @@UPID value.
- If update required: re-read record and compare the @@UPID with saved @@UPID value. If different issue "crossed update" error message, else add 1 to @@UPID and update the file record.

When writing user application programs (in non-LANSA applications) to write new records or update existing records in database files created by LANSA the following is recommended:

1.Set @@UPID to 1 when writing new records.

2.Add 1 to @@UPID when updating an existing record.

This effectively emulates the logic automatically used in all LANSA functions.

Note: COBOL programs will not like the field name @@UPID.

To solve this problem, alter the data dictionary definition of field @@UPID so that it has an associated **alias name** acceptable to COBOL (unless this has already been done).

Force recreation of all database files (that do not already have the alias name included), and then in the COBOL programs, use the COPY DD option to ensure that where a field has an alias name, it is to be used in the program in preference to its real name.

Warning: The field @@UPID should not be used at 4GL level, except where you have received specific instructions from LANSA on how to use it.

Note: When a file contains BLOB or CLOB fields, @@UPID may be incremented multiple times for a single UPDATE command. This occurs once for the main file, and once for each BLOB or CLOB field included in the UPDATE command.

Also see

RESET_@@UPID Built-In Function in the *Technical Reference Guide*. Triggers - Some Do's and Don'ts.

Commitment Control

The IBM i operating system provides a facility called "commitment control" that allows programs to define database transaction boundaries.

When a database transaction is in progress, all changes to the database are considered to be "temporary" until the transaction boundary is reached. At the boundary, the program may choose to COMMIT the changes to the database, or ROLLBACK the changes to the last completed transaction boundary (thus removing them from the database).

If the job should end before reaching the transaction boundary, the operating system will automatically ROLLBACK to the last completed transaction boundary, thus removing any incomplete transactions from the database. For interactive jobs, the job begins when the user signs on and finishes with the job ending abnormally, or the user signing off.

It is possible to define and use database files within LANSA that are under IBM i commitment control. However, before doing this you should consider the following points:

- **Commitment control is a native part of the operating system** You should read all available IBM supplied documentation before attempting to use commitment control in LANSA. There is nothing special about the LANSA implementation of commitment control, it just uses the standard facilities provided by the operating system. Generally, when problems occur they can be traced to non-compliance with the operating system requirements, rather than to LANSA.
- Using commitment control may use up more computer resources You should consider this feature of commitment control before attempting to implement it on a large scale.
- When a file is defined to LANSA, an I/O module is created for the file. All LANSA I/O to the file is via the I/O module. It is the I/O module that provides the commitment control support. When an I/O module performs one of the following operations against a file (even a logical file) it always uses the associated / underlying physical file. It is the physical file that LANSA defines as being under commitment control. This feature should be considered with regard to the records that are locked when using commitment control.

Affected operations are:

• write

- update
- delete
- read with locking
- read via relative record number.
- When using commitment control you must begin **journaling** the associated physical file outside of LANSA. Refer to the IBM i operating system commands STRJRNPF (start journaling physical files) and ENDJRNPF (end journaling physical files) for details.
- When attempting to **make operational changes to a file definition** that is being journaled it may be necessary to end journaling beforehand. Failure to do this may cause the "make operational" batch job to fail because it cannot allocate the file for exclusive use.
- After making operational changes to a file definition that uses journaling it will be necessary to begin journaling again.
- **LANSA does not directly support access path journaling.** If you have this requirement please contact your product vendor.

If the commitment control options available when setting up a file definition are used, they specify that a file is under commitment control **all** the time in **all** applications. In this situation, the starting and ending of commitment control is the responsibility of the programmer.

- When using commitment control on the IBM i you must issue a **STRCMTCTL command** (start commitment control) before accessing LANSA. The STRCMTCTL command is part of the IBM i operating system.
- When using commitment control on the IBM i you must issue an **ENDCMTCTL command** (end commitment control) after exiting LANSA. The ENDCMTCTL command is part of the IBM i operating system.
- To selectively override commitment control for all files subject to modification by a function, specify FUNCTION OPTIONS(*NOPGMCOMMIT). The function will **NOT** be under commitment control all individual file definition options regarding commitment control are overridden and superseded.

*PGMCOMMIT

To **selectively use commitment control** within just one function, you can use the FUNCTION OPTIONS(*PGMCOMMIT) command to specify function

level commitment control is required. When this option is used, **all** files subject to modification by the function are automatically placed under commitment control. Commitment control is automatically started and ended by the program. All individual file definition options regarding commitment control are overridden and superseded when using this option.

- When a function fails through the *ABORT value for an RDML keyword or the execution of the ABORT RDML command, the user exit F@ENDCMT is called. As supplied, this consists of the IBM i command ENDCMTCTL. When this is executed with pending changes in an interactive job, the user is sent an inquiry message with the options to roll back the pending changes, cancel the ENDCMTCTL or commit the pending changes. When this is executed with pending changes in a non-interactive job, the pending changes are implicitly rolled back.
- When a function fails through any other reason, commitment control is left active with any pending changes. When the job ends, whether normally or abnormally, the pending changes are implicitly rolled back.

Under certain circumstances, LANSA automatically starts and ends commitment control using User Exit F@BGNCMT - Start Commitment Control and User Exit F@ENDCMT - End Commitment Control. This happens for:

- Functions with OPTIONS(*PGMCOMMIT).
- SuperServer jobs when Start Commitment Control is set to Y or 1.
- LANSA Open jobs when LceSetCommitmentOn has been called before opening the session.
- LANSA Open for .Net jobs when the Session's CommitmentControlOnServer property is True on connection.

Using *DBOPTIMIZE / *DBOPTIMIZE_Batch

If you write an RDML program like this:

```
REQUEST FIELDS(#PRONUM)
```

FETCH FIELDS(#PANEL01) FROM_FILE(PRODMST) WITH_KEY(#PR

FETCH FIELDS(#PANEL01) FROM_FILE(PRODCAT) WITH_KEY(#PR(

then compile and execute it, the objects being used would look something like this:



This is a typical LANSA application, using I/O module calls to control all access to all database files.

Using this approach has several advantages, the chief of which is the ability to alter the definition of a database file and/or its validation rules without ever having to recompile the RDML functions that access it.

Only the I/O module usually needs to be recompiled.

However, there is a disadvantage as well. Using this approach involves a performance overhead in the call to the I/O module.

Usually this overhead is small and acceptable, given the advantages it brings, however in some situations the call overhead becomes magnified by other factors.

The factors can range from accessing a file with a very large number of records to trying to run a large and complex application system on a small or undersized machine. To alleviate this call overhead, a facility called *DBOPTIMIZE (or *DBOPTIMISE) can be used.

If the program above was changed to be like this:

```
FUNCTION OPTIONS(*DBOPTIMIZE) or OPTIONS(*DBOPTIMISE)
REQUEST FIELDS(#PRONUM)
```

FETCH FIELDS(#PANEL01) FROM_FILE(PRODMST) WITH_KEY(#PR

FETCH FIELDS(#PANEL01) FROM_FILE(PRODCAT) WITH_KEY(#PR(

then (re)compiled and executed, the objects being used would look something like this:



This alleviates the I/O module call overhead imposed by the IBM i operating systems.

This technique also produces efficient code, because only the code required for the specific database access method(s) required by the RDML function are actually brought into the RDML function.

It can produce significant performance benefits in some situations.

However, it also has several disadvantages that you should be aware of:

- **Recompilations of RDML functions are now required** If the definition of a database file is changed, all functions that use *DBOPTIMIZE, and reference the file (directly or indirectly), need to be recompiled.
- **RDML functions produce larger compiled objects** Since the RDML function now contains more database access logic, it will

produce larger compiled objects.

• **RDML functions take longer to compile** Since the RDML function now contains more database access logic, it will take longer to compile.

Some RDML commands cannot be used

The current release of LANSA does not allow the BROWSE or RENAME commands to be used in an RDML function that uses the *DBOPTIMIZE option. These restrictions are checked by the full function checker.

• Virtual derivation logic may be more complex

Virtual field derivation code (refer to section on virtual fields) can be made more complex when working in a *DBOPTIMIZE environment. For instance, 2 different database files have the same data structures and subroutines defined to handle a date reversal. This works fine when using I/O modules, but if an RDML program is written that accesses both files, it will fail to compile because the data structure and subroutine would be included into the translated RPG code twice. This problem can be overcome by using the VC_COPY command. Refer to the section of this guide that deals with virtual fields for details.

• RDML level overrides may not work the same way

Some LANSA users have RDML functions that issue specific file overrides by setting the file up as SECURE=NO and then issuing overrides by calling QCMDEXC or QCAEXEC. These may not work as they do when using I/O modules because IBM i and CPF handle overrides issued at the same invocation level differently to overrides issued at different invocation levels.

• A limit to how many files that can be accessed now exists

When using I/O modules, no effective limit as to the number of files that could be accessed in an RDML function existed. However, since *DBOPTIMIZE causes all the required code to be centralized into the RDML function, the RPG restriction now applies. To calculate file usage total use the following:

- If the RDML function uses DISPLAY, REQUEST or POP_UP commands, count 1.
- For each file referenced in the RDML function, count 1.
- For each file referenced in the RDML function for insert, update, delete or fetch by relative record, count another 1.
- For each report produced by the RDML function, count 1.

- For each table/code file validation check referenced by one or more of the files referenced in the RDML function, count 1.
- For each batch control file referenced by one or more of the files referenced in the RDML function, count 1.
- For each file accessed by one or more access routes which have predetermined join fields that are referenced in the function count 1. If this total exceeds 49, the translated RPG code will not compile. Note that one file spot is reserved by LANSA, so the limit is 49, not 50. This limit is **not** checked by LANSA in the current release.
- Making file changes operational required before compiling RDML When I/O modules are used, RDML functions that access a file can be coded and compiled before the file actually exists. However, when using *DBOPTIMIZE:
 - New database files must be made operational, with or without an I/O module, **before** attempting to compile any RDML function that references the file.
 - Modified database files must be made operational **before** attempting to (re)compile new or existing RDML functions that access the file.
 - After a changed file definition has been made operational, all existing RDML functions that reference the file, directly or indirectly, and use *DBOPTIMIZE, should be recompiled.

