LANSA/AD

Specialized Built-In Functions Guide

Release 7.0

Contents

Chapter 1. Field Related Built-In Functions

DLT_FIELD

Category: Data Dictionary built in functions

Description: Deletes a field definition from the LANSA data dictionary.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of field to be deleted from data dictionary	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code. OK = field details returned ER = field not accessible. In case of "ER" return code error message(s) are issued automatically.	2	2		

GET_FIELD

Category: Data Dictionary built-in functions

Description: Retrieves attributes of a field stored in the LANSA data dictionary and returns them to the calling RDML function.

Returned values are exactly as per information presented on the "Review or Change field definitions" screen described in the User Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Arguments

No	**Type A/N**	**Req/ Opt**	**Description**	**Min Len**	**Max Len**	**Min Dec**	**Max Dec**
1	A	Req	Name of field to be retrieved from data dictionary	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = field details returned ER = field not accessible In case of "ER" return code error message(s) are issued automatically.	2	2		
2	A	Opt	Field type A = alphanumeric S = signed decimal numeric P = packed decimal numeric	1	1		
3	N	Opt	Length of field or total number of digits in field Note - maximum no. of digits for AS/400 is 30.	3	15	0	0
4	N	Opt	Number of decimal positions Not applicable to type A field	1	15	0	0
5	A	Opt	Reference field name	1	10		
6	A	Opt	Field description	1	40		
7	A	Opt	Field label	1	15		
8	A	Opt	Field column headings List of 3 * A(20) headings Bytes 1-20 are column head 1	1	60		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Bytes 21-40 are column head 2				
			Bytes 41-60 are column head 3				
9	A	Opt	Output attributes list. List of 10 * A(4) attributes	1	40		
10	A	Opt	Input attributes list. List of 10 * A(4) attributes	1	40		
11	A	Opt	Edit code or edit word If first char is a quote (') then value is an edit word. Otherwise it is an edit code. Not applicable to type A field.	1	20		
12	A	Opt	Default value of field	1	20		
13	A	Opt	Optional alias name of field	1	30		
14	A	Opt	System field flag	3	3		
			YES = a system field				
			NO = not a system field				
15	A	Opt	Keyboard shift	1	1		

GET_FIELD_INFO

Category: Data Dictionary built-in functions

Description: Retrieves a list of field related information from the LANSA internal database and returns it to the calling RDML function in variable length working lists.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Field name	1	10		
2	A	Req	Level at which information is requested D = Dictionary level F = File level	1	1		
3	A	Req	Type of field related information to retrieve. Valid types are : FIELDCHECK - Validation rules MLATTR- Multilingual attributes	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
4	A	Opt	Physical file name. Required if file level information is requested.	1	10		
5	A	Opt	Physical file library. Required if file level information is requested.	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = list returned partially or completely filled. No more of this type of information exists for this field. OV = list returned completely filled, but more of this type of information than could fit in the list exists. NR = list was returned empty. Last entry in the list is returned as null. ER = Field not found. Last entry in the list is returned as null.	2	2		
2	List	Req	Header working list to contain field related information. The calling RDML	100	100		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			function must provide a working list with an aggregate entry length of exactly 100 bytes.				

Bytes Description

1-5 Number of the first entry in the detail list for this entry in character format. A value of '00000' denotes that there are no entries in the detail list for this entry.

6-10 Number of the last entry in the detail list for this entry in character format

11-100 Rest of information

<u>For type FIELDCHECK:</u>

Each header list entry is formatted as follows:

Bytes Description

1-5 as above

6-10 as above

11-12 Type of check.
SL = Simple Logic,
DC = Date Check,
CF = File Check,
CL = Complex Logic,
RV = Range of Values,
LV = List of Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			13-42 Description of check				
			43-43 Enable check for ADD. Y = Check performed on ADD, U = Check performed on ADD USE, N = Check not performed on ADD .				
			44-44 Enable check for CHANGE. Y = Check performed on CHG, U = Check performed on CHG USE, N = Check not performed on CHG.				
			45-45 Enable check for DELETE. Y = Enable check, N = Do not enable check				
			46-46 Action if check is true. N = Perform NEXT check, E = Issue fatal ERROR, A = ACCEPT value and do no more checking				
			47-47 Action if check is false. N = Perform NEXT check, E = Issue fatal ERROR, A = ACCEPT value and do no more checking.				
			48-54 Error Message Number				
			55-64 Message File Name				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			65-74 Message File Library 75-84 Name of program to be called to perform Complex Logic check. Blank if not Complex Logic check. 85-94 Name of file used in File check. Blank if not File check. **<u>For type MLATTR:</u>** Each header list entry is formatted as follows: **Bytes Description** 1-5 as above 6-10 as above 11-14 Language code 15-29 Label 30-49 Col heading 1 50-69 Col heading 2 70-89 Col heading 3				
3	List	Req	Detail working list to contain field related information. The calling RDML function must provide a working list with an aggregate entry length of exactly 100 bytes.	100	100		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			<u>For type FIELDCHECK:</u> Type of check <u>Simple Logic</u>: Each detail list entry is formatted as follows: **Bytes Description** 1-79 Condition line Type of check <u>Date Check</u>: One detail list entry is formatted as follows: **Bytes Description** 1 - 8 Date format 9 - 15 Number of days allowed into the past for specified date in character format 16 - 22 Number of days allowed into the future for specified date in character format Type of check <u>File Check</u>: Each detail list entry is formatted as follows: **Bytes Description** 1-20 Value used as a key to the file. Type of check <u>Complex Logic</u>: Each detail list entry is formatted as follows:				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Bytes Description 1-20 Value used as an additional parameter. Type of check <u>Range of Values</u>: Each detail list entry is formatted as follows: **Bytes Description** 1-20 Value used as low limit of range. 21-40 Value used as high limit of range. Type of check <u>List of Values</u>: Each detail list entry is formatted as follows: **Bytes Description** 1 - 20 Value used as a list element. **<u>For type MLATTR:</u>** Each detail list entry is formatted as follows: **Bytes Description** 1 - 40 Field description				

GET_FIELD_LIST

Category: Data Dictionary built-in functions

Description: Retrieves a list of fields and their descriptions from the data dictionary and returns them to the calling RDML function in a variable length working list.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Positioning field value. The returned list starts with the first field from the dictionary whose name is greater than the value passed in this argument.	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain Field information. The calling RDML function	60	60		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			must provide a working list with an aggregate entry length of exactly 60 bytes. Each returned list entry is formatted as follows: **Bytes Description** 1-10 Field Name 11-50 Field Description 51-60 <<future expansion>>				
2	A	Opt	Last field in returned list. Typically this value is used as the positioning argument on subsequent calls to this built-in function.	1	10		
3	A	Opt	Return code. OK = list returned partially or completely filled with field details. No more fields exist beyond those returned in the list. OV = list returned completely filled, but more fields than could fit in the list exist. Typically used to indicate "more" fields in page at a time style list displays. NR = list was returned empty. Last field in the list is returned as blanks.	2	2		

Example

A user wants to customize some field definitions by changing labels, and column headings.

```
GROUP_BY      NAME(#FLDDTL) FIELDS((#FLDNAM *NC) (#FLDDES *NC)
              #FLDLBL #FLDCH1 #FLDCH2 #FLDCH3)
DEF_LIST      NAME(#FLDLST) FIELDS(#FLDNAM #FLDDES #SPARE)
              TYPE(WORKING) ENTRYS(1000)
**********    -Request field-
REQUEST       FIELDS(#STRFLD) TEXT(('Field to start from' 5 5))
**********    -Get list of fields-
USE           BUILTIN(GET_FIELD_LIST) WITH_ARGS(#STRFLD)
              TO_GET(#FLDLST #LAST #RETCOD)
**********    -Process lists-
SELECTLIST    NAMED(#FLDLST)
USE           BUILTIN(GET_FIELD) WITH_ARGS(#FLDNAM) TO_GET(#RETCOD
              #FLDTYP #FLDLEN #FLDDEC #FLDREF #FLDDES #FLDLBL #FLDCOL)
**********    < break column headings into FLDCH1, FLDCH2, FLDCH3 >
**********    -Change field definition-
REQUEST       FIELDS(#FLDDTL)
USE           BUILTIN(PUT_FIELD) WITH_ARGS('NNN' #FLDNAM #FLDLEN
              #FLDDEC #FLDREF #FLDDES #FLDLBL #FLDCOL)
ENDSELECT
```