None of these situations is specifically checked for by the current release of LANSA. However, the cross-reference abilities of LANSA have been enhanced to aid in the identification of all RDML functions that use *DBOPTIMIZE.

• Accessing "OTHER" files with omitted fields requires more care When "OTHER" (i.e. non_LANSA) files are made known to LANSA, fields within the file are sometimes "omitted" from the loaded definition because they are too long, have an unknown type or conflict with the definition of a field already in the dictionary.

For example: a S/36 file which has no external definition may be loaded. It contains one 300 byte field called DATA. The load procedures generally ignore this field, and the developer uses the "virtual fields" facility to assemble and disassemble the field DATA during reads from and writes to

the file.

When an I/O module is used, this all works okay, and field DATA is extracted from the external file definition, allowing it to be accessed by virtual field derivation code, although, strictly speaking, LANSA does not know that this field exists.

However, if the file is used with *DBOPTIMIZE and another file also contains a field called DATA, two different types of conflicts are possible:

- Field DATA has a different type and/or length in both files. This is the best conflict as the RDML function will not compile, clearly indicating a conflicting situation. The only resolution in the current release of LANSA, for this situation, is not to use *DBOPTIMIZE.
- The field DATA has the same attributes in both files. In this situation it is possible that the value of field DATA will be overwritten erratically, possibly affecting user defined virtual derivation code. In this situation the use of *DBOPTIMIZE is not recommended unless very extensive and thorough testing is performed.

Some other things to think about when using *DBOPTIMIZE are:

• Using *DBOPTIMIZE_BATCH for large volume batch updates/deletes

The *DBOPTIMIZE_BATCH keyword invokes the **same** facilities as *DBOPTIMIZE, but in a form that is more appropriate for batch applications doing large volumes of update or delete operations.

To understand how *DBOPTIMIZE_BATCH changes the way file records are processed, you must first understand how records are processed by I/O modules, or by normal *DBOPTIMIZE logic.

Consider the following RDML function:

FUNCTION OPTIONS(*DBOPTIMISE)

-- SELECT FIELDS(#FIELD01) FROM_FILE(TEST)

- CHANGE FIELD(#FIELD01) TO('#FIELD01 * 1.1')
- UPDATE FIELDS(#FIELD01) IN_FILE(TEST)
- -- ENDSELECT

This RDML logic is translated into I/O requests like this:

SELECT - Read next record from file TEST (in keyed order)

and save its associated relative record number (RRN). The record is not locked because the open data path being used is only open for input, not for update.

UPDATE | Reread the record from file TEST via an alternate | open data path using its RRN. This RRN access is | much faster than access by key. The record is | locked because this open data path is open for | update.

Check for any changes to the record since it was originally read via the input open data path.

Update the record via the update open data path and release the record lock.

ENDSELECT - Go back to the SELECT command

This logic produces very strong program logic, does not leave records locked for a long period of time, and may actually work faster than an *DBOPTIMIZE_BATCH version of the program when low percentages of the records read by the SELECT loop are updated. This is because the input open data path being used to read records does not have to lock every record it reads.

However, when large volumes of the records are being updated (or deleted), the extra open data path and the extra RRN I/O performed on it, may impose an unnecessary overhead.

To remove this overhead, you can make the following change to the function: FUNCTION OPTIONS(*DBOPTIMIZE_BATCH)

This changes the translated I/O requests to be like this ...

SELECT - Read next record from file TEST (in keyed order).

The record is locked because the open data path being used is open for update.

UPDATE | Update the file record. No check for changes | since the record was read is required because the | file record has been locked since it was read. Use the same open data path as the read operation.

ENDSELECT - Go back to the SELECT command

which will probably execute faster than the *DBOPTIMIZE version. Some other things to note about *DBOPTIMIZE_BATCH include:

- The I/O logic used for read only files in RDML programs (i.e. files that do **not** have any INSERT, UPDATE or DELETE operations performed against them) is identical, regardless of whether *DBOPTIMIZE or *DBOPTIMIZE_BATCH is used.
- The I/O logic used for update files is changed. Files that have UPDATE, INSERT or DELETE operations performed against them do use different logic. This logic only opens one data path to the file (instead of 2), and it is open for update operations. This means that a record read from the file by any I/O command like SELECT or FETCH leaves the record locked until a subsequent read, update or delete operation releases the lock.
- The previously stated method for calculating how many files can be used in a function when using *DBOPTIMIZE is slightly changed. Files accessed for update, insert or delete do not have to be counted twice when *DBOPTIMIZE_BATCH is used.
- Do not use *DBOPTIMIZE_BATCH in online functions unless **extreme** care is taken. The use of *DBOPTIMIZE_BATCH in online programs that have any form of screen panel interaction is an inappropriate use of this facility, and it will most likely cause all sorts of record locking (and releasing) problems that will enormously complicate the function logic. Do **not** use this facility in this type of function.
- Do not use *DBOPTIMIZE_BATCH in programs that use FETCH, UPDATE or DELETE operations that address records by relative record number (RRN). Such operations use the WITH_RRN parameter of the associated command. The use of the RETURN_RRN parameter is okay.
- Using data file SHARE options can improve processing

The sharing of an open data path (SHARE=YES) facility provided by IBM i and CPF does not have any significant performance impact when using I/O modules, because all I/O is centralized into one I/O module anyway.

However, when using *DBOPTIMIZE, I/O operations become dispersed

across several RDML functions, so using SHARE=YES can have performance benefits.

Please ensure that the concept of an open data path and the conflict and interference implications of using an open data path are fully understood before attempting to use this option. Additionally, the difference between a shared open data path and a shared access path should be clearly understood.

Use of SHARE=YES on files under batch control logic (i.e. the file the batch control totals are kept in) is not recommended as it will almost certainly cause shared open database usage conflicts.

• Cross-reference reports show RDML functions using *DBOPTIMIZE

The cross-reference reports for field and file usage clearly indicates RDML functions that use the *DBOPTIMIZE facility. This enables functions that need to be recompiled to be identified quickly.

• You don't have to have an I/O module

If all RDML functions that access a file use *DBOPTIMIZE, then the file's I/O module is not really required as it will never be invoked. In this situation, you can actually define the file so as to permanently indicate that no I/O module is required.

Refer to Database File Attributes for details of this option. If this option is used to indicate that no I/O module is required, any existing I/O module will be deleted to maintain future synchrony between the database and I/O module(s).

Please note that the "File Maintenance Utility" performs database access via I/O modules **only**. Attempting to use the FMU on a file that does not have an I/O module will cause it to fail.

Generally this is not a limitation because a purpose built RDML function, using *DBOPTIMIZE, can be generated in a matter of minutes from an application template to perform any role for which the FMU was required.

• Be selective about the use of *DBOPTIMIZE

Most commercial applications more or less follow the 80/20 rule. This

means that 20% of the application programs are used 80% of the time. If this is true of your system, you should plan on using *DBOPTIMIZE in the 20% of the applications, and not bother with the other 80%. It is better to optimize an online inquiry that is used all day by lots of users, than a table update routine that is used once a month.

Trigger Functions

What is a Trigger Function? Creating a Trigger Function Activating a Trigger Function Exactly When Are Triggers Invoked? The TRIG_OPER and TRIG_RETC Variables and TRIG_LIST Working List What Codes Are Passed in TRIG_OPER to the Trigger? How Many Entries Are Passed in the TRIG_LIST? What Return Codes Are Used in TRIG_RETC and How Can They Be Set? Triggers - A Classic Example Examples of how Triggers Might be Used Triggers - Restrictions and Limitations Triggers - Some Do's and Don'ts

What is a Trigger Function?

A trigger function is a type of LANSA function which will be invoked **automatically** when a specific type of I/O operation occurs to a file and when a specific set of conditions are met.

For example, when an application developer defines the "Cancel of an Order" via the RDML command DELETE FROM_FILE(ORDHDR) WITH_KEY(#ORDNUM), they have initiated an "event", which may automatically cause other functions to be "triggered".

When the order is canceled it may "trigger" the following:

Activity A Flag Order Historical Details

Activity B Print Outstanding Credit Invoices

Activity C Send a Message to the Sales Department

Additionally, the list of activities that happen when an order is canceled can be added to or changed at **any time** without having to change or even recompile the original DELETE FROM_FILE(ORDHDR) function.

A trigger function allows a business activity to be associated directly with a database file (i.e. the "object"). When a specified event happens to information in the file, then the trigger(s) will be automatically invoked.

For example, if the business rules stated that when an order is canceled you must also perform activities A, B and C, then a "traditional design" would include A, B and C as direct logic (or calls) into the interactive function called "Cancel an Order".

In fact there may be several sources from which an Order may be canceled:

Source The typical interactive "Cancel an Order" transaction.

Source Monthly Batch Automatic Canceling of Unfilled Orders. 2

Source Requests arriving via LANSA Open transactions from sales peopleusing dial up PC systems.

and the most "dangerous" source of all:

SourceThe transaction that will be defined by someone else in 2 yearsXtime.

You have to remember to include the A, B and C activities (or at least the initiation of it) into sources 1 through 3 now.

In fact it is often the last case, "Source X", that will cause the most problem when the new designer fails to realize that the A, B and C logic exists, or even to understand fully how and when it is used.

Triggers solve all these problems because they link the activities A, B and C to the "object" Order, and thus are always invoked, no matter what the "source" of the event is.

And best of all, you can add new activities D, E and F to Order, at any time, without having to change any of the event "sources" in any way.

Sensible use of triggers may transform the way that an application is designed. The user interface can be fully designed, and then the complexities and rules can be introduced later by using data dictionary validation rules and database trigger functions.

The resulting design is much more in the "object oriented" style.

Triggers separate "business function" from "user interface" in a much clearer and easier way.

Creating a Trigger Function

To define a function as being a trigger function, use the TRIGGER parameter on the FUNCTION command.

If the function is to act as a data dictionary level trigger, enter *FIELD into the first part of the parameter, and the associated data dictionary field name into the second part of the parameter.

If the function is to act as a database level trigger, enter *FILE into the first part of the parameter, and the associated database file name into the second part of the parameter. The file specified must be a physical file.

As an aid to defining new trigger functions LANSA is shipped with the following Application Templates that can be used to form the base of a trigger function:

BBFLDTRIG Field Level Trigger Function

BBFILTRIG File Level Trigger Function

When a function is defined as a trigger function you **must** follow these guidelines:

- The parameter RCV_LIST(#TRIG_LIST) must be used.
- The parameter RCV_DS must not be used.
- Option *DIRECT must also be used.
- Options xxx_SYSTEM_VARIABLE or xxx_FIELD_VALIDATE must not be used.
- The list #TRIG_LIST must be defined by a DEF_LIST command as DEF_LIST NAME(#TRIG_LIST) TYPE(*WORKING) ENTRYS(2) and must not include any fields in the FIELDS parameter. The required fields will be automatically added.
- No DISPLAY, REQUEST or POP_UP commands may be used. This is a deliberately imposed design/usage constraint that may be removed in later versions.
- No CALL can exist to another process/function. This is a deliberately imposed design/usage constraint that may be removed in later versions.
- Trigger functions cannot be defined within an action bar process. This is not to say that they cannot be referenced from within an action bar, it just means

that a trigger function cannot be defined as part of a process that is of action bar type.

- The associated process must not have any parameters.
- The exchange list may not be used. This is a deliberately imposed design/usage constraint imposed to enforce insulated and modular design and use of trigger functions.

When a function is defined as a trigger function you should follow these guidelines in most situations:

- Understand how triggers are defined and how they should be used by reading the Field Rules/triggers sections and Trigger Functions section.
- Use options *NOMESSAGES and *MLOPTIMIZE.
- Options *HEAVYUSAGE and *DBOPTIMIZE may also be considered.
- Do not directly or indirectly access the database file that the trigger is, or will be, linked to.
- Where triggers are heavily and constantly invoked avoid resource intensive operations. Such operations will slow down access to the associated file. Whenever reasonable make the trigger "submit" another transaction thus not delaying the source of the event significantly.
- Recursive implementations may be defined, but will fail to execute correctly. For instance a field trigger function invoked during an insert to file A could attempt to insert data into file B, possibly causing itself to be invoked in a recursive situation, and thus to fail.

Activating a Trigger Function

To associate a data dictionary field or a file to a trigger function it is necessary to take either the "Review, change or create field rules and triggers" from the field control menu if the trigger is to be created at dictionary level, or the "Review or change file rules and triggers" if the trigger is to be created at the file level.

When the "Add a Trigger" option has been selected, the field must then be associated with a trigger function.

Information can then be entered which will specify when the trigger function is to be activated (for example, before an update of the field is carried out), and if the trigger function should only be activated under certain conditions. See Field Rules/Triggers for details on entering trigger information.

Exactly When Are Triggers Invoked?

Before Open / Close

Before open/close triggers are invoked immediately before an attempt is made to open or close a file (or a view of it).

A before open trigger is invoked when a file or a view is opened and or another view has not already opened the file.

This means that if your logic is "Open View 1", then "Open View 2" (where view 1 and view 2 are both based on the same file) then the trigger would be invoked when you open view 1, but not when you open view 2.

A before close trigger is invoked when a file or a view is closed and another view still does not have the file open.

This means that if your logic is "Close View 1", then "Close View 2" (where view 1 and view 2 are both based on the same file) then the trigger would be invoked when you close view 2, but not when you closed view 1.

After Open / Close

Is invoked identically to the "before" options but immediately after a successful attempt has been made to close the file.

Before Read

Is invoked immediately before an attempt is made to read a record from a file. Before input triggers have no access to "information" from the file (because the information has not been input yet) so their use should be considered carefully. Access to the key(s) being used to access the file is not possible in this mode so do not design triggers based on this premise.

After Read

Is invoked after a record has been successfully read from a file and just before the details of the record are passed back to the invoking function. Any virtual field logic has been completed by this stage.

Before Insert

Is invoked immediately before an attempt is made to insert a new record into a file. Please note the following:

- The trigger is run even if the requester uses CHECK_ONLY(*YES)
- The insert may still fail (e.g.: duplicate key error). Before insert triggers should not perform database changes. If database changes are to be done
move the trigger into the "after insert" position instead.

• All virtual logic has been completed when the trigger is invoked.

After Insert

Is invoked immediately after a new record has been inserted into a file. Please note the following:

- The trigger is not run when the requester uses CHECK_ONLY(*YES)
- At the time of invocation all batch control logic has been executed.
- If AUTOCOMMIT is used then the commit is issued before the trigger is invoked.

Before Update

Is invoked immediately before an attempt is made to update an existing record in a file. Please note the following:

- The trigger is run even if the requester uses CHECK_ONLY(*YES)
- The update may still fail (e.g.: duplicate key error). Before update triggers should not perform database changes. If database changes are to be done move the trigger into the "after update" position instead.
- All virtual logic has been completed when the trigger is invoked.

After Update

Is invoked immediately after an existing record has been updated in a file. Please note the following:

- The trigger is not run when the requester uses CHECK_ONLY(*YES)
- At the time of invocation all batch control logic has been executed.
- If AUTOCOMMIT is used then the commit is issued before the trigger is invoked.

Before Delete

Is invoked immediately before an attempt is made to delete an existing record from a file. Please note the following:

- The trigger is run even if the requester uses CHECK_ONLY(*YES)
- The delete may still fail (even though very unlikely). Before delete triggers should not perform database changes. If database changes are to be done move the trigger into the "after delete" position instead.

After Delete

Is invoked immediately after an existing record has been deleted from a file. Please note the following:

- The trigger is not run when the requester uses CHECK_ONLY(*YES)
- At the time of invocation all batch control logic has been executed.
- If AUTOCOMMIT is used then the commit is issued before the trigger is invoked.

The TRIG_OPER and TRIG_RETC Variables and TRIG_LIST Working List

When a trigger function is invoked it receives 2 things from the invoker:

TRIG_OPER

TRIG_OPER: is an A(6) field which must be defined in the data dictionary. The content of this field defines what database operation has, or is about to be, performed. Refer to What Codes Are Passed in TRIG_OPER to the Trigger? for details.

TRIG_LIST

TRIG_LIST: is a 2 entry working list containing 0,1 or 2 entries. The number of entries passed depends upon the database operation being performed. Refer to How Many Entries Are Passed in the TRIG_LIST? for details.

You must define TRIG_LIST as a working list with 2 entries but you must **not** define any fields within it. The required fields are automatically defined by the RDML compiler.

If your trigger is for field #CUSTNO then the single field #CUSTNO is automatically defined in the list just as if you had typed in:

DEF_LIST NAMED(#TRIG_LIST) FIELDS(#CUSTNO)

If your trigger was for file Z which contained real fields X, A, T and virtual fields Q and B then the list is automatically defined just as if you had typed in:

DEF_LIST NAMED(#TRIG_LIST) FIELDS(#X #A #T #Q #B)

This automatic definition ensures that the correct names are used and that you do not have to know or key in the correct names in the correct order.

Remember that the automatic definition is done from the "active" definition of file Z. So if you changed file Z to have X, A, V, T, Q and B as fields and then recompiled the trigger **before** you made the changed file Z "operational" it would automatically define from the unchanged "active" (X, A, T, Q, B) version of file Z.

If you then made the file Z "operational" it would set up its trigger invocations using X, A, V, T, Q, B as the list layout. This is a clear mismatch and would cause unpredictable results.

This mistake would be typified by decimal data errors or by data being "offset" within fields in the list. If this type of problem occurs when a trigger is invoked you should recompile it.

So, when changing a file definition, always make the file "operational" **before** attempting to recompile its associated trigger functions.

To "get" values from the list use the GET_ENTRY command.

This means that when using a trigger for field #CUSTNO you must "get" the correct value of #CUSTNO from the list by using GET_ENTRY NUMBER(?) FROM_LIST(#TRIG_LIST).

Likewise, to get the values of #X through #B in the file Z example you would need also need to "get" them by using GET_ENTRY NUMBER(?) FROM_LIST(#TRIG_LIST).

Only use the list operations SELECTLIST, GET_ENTRY and UPD_ENTRY against the list TRIG_LIST. Only ever issue UPD_ENTRY operations against entry number 1. When a trigger function terminates it returns 2 things to the invoker:

TRIG_RETC: is an A(2) field which must be defined in the data dictionary. At the point of return it must be set to "OK", "ER", and in some situations, "VE". See the following sections for more details of the meaning and use of these return codes.

TRIG_LIST: is the 2 entry working list containing 0,1 or 2 entries previously described. You may alter the data in the **first** entry passed by using the UPD_ENTRY command. If you do this in a "before" operation then you will actually alter the data that is inserted or updated into the file.

Likewise if you do this in an "after read" operation you will alter the data that is passed back to the function that issued the read request.

It is strongly recommended that you do **not** use this facility to "communicate" between serially invoked trigger functions.

Only use the list operations SELECTLIST, GET_ENTRY and UPD_ENTRY against the list TRIG_LIST. Only ever issue UPD_ENTRY operations against entry number 1.

What Codes Are Passed in TRIG_OPER to the Trigger?

When a trigger function is invoked it can access field TRIG_OPER to determine what operation was in progress when it was invoked.