GET_HELP

Category: Data Dictionary built-in functions

Description: Gets a list of help text for a specified field, function or process.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Object name The name of a field, function or process.	1	10		
2	A	Req	Object extension name If the object type is a function then this value should contain the name of the process that the function is defined in. If the object type is not a function then this value should be blank.	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
3	A	Req	Object type Values: DF - Field PD - Process PF - Function	2	2		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain help text. The calling RDML function must provide a working list with an aggregate entry length of exactly 77 bytes. Each returned list entry is formatted as follows: **Bytes Description** 1-77 Help Text	1	77		
2	A	Req	Return code OK = list returned partially or completely filled with help text for this object No more help text exists for this object. OV = list returned completely filled, but more help text than could fit in the list exist. Typically used to indicate	2	2		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			"more" functions in page at a time style list displays. ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.				

Example

A user wants to retrieve the help text of a specific object and display it without the use of the HELP key.

```
*********   Define arguments and lists
DEFINE      FIELD(#OBJNAM) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#OBJEXT) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#OBJTYP) TYPE(*CHAR) LENGTH(2)
DEFINE      FIELD(#HLPTXT) TYPE(*CHAR) LENGTH(77)
DEFINE      FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2)
DEF_LIST    NAME(#WKHLPL) FIELDS((#HLPTXT)) TYPE(*WORKING)
DEF_LIST    NAME(#BWHLPL) FIELDS((#HLPTXT))
GROUP_BY    NAME(#RQSOBJ) FIELDS((#OBJNAM) (#OBJEXT) (#OBJTYP))
GROUP_BY    NAME(#DSPHLP) FIELDS((#OBJNAM) (#OBJEXT) (#OBJTYP))
*********   Clear working and browse lists
BEGIN_LOOP
CLR_LIST    NAMED(#WKHLPL)
CLR_LIST    NAMED(#BWHLPL)
*********   Request Object Name, Extension and Type
REQUEST     FIELDS(#RQSOBJ) BROWSELIST(#BWHLPL)
*********   Execute built-in-function - GET_HELP
USE         BUILTIN(GET_HELP) WITH_ARGS(#OBJNAM #OBJEXT #OBJTYP)
            TO_GET(#WKHLPL #RETCOD)
```

```
*********   Help text was retrieved successfully
IF          COND('#RETCOD *EQ ''OK''')
*********   Move Help text from the working list to the browselist
SELECTLIST  NAMED(#WKHLPL)
ADD_ENTRY   TO_LIST(#BWHLPL)
ENDSELECT
*********   Allow Help text to be reviewed for the specified object
DISPLAY     FIELDS((#DSPHLP)) BROWSELIST(#BWHLPL)
*********   Working list overflowed, more help text to retrieve
ELSE
IF          COND('#RETCOD *EQ ''OV''')
MESSAGE     MSGTXT('List not big enough to fit all help text')
*********   GET_HELP failed with errors, report error
ELSE
MESSAGE     MSGTXT('GET_HELP failed with errors, try again')
ENDIF
ENDIF
END_LOOP
```

GET_MULTVAR_LIST

Category: Data Dictionary built-in functions

Description: Retrieves a list of multi-lingual variables (*MTXT) and their value in the current language and returns them to the calling RDML function in a variable length working list.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Positioning *MTXT variable. The returned list starts with the first *MTXT variable whose name is greater than the value passed in this argument.	1	20		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain *MTXT variable information. The calling RDML function must provide a working list with an aggregate entry length of exactly 108 bytes. Each returned list entry is formatted as follows: **Bytes Description** 1-20 *MTXT variable name 21-98 *MTXT value in current language 99-108 << for future expansion >>	108	108		
2	A	Opt	Last *MTXT variable in list Typically this value is used as the positioning argument on subsequent calls to this built-in function.	1	20		
3	A	Opt	Return code. OK = list returned partially or completely filled with *MTXT variable details. No more *MTXT variables exist beyond those returned in the list. OV = list returned completely filled, but	2	2		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			more *MTXT variables than could fit in the list exist. Typically used to indicate "more" *MTXT variables in page at a time style list displays. NR = list was returned empty. Last *MTXT variable in the list is returned as blanks.				