The values passed in TRIG_OPER are:

-	
Operation In Progress	Value In TRIG_OPER
Before Open	BEFOPN
After Open	AFTOPN
Before Close	BEFCLS
After Close	AFTCLS
Before Read	BEFRED
After Read	AFTRED
Before Insert	BEFINS
After Insert	AFTINS
Before Update	BEFUPD
After Update	AFTUPD
Before Delete	BEFDLT
After Delete	AFTDLT

How Many Entries Are Passed in the TRIG_LIST?

When a trigger is invoked it is passed a working list called TRIG_LIST that contains details of the field or file record that is being actioned.

When invoked, TRIG_LIST may contain 0, 1 or 2 entries:

Number of Entries	Meaning / Content
0	There are no details available in this context (e.g.: before open, before read).
1	There is one set of details available in the current context (e.g.: after read).
2	There are two sets of details available in the current context. Entry 1 is the new details and entry 2 is the previous details. (e.g.: before/after update images).

The number of entries passed in the list varies with the operation in progress according to the following table:

Operation In Progress	Number Of Entries In TRIG_LIST
Before Open	0
After Open	0
Before Close	0
After Close	0
Before Read	0
After Read	1
Before Insert	1
After Insert	1
Before Update	2

After Update	2
Before Delete	1
After Delete	1

What Return Codes Are Used in TRIG_RETC and How Can They Be Set?

Whenever a trigger function completes its completion status is subjected to 3 tests:

- 1. The return code TRIG_RETC is checked for "OK" (uppercase). If not "OK", then the trigger is deemed to have failed.
- 2. The function completion status is tested. If not okay, then the trigger is deemed to have failed. A function will give a "bad" completion status if it issues an ABORT command, or hits an ENDCHECK with no last display, or uses an invalid array or substring reference, etc, etc.
- 3. The IBM i completion status is tested. If not okay, then the trigger is deemed to have failed (e.g.: trigger function not found).

When a trigger function is deemed to have failed, a return code is then issued to the actual invoking function issuing the database operation (i.e. the function doing the SELECT or UPDATE or INSERT) according to the following table:

Operation In Progress Return Code

Before Open	OK, ER
After Open	OK, ER
Before Close	OK, ER
After Close	OK, ER
Before Read	OK, ER
After Read	OK, ER
Before Insert	OK, VE
After Insert	OK, ER
Before Update	OK, VE
After Update	OK, ER
Before Delete	OK, VE
After Delete	OK, ER

You should not return values in TRIG_RETC outside of those specified in this table.

Nor should the invoking RDML I/O command be set up to do any special "trapping" or "handling" when it knows that there is an underlying trigger.

Such an approach would create very complex designs and defeat the whole purpose for which triggers were introduced (i.e. being "invisible" to the upper layer of functionality).

RDML functions doing I/O operations are recommended to not use the IO_ERROR or VAL_ERROR parameters (like any other normal RDML functions).

Leave the default values and let the standard error handling solve the problem. Use a "binary" approach to doing I/O operations - it either worked or it didn't work - and if it didn't work let the standard error handling solve the problem.

Coding your own I/O error traps in RDML functions is not recommended unless they are of a very specialized nature (e.g.: setting up work files). Failing to observe this recommendation will lead to overly complex implementations that exhibit no real business benefit and cost significantly more to develop and maintain.

An "OK" response indicates that the operation completed normally.

An "ER" response sets the IO_ERROR parameter in the RDML command that issued the I/O request.

A "VE" response sets the VAL_ERROR parameter in the RDML command that issued the I/O request.

Note that the "VE" response is **only** possible for before insert, before update and before delete.

This allows triggers in these positions to act like "extended validation checkers".

However, a trigger set up as an "extended validation checker" cannot actually "flag" a specific field as being in error.

It can indicate an error has occurred, and it can issue error message details, but it cannot flag the specific field in error in the same way that a normal validation rule can, or in the same way that a normal validation checking function can (see function option *xxx_FIELD_VALIDATE).

However, an "extended validation checker" defined as a trigger has one

advantage over a normal validation checking function: it has access to all the values in the record of a file that is being inserted, deleted or updated - whereas a normal validation function only has access to the individual field value with which it is associated.

Using a trigger as an "extended validation checker" is a very powerful facility, especially when the "before and after images" available to the before update check are considered, and as such it can be very useful at times.

However:

- Reserve the use of "extended validation triggers" for truly complex situations. Do not use this facility without even considering the normal dictionary facilities.
- Where "extended validation check" type triggers are to be used, have just one per file and encapsulate all the rules inside it. You then have just one trigger that supports the action "Validate".
- Make trigger functions exhibit "insulated modularity". Like action bar functions they should exhibit these "OO" like characteristics:
 - They should perform one and only one "action".
 - They should not expect other triggers or virtual code logic to precede, or to follow them.
 - They should operate "standalone".
 - They should be small and robust. When a trigger is invoked to perform an action it should just do that single action or issue an error message indicating why it cannot.

Triggers - A Classic Example

The following example is a classic example of how a trigger function should be used.

It takes a complex business rule and "encapsulates" it into a trigger.

Next the trigger is linked to the associated database file and the business rule is performed automatically whenever the specified event occurs.

It is a classic example because it clearly demonstrates how triggers can "encapsulate" complex rules and associate them directly with the "object" (i.e. file).

The Business Problem

ACME Engineering run a payroll system.

The Employee Master file (EMPL) contains two fields called "SALARY" and "WEEKPAY".

SALARY is the annual salary that the company has contracted to pay the employee.

WEEKPAY is the amount paid to the employee each week.

WEEKPAY is arrived at via a complex set of rules.

For a new employee the WEEKPAY calculation is relatively simple, but when an employee's SALARY is changed the complex calculation involves both the new SALARY figure and the **previous** SALARY figure.

The Trigger Function

The first step in defining the trigger is to define the trigger function that encapsulates **all** the WEEKPAY rules into one and only one place.

This is a fundamental of good trigger design.

The following function may have been coded to handle this:

FUNCTION OPTIONS(*DIRECT *NOMESSAGES *MLOPTIMIZE) RCV_LIST(#TRIG_LIST) TRIGGER(*FILE EMPL)

*/

/* Define the standard trigger list which will contain the */

/* before and after images of the EMPL file record. These */

/* fields are automatically added to the list definition */

/* by the RDML compiler.

DEF_LIST NAME(#TRIG_LIST) TYPE(*WORKING) ENTRYS(2)

/* Now examine exactly what event has occurred */

CASE OF_FIELD(#TRIG_OPER)

/* A new employee is being created */

WHEN VALUE_IS('= BEFINS') GET_ENTRY NUMBER(1) FROM_LIST(#TRIG_LIST) << calculate correct value into field WEEKPAY >> UPD_ENTRY IN_LIST(#TRIG_LIST)

/* An existing salary has been changed */

WHEN VALUE_IS('= BEFUPD')

DEFINE FIELD(#OLDSALARY) REFFLD(#SALARY) GET_ENTRY NUMBER(2) FROM_LIST(#TRIG_LIST) CHANGE FIELD(#OLDSALARY) TO(#SALARY)

GET_ENTRY NUMBER(1) FROM_LIST(#TRIG_LIST) << calculate correct value into WEEKPAY >> << using OLDSALARY in the calculations >> UPD_ENTRY IN_LIST(#TRIG_LIST)

OTHERWISE

ABORT MSGTXT('WEEKPAY trigger function invalidly invoked')

ENDCASE

CHANGE FIELD(#TRIG_RETC) TO(OK) RETURN

Activating the Trigger Function

Now that the trigger function has been defined it needs to be activated. To do this, access the definition of file EMPL and associate two trigger invocation events with it.

The first would be specified as "BEFORE INSERT" and would not have any associated conditions. This means that the trigger function will be called whenever an attempt is made to create a new employee.

The second would be specified as "BEFORE UPDATE" and would have an associated condition which would look something like this:

SALARY NEP SALARY

i.e. salary is not equal to previous salary

which says that the trigger should be activated "BEFORE UPDATE" but only if the employee's SALARY has changed.

Defining the "BEFORE UPDATE" event like this is very efficient because it means that the trigger will not be activated when the employee's salary has not been changed (which will probably be most of the time).

If WEEKPAY had to be recalculated when the SALARY changed or when the COMPANY that the employee worked for changed, then you would define the invocation event like this instead:

SALARY NEP SALARY OR COMPANY NEP COMPANY

- i.e. salary is not equal to previous salary
 - or company is not equal to previous company

If WEEKPAY was to always be recalculated, then you would not have to define two separate invocation events. You could simply define one event (with no conditions) and indicate that the trigger should be invoked "BEFORE INSERT" and "BEFORE UPDATE".

Of course, this means that every single insert or update of an employee would cause the trigger function to be invoked.

Key Things to Note About this Example

This example demonstrates some of the key elements of good trigger design and use:

- The "encapsulation" principle. The WEEKPAY calculation "method" is "encapsulated" in **one and only one** function. If it has to be changed it only has to be changed in one place.
- Deferment. The existence of the WEEKPAY method does not have to be defined, or even known about, during initial system design.

This also means that a "method" can be introduced into an application design at any time. For instance, the WEEKPAY method does not have to be defined before any applications that create or update employees are. The create and update applications can be defined and tested first. When the WEEKPAY method is created and defined it will immediately begin to affect the processing of all existing applications.

- Reusability. The WEEKPAY calculation method is automatically and implicitly reused by any application that creates or changes employee details. The trigger could be activated from a normal NPT device via an "Employee Maintenance" function, or from a PC application via the LANSA Open facility.
- Transparency. The fact that the WEEKPAY logic is present and being used is invisible and probably immaterial to an RDML builder creating an "Employee Maintenance" function.
- Separation of the "method" from the "event". The trigger function defines what to do when an "event" happens (i.e. the "method").

However, it does **not** have to detect the occurrence of the event.