Example

A user wants to print a list of all *MTXT variables.

```
DEF_LIST      NAME(#MTXLST) FIELDS(#MTXNAM #MTXVAL #SPARE)
              TYPE(*WORKING) ENTRYS(1000)
**********    -Define the report layout-
DEF_REPORT    PRT_FILE(QSYSPRT)
DEF_HEAD      NAME(#HEAD01) FIELDS(#TEXT #PAGE . . . )
DEF_LINE      NAME(#MTXPRT) FIELDS(#MTXNAM #MTXVAL)
**********    -Set start *MTXT variable to blanks-
CHANGE        FIELD(#MTXVAR) TO(*BLANKS)
**********    -Get list of system variables-
USE           BUILTIN(GET_MULTVAR_LIST) WITH_ARGS(#MTXVAR)
              TO_GET(#MTXLST)
**********    -Process list-
SELECTLIST    NAME(#MTXLST)
**********    -Print *MTXT variables-
PRINT         LINE(#MTXPRT)
ENDSELECT
**********    -Close printer file-
ENDPRINT
```

GET_ML_VARIABLE

Category: Data Dictionary built-in functions

Description: Retrieves a multilingual variable definition.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Multilingual variable name	5	20		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code (OK, ER)	2	2		
2	N	Req	Length / Total digits	3	3	0	0
3	List	Req	Working list to contain multilingual definition information. The calling RDML function must provide a working list with an aggregate entry length of exactly 82 bytes. **Bytes Description** 1-4 Language code 5-82 Multilingual variable value.	82	82		

GET_SYSVAR_LIST

Category: Data Dictionary built-in functions

Description: Retrieves a list of system variables, their descriptions, programs and program types and returns them to the calling RDML function in a variable length working list.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Positioning system variable. The returned list starts with the first system variable whose name is greater than the value passed in this argument.	1	20		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain system variable information. The calling RDML function must provide a working list with an aggregate entry length of exactly 80 bytes. Each returned list entry is formatted as follows: **Bytes Description** 1-20 System variable name 21-60 System variable description 61-70 System variable program 71-73 Program type 74-80 << for future expansion >>	80	80		
2	A	Opt	Last system variable in list. Typically this value is used as the positioning argument on subsequent calls to this built-in function.	1	20		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
3	A	Opt	Return code. OK = list returned partially or completely filled with system variable details. No more system variables exist beyond those returned in the list. OV = list returned completely filled, but more system variables than could fit in the list exist. Typically used to indicate "more" system variables in page at a time style list displays NR = list was returned empty. Last system variable in the list is returned as blanks.	2	2		

Example

A user wants to print a list of all system variables.

```
DEF_LIST        NAME(#VARLST) FIELDS(#VARNAM #VARDES #VARPGM
                #VARTYP #SPARE)
                TYPE(*WORKING) ENTRYS(1000)
**********      -Define the report layout-
DEF_REPORT      PRT_FILE(QSYSPRT)
DEF_HEAD        NAME(#HEAD01) FIELDS(#TEXT #PAGE . . . )
DEF_LINE        NAME(#VARPRT) FIELDS(#VARNAM #VARDES #VARPGM #VARTYP)
**********      -Set start system variable to blanks-
CHANGE          FIELD(#STRVAR) TO(*BLANKS)
**********      -Get list of system variables-
USE             BUILTIN(GET_SYSVAR_LIST) WITH_ARGS(#STRVAR)
                TO_GET(#VARLST #LAST #RETCOD)
**********      -If return code is OK then process list-
IF              COND('#RETCOD *EQ OK')
SELECTLIST      NAMED(#VARLST)
**********      -Print system variables-
PRINT           LINE(#VARPRT)
ENDSELECT
**********      -Otherwise issue an error-
ELSE
MESSAGE         MSGTXT('An error has occurred. Report not produced.')
ENDIF
**********      -Close printer file-
ENDPRINT
```

GET_SYSTEM_VARIABLE

Category: Data Dictionary built-in functions

Description: Retrieves a system variable definition.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	**Type A/N**	**Req/ Opt**	**Description**	**Min Len**	**Max Len**	**Min Dec**	**Max Dec**
1	A	Req	System variable name	5	20		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code (OK, ER)	2	2		
2	A	Opt	Description	1	40		
3	A	Opt	STATIC or DYNAMIC	7	7		
4	A	Opt	Data type (ALPHA, NUMBER)	6	6		
5	N	Opt	Length / Total digits	3	3	0	0
6	N	Opt	Decimal positions	1	1	0	0
7	A	Opt	Evaluation program	10	10		
8	A	Opt	Ev. program type (FUN, 3GL)	3	3		

PUT_FIELD

Category: Data Dictionary built-in functions

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Description: Either inserts a new field into the LANSA data dictionary or updates details of an existing field.

Optionally this built-in function can present a prompt screen to the user that will allow details of a new or amended field to be further specified.

Argument values are exactly as per information input on the "Create new field definitions" screen described in the User Guide.

When a new field is being inserted into the dictionary, arguments that are not passed to the built-in function (or passed as null values) will adopt default values as described in the User Guide.

When an existing field is being updated in the data dictionary, arguments that are not passed to the built-in function (or passed as null values) will remain unchanged by the update operation.

When zero is input as the 'Number of Decimals' parameter it is treated as a null value. Use -1 in 'Number of Decimals' parameter to indicate a request to change the number of decimals of a field to zero.

If the copy validation checks option is used all checks from the sequence number specified are deleted, then the validation checks are copied from the 'from field'. Any reference to the 'from field' in copied validation checks are replaced by the name of the field being inserted/updated.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Prompt control details **Byte 1 - Prompt required** Y = prompt the user N = do not prompt the user **Byte 2 - EXIT/SYSTEM key** Y = enable EXIT/SYSTEM key N = do not enable EXIT/SYSTEM key **Byte 3 - MENU key** Y = enable MENU key N = do not enable MENU key	1	3		
2	A	Req	Name of field to be inserted or updated	1	10		
3	A	Opt	Field type A = alphanumeric S = signed decimal numeric P = packed decimal numeric	1	1		
4	N	Opt	Length of field or total number of digits in field.	3	30	0	0

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Note: For type A must be in range 1 - 256.				
			For type P or S must be in range 1 - 30.				
5	N	Opt	Number of decimal positions Not applicable to type A field	1	9	0	0
6	A	Opt	Reference field name	1	10		
7	A	Opt	Field description	1	40		
8	A	Opt	Field label	1	15		
9	A	Opt	Field column headings	1	60		
			List of 3 * A(20) headings				
			Bytes 1-20 are column head 1				
			Bytes 21-40 are column head 2				
			Bytes 41-60 are column head 3				
10	A	Opt	Output attributes list	1	40		
			List of 10 * A(4) attributes				
11	A	Opt	Input attributes list	1	40		
			List of 10 * A(4) attributes				
12	A	Opt	Edit code or edit word If first char is a quote (') then value is an edit word. Otherwise it is an edit code.	1	20		
			Not applicable to type A				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			field				
13	A	Opt	Default value of field	1	20		
14	A	Opt	Optional alias name of field	1	30		
15	A	Opt	System field flag YES = a system field NO = not a system field	3	3		
16	A	Opt	Initial public access Ignored for update operations	1	7		
17	A	Opt	Keyboard shift	1	1		
18	A	Opt	Prompting Process/Function The first 10 bytes are PROCESS name, the next 7 are FUNCTION name.	1	17		
19	A	Opt	(Re)Copy validation checks from data dictionary field	1	10		
20	N	Opt	Starting sequence for copy	1	3	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = field inserted/updated EX = prompt was terminated by EXIT/SYSTEM function key MU = prompt was terminated by MENU/CANCEL function key ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

PUT_FIELD_ML

Category: Data Dictionary built-in functions

Description: Puts/updates a list of field multilingual attributes in different languages.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Field name	1	10		
2	List	Req	Working list to contain language code and field multilingual attributes. The function must supply a working list with an aggregate entry length of exactly 119 bytes. Each list entry sent should be formatted as follows: **Bytes Description** 1-4 Language code 5-44 Field description	119	119		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			45-59 Field label 60-79 Field column heading 1 80-99 Field column heading 2 100-119 Field column heading 3				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = multilingual attributes added / updated to the database successfully. ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