For example, the function defined previously defines a "method" called "Calculate Weekly Pay".

The business rules says that weekly pay must be (re)calculated when a new employee is taken on, or when an existing employee's salary is changed, or when an existing employee moves to another company.

The actual "event" is defined in the LANSA data dictionary.

Examples of how Triggers Might be Used

Example 1: Calculates the current balance of the account:

Object : Account (ACNT)

Trigger Field : "Account Balance" (ACCBAL)

Trigger Method : Calculates the current balance of the account.

Trigger Event(s) : 1. AFTER READ when ACCBAL is REF (referenced)

Invoked By : (i.e. How is an account balance retrieved ?)

FETCH FIELDS(....#ACCBAL....) FROM_FILE(ACNT) or SELECT FIELDS(....#ACCBAL....) FROM_FILE(ACNT)

Comments : Quite efficient because ACCBAL is only calculated when the requester asks for it.

Example 2: Submits batch order print job:

Object : Order (ORDR)

Trigger Field : "Print Required" (PRINT_REQ)

Trigger Method : Submits batch order print job. Batch function prints order and updates PRINT_REQ to 'N'.

Trigger Event(s) : 1. AFTER INSERT when PRINT_REQ EQ 'Y'

2. AFTER UPDATE when PRINT_REQ EQ 'Y' and PRINT_REQ NEP PRINT_REQ

Invoked By : (i.e. How is an order submitted for printing ?)

CHANGE #PRINT_REQ 'Y' UPDATE FIELD(#PRINT_REQ) IN_FILE(ORDR)

Comments : It may seem a little obscure as an IBM i based trigger but what if the update comes from a PC based application via the LCOE facility, or from a batch job that starts automatically every morning and automatically selects certain orders for printing ?

Triggers - Restrictions and Limitations

Trigger logic on associated batch control files is not performed. If file A uses file B as a batch control header file, then triggers associated with B are not invoked when I/O operations to A need to insert/update file B "batch header" records.

Triggers - Some Do's and Don'ts

Some Do's

- Do experiment with small test cases using triggers so that you are comfortable with what they are and how they work before attempting to implement a complex application involving triggers.
- Do remember that when you change the type or length of a field in the data dictionary (that has associated triggers) you should recompile:
 - All trigger functions associated with the field.
 - All I/O modules of files that contain the field as a real or virtual field.
 - All functions that make *DBOPTIMIZE references to file(s) containing the field.
- The list of objects to recompile is easily obtained by producing a full listing of the definition of the field.
- Remember that when you change the layout of a database file (that has associated triggers) you should recompile:
 - The I/O module of file.
 - All trigger functions associated with the file.
 - Any functions that make *DBOPTIMIZE references to the file.

The list of objects to recompile is easily obtained by producing a full listing of the definition of the file.

Some Don'ts

- Do not do any I/O to the file with which the trigger is linked. Attempting such I/O directly, or indirectly, may cause a recursive call to the file I/O module. Do not attempt to use *DBOPTIMIZE to circumvent this rule. Such attempts will cause the file cursor of the active I/O module to become lost or corrupted.
- Do not use triggers on files that have more than 799 real and virtual fields (the 800th field position is reserved for the standard @@UPID field).
- Do not make triggers too expensive to execute. For example, an unconditioned trigger that is always executed after reading from a file doing, say, 3 database accesses, will at least quadruple the time required to read the base file. Triggers are a very useful facility but they are **not** magic. When you set up a trigger to do a lot of work, then your throughput will be reduced

accordingly. The use of triggers and the estimation of the impact that they exert on application throughput is entirely your responsibility as an application designer.

- Do not introduce dependencies between triggers. For instance, trigger A (before update) sets a value in field X, say. Setting up trigger B (also before update), to run after trigger A, with the "knowledge" that trigger A has been executed first (and thus set field X) is not a good idea. This is an example of "interdependence" between triggers and it is **not** a good way to use triggers. In this case the logic in trigger B should be inserted directly into trigger A following the point that it sets a value into field X.
- Do not use ABORT when a user exit is called from a Trigger function. When ABORT is issued in the Trigger Function, the I/O Module is able to intercept the ABORT and passes a Trigger error status back to the Function. However, when the ABORT is issued in the (user exit) Function, called by the Trigger, the ABORT is interpreted in the standard way because the Function is not aware that the call was from a Trigger and it does not make any difference. Using ABORT in these situations (e.g. validations) is not recommended.
- It is **very strongly recommended** that you do not design triggers in such a way as that "normal" RDML functions doing I/O operations are "aware" of their existence, and attempt to **directly** "communicate" with them in any way (e.g.: *LDA, data areas, etc).

Where trigger "requests" are to be supported, introduce a virtual (or real) field into the file definition and use it to "fire" the trigger in the normal way.

User Defined Reporting Attributes

User Defined Reporting Attributes (UDRA) give LANSA reports access to the many features of IBM i External Printer files. Currently LANSA reports are internally described in data structures, while UDRA's involve the use of Data Description Specifications (DDS).

Note: It is highly recommended that the IBM Manual *Programming: Data Description Specifications Reference* be reviewed before continuing. The use of UDRA's require technical knowledge of IBM i. Use of UDRA's will make your LANSA function **PLATFORM DEPENDENT** as DDS is only available on the IBM i.

UDRA's are specified as output attributes on the data dictionary, DEFINE and OVERRIDE commands. UDRA's have the form "URxx"; where xx is any alpha-numeric combination that contains no blanks.

UDRA's will permit access to features such as barcodes, different fonts and sizes etc in your reports.

UDRA's are defined by creating a member in the Source Physical file **DC@UDRA** with the same name as the required UDRA (e.g. UDRA = URB1, member name = URB1). This source member is of type "PRTF" and contains valid DDS statements that will be applied to the report.

These DDS statements must include valid keywords and/or comments, and be applicable to the level for which they are intended to be used. That is, if keywords are to be applied to a field then they must be applicable to the FIELD LEVEL (refer to IBM Manual *Programming: Data Description Specifications Reference* for more information).

It is recommended that the IBM Source Entry Utility (SEU) be used to enter UDRA DDS source.

It may be necessary to change the Printer File to DEVTYPE(*IPDS) or DEVTYPE(*AFPDS), depending on the DDS Keywords used. (The DEVTYPE() required for the different keywords is defined in the above mentioned IBM Manual). The Printer File DEVTYPE() is best modified via the DEF_REPORT parameter OTHER_OVR. e.g. DEF_REPORT REPORT_NUM(1) OTHER_OVR('DEVTYPE(*IPDS)').

LANSA cannot determine the effect of DDS keywords on the report spacing, therefore it is the user's responsibility to ensure that the report spacing is correct for their application.

When a UDRA is attached to a report no validation is done as to the existence of the corresponding member in the file DC@UDRA.

If a UDRA is attached to a report or UDRA statements are modified, the changes are not realized until the function is successfully recompiled.

The export and import of the UDRA source file, DC@UDRA is the responsibility of the user.

Step-by-Step Example

Print a field on a report as a barcode.

- 1. Review IBM Manual "*Programming: Data Description Specifications Reference*" and decide BARCODE is the keyword to use.
- Decide on the Keyword parameters necessary to produce required barcode. e.g. BARCODE(EAN8 2 *HRI) - barcode field using type EAN-8, barcode is 2 characters in height and includes Human Readable Interpretation below the barcode.
- 3. Select a UDRA code to represent the barcode. HINT: create standard UDRA's (URxx) that can be re-used for other reports. e.g. URBC.
- 4. Edit a new member of type "PRTF" in the file DC@UDRA with the same name as the UDRA code. e.g. URBC.
- 5. Enter the Keyword with parameters into the source member.
- 6. Save and Exit the source member.
- 7. Specify the UDRA code against the field by using the Data Dictionary, DEFINE or OVERRIDE command.
- 8. Create the report as you normally would through LANSA, except you MUST include FUNCTION OPTION(*OS400_EXT_PRINT) to utilize the UDRA's (for restrictions when using *OS400_EXT_PRINT see the command definition for FUNCTION in the *Technical Reference Guide*).
- 9. Compile the report.

The configuration of printers is beyond the scope of the document, but this problem has been encountered and it is not documented in IBM manuals:

When a printer device description is configured as AFP(*YES), the user must set the correct physical size of the page using the MPP, MPL, CPI and LPI values in the printer hardware.

Hindi Numerics

In many Arabic countries the numeric keypad shows two sets of numeric characters. The normal 0 through 9 (called ARABIC) plus the HINDI numeric characters. HINDI numerics are treated as characters by the IBM i.

To support this type of display a field attribute may be used to cause LANSA to translate the numeric field into its character HINDI equivalent when being displayed on a screen or printed on a report. The reverse translation applies when the HINDI field is input from a screen.

The "HIND" attribute in Field Input and Output Attributes Lists may be used as either an input or output attribute. Irrespective of which list the attribute is specified in, the Hindi translation will apply to all input and output of the field.

The translation to and from HINDI in no way affects LANSA's treatment of the field. LANSA still considers the field to be numeric. Therefore all edit codes, edit words and other numeric only operations can still be used.

When a field that contains the 'HIND' attribute is displayed on a screen, or printed on a report LANSA converts the numeric field to a character field. Any edit code or edit word that is attached to the field is applied, before the Arabic characters are translated to their Hindi equivalent.

When a field that contains the 'HIND' attribute is input from a screen, LANSA converts the character field back to a numeric field. All editing characters are removed, and any decimal points and negative signs recorded, and Hindi characters are translated back to their Arabic equivalent.

These translations to and from Hindi characters fields are transparent to users and developers of LANSA applications. The use of character fields and the methods used for the translations causes some limitations and restrictions.

Limitations and Restrictions

- The field must be numeric (Type S or P)
- The maximum field size is 30,9.
- Packed numeric fields on reports MUST be of odd length.
- Fields which contain decimal places MUST use an edit code or edit word which displays the decimal point. If this is not done unexpected results may occur.
- Fields which are to allow negative values MUST use an edit code or edit word which displays a negative balance indicator (the minus (-) sign , CR

symbol and floating minus are permitted).

• The 'HIND' attribute is an IBM i specific feature and may not be supported on other platforms.

GUI WIMP Constructs

GUI stands for Graphical User Interface, a type of user interface that takes advantage of high-resolution graphics. When applied to NPT's (Non-Programmable Terminals) there is a text subset of the graphical model.

WIMP represents the constructs used within a GUI environment. WIMP stands for Windows, Icons, Menus and Pointing devices and applies to the components being implemented.

GUI WIMP constructs are used in LANSA by defining a particular input/output attribute against a field. The attribute may be specified via the data dictionary, DEFINE or OVERRIDE command, and the PUT FIELD Built-In Function.

Only one GUI attribute may be used per field, for both the input and output attribute list.

DDS keywords were deliberately NOT used to implement these GUI WIMP constructs.

The implementation of these GUI WIMP constructs in LANSA was done to allow enhancement of applications which are primarily run via Visual LANSA. Implementation of these GUI WIMP constructs into applications that are solely run on NPT's (Non-Programmable Terminals) is at the discretion of the designer.

Due to the varying implementations on different platforms additional spacing may be used so that positioning of the GUI WIMP fields is consistent across platforms. Refer to Designing Client/Server Applications in the LANSA Application Design Guide for more information.

Also see

Push Buttons Check Boxes Radio Buttons Drop-Down Lists Guidelines, Hints and Tips

Push Buttons

A push button is an area on the screen labeled with text, which represents an action that will be initiated when a user selects it.

- It must be an alphanumeric field with a maximum length of 60.
- The text stored in the variable will be centered and displayed as the push button text.
- It must be no identification (*NOID).
- Push buttons are defined through the PBnn input/output attribute, where nn is the push button number from 01 to 20.
- Each push button may only be used once per screen command (i.e., DISPLAY/ REQUEST/ POP_UP). A particular push button attribute (i.e., PB08) may be used against more than one field in a function as long as those fields do not appear in the same screen command.
- Due to the graphical representation on PC displays, push buttons must be separated from other information immediately above and below it by a blank line.
- To determine if a push button has been pressed, test the IO\$KEY field for a push button value. Push button 1 to 9 returns "B1" to "B9" and push button 10 to 20 returns "BA" to "BK".

```
BEGIN_LOOP
REQUEST FIELDS(#SCREEN)
IF COND('#IO$KEY *EQ "BC")
MESSAGE MSGTXT('Button 12 was Pressed')
ENDIF
END_LOOP
```

and also through the WAS() option of the IF_KEY command:

```
BEGIN_LOOP
REQUEST FIELDS(#SCREEN)
IF_KEY WAS(*BUTTON1)
MESSAGE MSGTXT('Button 1 was Pressed')
ENDIF
END LOOP
```

Check Boxes

A check box is a choice with associated text that requires a true/false, yes/no, on/off answer.

- It must be an alphanumeric field of length 1.
- Check boxes are defined through the CBOX input/output attribute. The maximum number of check boxes that may be defined for a function is 999.
- Identification must be either label (*LABEL) or no identification (*NOID). Label is the default.
- Within RDML the valid values for a check box field are "1" for selected and " " for unselected. Use of other values may produce unpredictable results.
- When on a NPT screen a check box is selected with a "/" or countrydesignated character (defined in the DC@OSVEROP *SELECTCHAR=x option) and unselected with a " ". This character is locked in at compile time for the best performance when executing.
- LANSA generated code handles the conversion between screen and RDML values for check boxes (i.e. from "/" to "1" when selected).
- A new system value named *CHECKBOXSELECTED has been created to be used when testing the return value of a check box field. Use of this system variable makes your RDML code "self-documenting".
- A check box may only contain one of two values. Therefore the best way to test its value is to check if it is *EQ (equal) or *NE (not equal) to the selected value (*CHECKBOXSELECTED).
- To test if a check box has been selected:

```
BEGIN_LOOP
REQUEST FIELDS(#SCREEN)
IF COND('#CBOX01 *EQ *CHECKBOXSELECTED')
MESSAGE MSGTXT('Check box 1 is selected')
ENDIF
IF COND('#CBOX02 *NE *CHECKBOXSELECTED')
MESSAGE MSGTXT('Check box 2 is not selected')
ENDIF
ENDIF
```

the correct way to initialize check boxes:

CHANGE FIELD(#CHECKBOX1 #CHECKBOX2 #CHECKBOX3 #CHECKBOX4) TO(*NULL)

IF COND('#ACTIVE *EQ "Y"')

CHANGE FIELD(#CHECKBOX2) TO(*CHECKBOXSELECTED) ENDIF

CHANGE FIELD(#CHECKBOX4) TO(*CHECKBOXSELECTED)

Radio Buttons

A choice with text beside it. Radio buttons are combined to show a user a fixed set of choices from which only one can be selected.

- It must be an alphanumeric field of length 1.
- Radio buttons are defined through the RBnn input/output attribute, where nn = 01 to 99. The nn refers to the group that the radio button belongs to. The maximum number of radio buttons that may be defined for a function is 999.
- Identification must be either label (*LABEL) or no identification (*NOID). Label is the default.
- Within RDML the valid values for a radio button field are "1" for selected and " " for unselected. Use of other values may produce unpredictable results.
- When on a NPT screen a radio button is selected with a "/" or countrydesignated character (defined in the DC@OSVEROP *SELECTCHAR=x option) and unselected with a " ". This character is locked in at compile time for the best performance when executing.
- LANSA generated code handles the conversion between screen and RDML values for radio buttons (i.e. from "/" to "1" when selected).
- There are certain restrictions enforced when using and initializing radio buttons. One and only one radio button per group MUST be selected, or unpredictable results may occur.
- A new system value named *RADBUTTONSELECTED has been created to be used when testing the return value of a radio button field. Use of this system variable makes your RDML code "self-documenting".
- A radio button may only contain one of two values. Therefore the best way to test its value is to check if it is *EQ (equal) or *NE (not equal) to the selected value (*RADBUTTONSELECTED).
- To test if a radio button has been selected:

BEGIN_LOOP

```
REQUEST FIELDS(#SCREEN)

IF COND('#RADIO01 *EQ *RADBUTTONSELECTED')

MESSAGE MSGTXT('Radio button 1 is selected')

ENDIF

IF COND('#RADIO02 *NE *RADBUTTONSELECTED')

MESSAGE MSGTXT('Radio Button 2 is not selected')
```

ENDIF END_LOOP

• The correct way to initialize radio button:

CHANGE FIELD(#RADIO1 #RADIO2 #RADIO3 #RADIO4) TO(*NULL)

CASE OF FIELD(#TAXRATE) WHEN VALUE_IS('= 0') FIELD(#RADIO1) TO(*RADBUTTONSELECTED) CHANGE VALUE IS('= 10') WHEN FIELD(#RADIO2) TO(*RADBUTTONSELECTED) CHANGE WHEN VALUE_IS('= 20') FIELD(#RADIO3) TO(*RADBUTTONSELECTED) CHANGE WHEN VALUE_IS('= 30') FIELD(#RADIO4) TO(*RADBUTTONSELECTED) CHANGE ENDCASE

Drop-Down Lists

A control that is a variation of a list box. A drop-down list only displays one item until the user takes an action to display the other objects or choices.

- Must be an alphanumeric field with a maximum length of 60.
- Any identification is valid.
- On NPT's the field will display the same as a normal field, except that the field will be output-only and followed by the drop-down indicator "V", or the character specified in DC@OSVEROP *DROPDOWN=x option. This character is locked-in at compile-time for the best performance when executing.
- Drop-Down lists are defined through the DDxx input/output attribute, where xx is AA to 99. The xx defines the drop-down list and links the field to the list data.
- Cannot wrap onto another line.
- A drop-down list may be removed (or effectively cleared) by the DROP_DD_VALUES Built-In Function described in the *Technical Reference Guide*.
- The list is created and data is added to the drop-down list through the ADD_DD_VALUES Built-In Function described in the *Technical Reference Guide*.
- It is suggested that a site and/or application standard be created for dropdown attributes. The advantages of this is that conflicts are prevented and drop-downs could be loaded in an initialization function then the drop-down attribute could be specified in any function after that. For example the dropdown attribute 'DDST' could be loaded with valid state codes. Then any field that requires a state code to be entered, just requires the 'DDST' attribute to be specified against it for the drop-down list to be active.
- A drop-down list exists for the entire life of your LANSA session.
- There is a limit of 20 active drop-down lists per LANSA session. More than 20 lists may be added and dropped, but the total number at any one time cannot exceed 20.
- There is also a limit on the size of each drop-down list. The maximum size of each drop-down list, including value separators is 6200 characters.
- The DROP_DD_VALUES Built-In Function should always be used

BEFORE a drop-down list is loaded with its initial data to ensure there is no other data in the list.

- The separator character specified should not exist in the data being added or unexpected results will occur.
- There are a number of ways of using the ADD_DD_VALUES Built-In Function to load the data into a drop-down list. The following examples illustrate some of the different methods.

One value at a time. Hard-coded values.

DEFINE FIELD(#DROPDATA) TYPE(*CHAR) LENGTH(10) INPUT_ATR(DDST)

- USE BUILTIN(DROP_DD_VALUES) WITH_ARGS(DDCO)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDCO *DFT 'RED') TO_GET(#RETCD)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDCO *DFT 'BLUE') TO_GET(#RETCD)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDCO *DFT 'YELLOW') TO_GET(#RETCD)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDCO *DFT 'GREEN') TO_GET(#RETCD)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDCO *DFT 'WHITE') TO_GET(#RETCD)

Many values at once, separated by the defined separator character.

DEFINE FIELD(#DROPDATA) TYPE(*CHAR) LENGTH(3) INPUT_ATR(DDST)

- USE BUILTIN(DROP_DD_VALUES) WITH_ARGS(DDST)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDST "'/"' 'NSW/VIC/QLD/SA/WA/NT/TAS') TO_GET(#RETCD)

One value at a time, loaded from a file.

DEFINE FIELD(#DROPDATA) TYPE(*CHAR) LENGTH(24) INPUT_ATR(DDCT) OVERRIDE FIELD(#CATEGORY) TO_OVERLAY(#DROPDATA 1) OVERRIDE FIELD(#CATDESC) TO_OVERLAY(#DROPDATA 5)

- USE BUILTIN(DROP_DD_VALUES) WITH_ARGS('DDCT')
- SELECT FIELDS((#CATEGORY) (#CATDESC)) FROM_FILE(CATMST)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS('DDCT' *DFT #DROPDATA)

ENDSELECT

Guidelines, Hints and Tips

- It is highly recommended that the IBM publication "Common User Access: Guide to User Interface Design" be reviewed before attempting to implement GUI WIMP constructs.
- Before you start utilizing the new contsructs you should first familiarize yourself with them. Write a few pilot functions and experiment with the constructs, to ensure that they operate in the fashion that you require.
- GUI WIMP constructs cannot be defined on fields used in browse lists.
- Due to the graphical representation on PC displays, push buttons must be separated from other information by a blank line.
- Because of the implementation of radio buttons on NPT's their use on NPT's is discouraged.
- It is recommended that if more than 50 entries are needed for a drop-down list than an alternative method should be considered (i.e., POP_UP, Prompt etc). Large numbers of entries in a drop-down make the list cumbersome and hard to use, opposing the main reason for their use.
- The choices in a radio button group should be formatted in one of two ways: vertically in a left-aligned column, or horizontally on the same line. An imaginary rectangle encompassing the radio button group must not contain any other fields, or portability to other platforms will be affected.
- There are two methods of implementing a drop-down list. One is where the valid field codes are loaded into a drop-down list, then the user selects a code. This method can be confusing when the codes are not well known.

For example #PAYTERM is a field that contains the payment terms for a customer. Its valid values are A, B, C, D and E, but without any description the user could be confused as to there meaning and select the wrong code.

DEFINE FIELD(#PAYTERM) TYPE(*CHAR) LENGTH(1) INPUT_ATR(DDPT)

- USE BUILTIN(DROP_DD_VALUES) WITH_ARGS(DDPT)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDPT "'/''' 'A/B/C/D/E') TO_GET(#RETCD)

```
BEGIN_LOOP
REQUEST FIELDS((#PAYTERM))
END_LOOP
```

A better solution would be to load the valid field codes and descriptions into the drop-down list using a different field. The code field then overlays the new field in the position that the code appears. So when the entry in the dropdown list is selected the code field is automatically updated with the code from the drop-down entry because of the overlay.

DEFINE	FIELD(#PAYDROP) TYPE(*CHAR) LENGTH(24)
IN	PUT_ATR(DDPT)
DEFINE	FIELD(#PAYTERM) TYPE(*CHAR) LENGTH(1)
ТО	_OVERLAY(#PAYDROP 1)

- USE BUILTIN(DROP_DD_VALUES) WITH_ARGS(DDPT)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDPT *DFT 'A - PAYMENT WITH ORDER') TO_GET(#RETCD)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDPT *DFT 'B - CASH ON DELIVERY') TO_GET(#RETCD)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDST *DFT 'C - 7 DAYS') TO_GET(#RETCD)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDST *DFT 'D - 30 DAYS') TO_GET(#RETCD)
- USE BUILTIN(ADD_DD_VALUES) WITH_ARGS(DDST *DFT 'E - UNLIMITED') TO_GET(#RETCD)

BEGIN_LOOP

REQUEST FIELDS((#PAYDROP)) END_LOOP

ILE Implementation

It is not the aim of this section to explain the workings of the Integrated Language Environment (ILE), but rather LANSA's implementation of ILE. It is recommended the IBM Manual *ILE Concepts* be reviewed and understood before reading on.

The decision to migrate to the ILE environment should first involve addressing the advantages and disadvantages of ILE. Once the decision is made to migrate to ILE environment and its methodology (i.e.many small, modular programs, call intensive) it becomes uneconomical to revert back.

The ability to run and develop ILE programs is enabled by different system features. To develop ILE applications requires the ILE RPG/400 compiler. The ILE RPG/400 compiler must be purchased separately.

LANSA places all called programs into the default activation group (i.e.*CALLER is used in the ACTGRP() parameter of CRTPGM). In this way ILE programs behave in the same manner as OPM programs.

There is a memorandum from IBM to IBM i ILE RPG/400 customers on the subject of migrating to ILE RPG/400 that should be read. In summary this document mentions that:-

- "The disk storage requirements for ILE/RPG programs have increased to 1.5 to 3 times relative to OPM RPG/400"
- "If you recompile a traditional OPM RPG/400 program with the ILE RPG/400 compiler, you will experience compile-time degradation of greater than 2X the performance compared to OPM RPG/400 compiler."
- Suggestions for the amount of memory needed by the ILE RPG/400 compile process are to "Use 12-16MB to compile medium sized programs" and "Use 20 MB to compile large programs".

Also see

Compilation Options Debug Service Programs

Compilation Options

With OPM programs LANSA uses the CRTRPGPGM command to compile source, but with ILE RPG/IV LANSA uses the CRTRPGMOD command. The differences between the two commands are:

- OPM uses OPTION(*OPTIMIZE) to optimize the compilation object and ILE uses OPTIMIZE(*FULL).
- OPM programs are always compiled with debug information included and have it stripped-out later if requested. Whereas ILE compiles have a DBGVIEW() parameter that controls the level of debug information included. If debug information is required in an ILE program, LANSA uses the *STMT level.
- OPM uses the IGNDECERR() parameter to ignore decimal data errors, ILE replaced this parameter with the FIXNBR() parameter that actually fixes decimal data errors. These different responses to decimal data errors must be considered when migrating to ILE.

For more information on these parameters refer to the *ILE RPG/400 Reference Guide*. The implications of these differences must be considered before deciding to migrate to ILE.
Debug

Due to the IBM i changes to debug for ILE, LANSA debug has also been updated. LANSA debug no longer requires programs to be compiled with IBM i debug information included.

Therefore there is no necessity to compile functions with debug information included. This applies to both OPM and ILE programs. This should help reduce the impact of the storage size increase of ILE programs.

Service Programs

LANSA ships with service programs that include many of the LANSA internal programs. These internal programs will be bound-by-reference into the function program when the function is compiled with the *ILE flag set in ILE bind RPG/IV modules into program in Compile and Edit Settings.

Function Routing

What is Function Routing?

Function Routing allows you, at execution time, to automatically (re)route all invocations of a function to another function.

This automatic facility has been provided specifically for the use of software vendors who ship applications to customers and need to customize parts of the application to exactly match their customers' requirements.

So, if you are a software vendor and always ship basic function A with your package but need to make a customized version of function A for a particular customer, Function Routing allows you to automatically (re)route, for that customer, all invocations of function A to the new function, named A1.

The original variant (A) stays intact, exactly as you ship it, and the variants (e.g.: A1, A2, A3, etc) can be modified to match unique customer requirements. A special table called the Function Routing Table manages the automatic routing of an invocation of function A to A1.

All LANSA generated applications, be they IBM i or PC based, support the concept of function routing.

Also see

What is the Function Routing Table?Creating a Function Routing TableEditing a Function Routing TableRules You Must Understand Before Using a Function Routing TableWhat Type of Functions Can Be (re)Routed?How is the Function Routing Table Used?

What is the Function Routing Table?

The function routing table is a very simple source file called X_FUNRTR (IBM i systems) or X_FUNRTR.DAT (PC based systems) that specifies how functions should be automatically routed. (Note that routed functions must all be in the same process.)

Source lines in the X_FUNRTR files can be input via standard source editors (SEU or Edit File EDTF on the IBM i or Notepad on a PC) and have this format:

ffffffff,ttttttt

where "ffffffff" specifies the "from" function name and "ttttttt" specifies the "to" function name.

For example, the entries:

A,A1 B678V40,B678V42

specify that invocations of function A should be routed to function A1, and that invocations of B678V40 should be routed to function B678V42.

Of course it is assumed that A/A1 and B678V40/B678V42 have identical entry characteristics (such as number of parameters, type of parameters, etc).

Creating a Function Routing Table

On an IBM i

1. Decide in which library the routing table is to reside. This is normally the partition's module library. However, it is possible for the routing table to reside in a different library. If the routing table is to reside in a different library, the *FUNRTRLIBL flag in data area DC@OSVEROP must be set and the library must be in the library list when the routing takes place.

You can set the *FUNRTRLIBL flag via the Use function routing table from*LIBL option in Work with Execution and Security Settings.

- 2. Use the command CRTSRCPF to create a standard IBM i source file named X_FUNRTR in the selected library. Add a single member called X_FUNRTR to the file.
- 3. Use the command CHGOBJOWN (Change Object Owner) to change the owner of the source file just created to be the LANSA system owner (usually QOTHPRDOWN).

On a PC

There is no need to independently create this file. It will be created when you edit it, if you follow the notes for a PC in Editing a Function Routing Table.

Editing a Function Routing Table

On an IBM i

Use the command STRSEU or EDTF to edit the source file X_FUNRTR (member X_FUNRTR) in the selected library.

On a PC

Select the *Function Routing* option from the *Settings and Administration* menu of your Visual LANSA system to open the *Edit a Function Routing Table* dialog box.



Enter the partition for which the function routing table (X_FUNRTR.DAT) is required and press *OK*. If you need a refresher of the contents of the Function Routing Table, press the *Help* button.

After you press *OK*, the Notepad editor will be opened with the current X_FUNRTR.DAT file's details. If you haven't added any entries previously, this will be an empty file.

When you have finished your changes, select Save from the File menu. It will

saved in the correct location for the selected partition.

Rules You Must Understand Before Using a Function Routing Table

• On a IBM i, Function routing tables must reside in the partition's module library or another library, if *FUNRTRLIBL is set. If a library other than the partition's module library is used it must be in the library list when the LANSA command executes. If it is not the existence of the routing file will not be recognized.

You set *FUNRTRLIBL via the *Use function routing table from *LIBL* option in *Work with Execution and Security Settings*.

- For PC applications, Function routing tables must reside in the source directory. If the routing table is not placed into the correct directory then its existence will not be recognized.
- Function routings are partition based, and only apply to, the partition that is associated with the routing table.
- You can only specify a single function routing per line, in the form ffffffff,tttttt in a function routing table. The from function (fffffff) must begin in column 1. It must be immediately followed by a single comma, which in turn must be followed by the to function (tttttt) name. No other information can be included in the routing entry line. Entries that do not follow these formatting rules will be ignored and no exception is issued or noted.
- From and to function names in the routing table must be specified in **uppercase** characters.
- Up to **4500** entries can be specified in the function routing table (comments are not included in this count). Entries beyond entry number 4500 are ignored and no warning or error message is issued.
- You must not use the reserved function names EXIT, EOJ, MENU, RETRN, ERROR, HELP, SELECT, EOJ, *ANY, Cn, Fn or Pn (where "n" is in the range 1 to 999999) as either the "to" or "from" function names in a function routing table. Use of these reserved names may cause unpredictable results and/or application failure.
- You can enter comments into a function routing table. If a function routing table entry line starts with an "*" in column 1, it is regarded as a comment and it is ignored. For example:

* Route all standard reports to tailored versions GLR01,GLR0101

GLR02,GLR0201 GLR03,GLR0301 * End of routing table

- The function routing table is loaded into memory at the time that the LANSA (IBM i) or X_RUN (PC) command is invoked. This is done to optimize access to the routing entries in the table. It also means that changes you make to a routing table will not be effectively recognized until you exit and re-invoke the LANSA or X_RUN command.
- The function routing table is looked up once for a "from" function, to find a "to" function. Once a "to" function has been located, it is not looked up again. You must not form routing "chains". The example below must not be used to route a direct reference of A to A2. The Function Routing Table Verification will produce warnings if you do this and unpredictable results or errors may occur at execution time.

Function Routing Table Verification

While all these rules may seem complex, a facility is provided that allows you to verify the contents of your FRT (Function Routing Table) after it has been updated. To use this facility you must:

- Update the FRT as required.
- **On an IBM i system:** use the command LANSA REQUEST(VERIFYFRT) PARTITION(ppp). This command will produce a spool file (QSYSPRT) that details the verification of your updated FRT. You should review all the messages in this file, and if required, take corrective action before running the verification process again.
- **On a PC system:** use the command X_RUN PROC=*VERIFYFRT PART=DEM USER= etc, etc (refer to the X_RUN command for details of other parameters). This command will produce a display that details the verification of your updated FRT. You should review all the messages in this file, and if required, take corrective action before running the verification process again.

The reason that the "update-verify" model has been chosen for the FRT (Function Routing Table) is simply that the load and lookup of the FRT has to

be optimized for performance. This means that no effective analysis of the FRT can be performed when it is actually being used because the performance impacts would be significant.

Finally, the fatal error notification panel, used for IBM i applications notes, that function routing was in effect at the time that the application failed. This is a visual reminder that the presence and use of the FRT should be taken into account when you are trying to determine the cause of the application error.

What Type of Functions Can Be (re)Routed?

Only the following types of functions, under the following rules, circumstances and guidelines, can be automatically (re)routed by the function routing table. Failure to observe the rules and circumstances described may lead to application failure and/or unpredictable results.

- **Functions invoked from an SAA/CUA style menu:** providing that the "to" function belongs to the same process as the "from" function. If the "from" function is an *Attached Function* on the menu process, the "to" function must also be defined as an *Attached Function*. Note that the function must appear on the menu as the "from" function. When defining a variant "to" function you should set the "Display on Menu" option to NO so that it does not appear on the process menu. When the user selects the "from" function from the menu the request is automatically (re)routed into the "to" function.
- Function invoked from an ACT/BAR style menu: providing that the "to" function belongs to the same process as the "from" function. If the "from" function is an *Attached Function* on the menu process, the "to" function must also be defined as an *Attached Function*. Note that the function must appear in the action bar as the "from" function. Normally the "to" function is not attached to the action bar itself, but it effectively "adopts" the action bar and pull down option codes from the "from" function. In effect, it becomes the "from" function.
- Functions invoked by a CALL(process) FUNCTION(function) command: providing that the "to" function belongs to the same process as the "from" function.
- Functions invoked by a CALL(*DIRECT) FUNCTION(function) command: in all circumstances (but the normal *DIRECT calling rules must still be observed). This includes server based RDML functions called from client based applications.
- **Batch Functions invoked by a SUBMIT command:** providing that the "to" function belongs to the same process as the "from" function.
- System Variable Evaluation Functions: in all circumstances.
- Data Dictionary Validation Functions: in all circumstances.
- **Event Trigger Functions:** in all circumstances.

How is the Function Routing Table Used?

On an IBM i, when you start a LANSA application running via the LANSA command, the data area DC@OSVEROP is examined for the flag *FUNRTRLIBL.

- If the flag does not exist a check is made for the existence of a file called X_FUNRTR (i.e. a Function Routing Table) in the library associated with the current partition.
- If the *FUNRTRLIBL flag exists, a check is made for the existence of a file X_FUNRTR in *LIBL.
- If you choose to use the *FUNRTRLIBL flag you must ensure that the first X_FUNRTR file in the library list is the correct routing table for the current partition. If it is not, unpredictable results may occur.

You set *FUNRTRLIBL via the *Use function routing table from *LIBL* option in *Work with Execution and Security Settings*.

On a PC, when you start a LANSA application running via the the X_RUN command a check is made for the existence of a file called X_FUNRTR (i.e. a Function Routing Table) in the directory associated with the current partition.

If an X_FUNRTR file is found then its entries are loaded into memory. (This is why you must exit and re-invoke any active LANSA or X_RUN command for any changes that you make to the function routing table to become effective). All entries in the table are sorted into order, to optimize the time it takes to look up the table. Finally, a system-wide flag is set to indicate that function (re)routing is in effect.

The setting of the system flag is very significant because every single function call operation in LANSA is sensitive to the setting of this flag. The following "invoke a function" logic demonstrates why this is significant:

if (function routing is in effect)
 test to see if call to specified function should
 be (re)routed
 if it should be rerouted replace specified function name
 with(re)routed name
endif
invoke the specified function

You can see that where function routing is not in effect, no significant

performance impact exists (other than to test the system-wide flag).

However, when function routing is in effect, a performance impact results in all invoked function operations because the function name has to be looked up in the function routing table. Also there is an overhead when LANSA is started because the Function Routing Table has to be sorted and loaded. The larger the table the more time required for this initial load.

The function routing table is not checked for LANSA debug. Therefore, if it is required to debug a routed function, the "to" function must be specified when turning on debug mode.

Built-In Function Notes

Database Connection Email Built-In Functions Notes Zip Built-in Function Notes

Database Connection

DEFINE_DB_SERVER can be used in isolation from the other BIFs to just override the connection parameters and database type or it can be used with the full set of related BIFs in this sequence:

DEFINE_DB_SERVER, CONNECT_SERVER, CONNECT_FILE.

The full set of BIFS is only needed when the OTHER File is in a database with a different DSN to the one with which it was loaded.

To connect to all the default databases on startup, that is all the databases defined in the OAMs, it is necessary to DEFINE_DB_SERVER and CONNECT_SERVER for each database using the DSN that is in each OAM. CONNECT_FILE is not required if the DSN is the same as in the OAM. That is, an OTHER File is implicitly connected to the database from which it was loaded.

If your environment has a development, test and production version strategy, the simplest way of managing the differing locations of Other Files as the application passes through the various stages, is to use the same ODBC DSN but alter the definition of it to point to a different physical database. Thus, the default database embedded in the OAM will access a different physical database.

Email Built-In Functions Notes

An email message that is to be sent must be constructed commencing with a call to MAIL_START followed by other calls in the Email Handling series to add the data. The message is then sent using the MAIL_SEND call.

Note: Only use Email Handling Built-In Functions in applications that are to fully execute under the control of IBM i or Windows.

In the Windows environment, these Built-In Functions should not be used in functions that are invoked on remote Windows server systems.

Note: For Email BIFs to work locally, you must set the PROFILE correctly. To find the correct PROFILE, go to Outlook\Tools\Options\Mail Service.

Zip Built-in Function Notes

The ZIP BIFS utilise Info-ZIP's compression and uncompression utilities via unzip32.dll, zip32.dll and unzipsfx.exe. Info-ZIP's software (Zip, UnZip and related utilities) is free and can be obtained as source code or executables from Internet/WWW sites, including ftp://ftp.freesoftware.com/pub/infozip/Info-ZIP.html. Please refer to Info-ZIP's WWW site(s) for further information on the .ZIP file format and functionality of zip and unzip.

PKWARE introduced the .ZIP file format in 1989. According to strict interpretation of the zipfile specification (as specified by PKWARE and amended by Info-ZIP), the following limits apply to all zipfile archives:

Limit	Maximum
Number of Files	65536
Uncompressed size of a single file	4 GB
Compressed size of a single file	4 GB
Maximum size of archive that can be created	256 GB
Maximum size of archive that can be extracted	4 GB

Using the wildcard * in file specifications: If you want to include (or ignore) all files that are named readme or readme.* then add readme* to the list (rather than readme.*).