PUT_HELP

Category: Data Dictionary built-in functions

Description: Puts/updates a list of help text for a specified field, function or process.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Object name The name of a field, function or process.	1	10		
2	A	Req	Object extension name If the object type is a function then this value should contain the name of the process in which the function is defined. If the object type is not a function then this value should be blank.	1	10		
3	A	Req	Object type Values: DF - Field	2	2		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			PD - Process PF - Function				
4	List	Req	Working list to contain help text. The calling RDML function must provide a working list with an aggregate entry length of exactly 77 bytes. Each list entry sent should be formatted as follows: **Bytes Description** 1-77 Help Text	1	77		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = help text put/updated to database successfully. ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

Example

A user wants to retrieve and update the help text of a specific object without going through the LANSA options provided on the "Process Control Menu" and the "Field Control Menu", that enables the user to create/change help text for fields, functions and processes.

```
*********   Define arguments and lists
DEFINE      FIELD(#OBJNAM) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#OBJEXT) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#OBJTYP) TYPE(*CHAR) LENGTH(2)
DEFINE      FIELD(#HLPTXT) TYPE(*CHAR) LENGTH(77)
DEFINE      FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2)
DEF_LIST    NAME(#WKHLPL) FIELDS((#HLPTXT)) TYPE(*WORKING)
DEF_LIST    NAME(#BWHLPL) FIELDS((#HLPTXT))
GROUP_BY    NAME(#RQSOBJ) FIELDS((#OBJNAM) (#OBJEXT) (#OBJTYP))
GROUP_BY    NAME(#DSPHLP) FIELDS((#OBJNAM) (#OBJEXT) (#OBJTYP))
*********   Clear working and browse lists
BEGIN_LOOP
CLR_LIST    NAMED(#WKHLPL)
CLR_LIST    NAMED(#BWHLPL)
*********   Request Object Name, Extension and Type
REQUEST     FIELDS(#RQSOBJ)
*********   Execute built-in-function - GET_HELP
USE         BUILTIN(GET_HELP) WITH_ARGS(#OBJNAM #OBJEXT #OBJTYP)
            TO_GET(#WKHLPL #RETCOD)
*********   Help text was retrieved successfully
IF          COND('#RETCOD *EQ ''OK''')
*********   Move Help text from the working list to the browselist
SELECTLIST  NAMED(#WKHLPL)
ADD_ENTRY   TO_LIST(#BWHLPL) WITH_MODE(*CHANGE)
ENDSELECT
*********   Allow Help text to be changed for the object
REQUEST     FIELDS(#DSPHLP) BROWSELIST(#BWHLPL)
*********   Change the help text for this object
EXECUTE     SUBROUTINE(PUTHELP)
*********   Working list overflowed, more help text to retrieve
```

```
ELSE
IF          COND('#RETCOD *EQ ''OV''')
MESSAGE     MSGTXT('List not big enough to fit all help text')
*********   GET_HELP failed with errors, report error
ELSE
MESSAGE     MSGTXT('GET_HELP failed with errors, try again')
ENDIF
ENDIF
END_LOOP
*********   Subroutine to change help text for this object
SUBROUTINE  NAME(PUTHELP)
CLR_LIST    NAMED(#WKHLPL)
*********   Move Help text from the browselist to the working list
SELECTLIST  NAMED(#BWHLPL)
ADD_ENTRY   TO_LIST(#WKHLPL)
ENDSELECT
*********   Execute built-in-function - PUT_HELP
USE         BUILTIN(PUT_HELP) WITH_ARGS(#OBJNAM #OBJEXT
            #OBJTYP #WKHLPL) TO_GET(#RETCOD)
*********   Help text was changed successfully
IF          COND('#RETCOD *EQ ''OK''')
MESSAGE     MSGTXT('Help text for this object has been changed')
*********   PUT_HELP failed with errors, report error
ELSE
MESSAGE     MSGTXT('PUT_HELP failed with errors, try again')
ENDIF
ENDROUTINE
```

PUT_ML_VARIABLE

Category: Data Dictionary built-in functions

Description: Adds/Updates a multilingual variable definition. to the data dictionary.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Multilingual variable name	5	20		
2	N	Req	Length / Total digits	1	3	0	0
3	List	Req	Working list to contain multilingual definition information. The calling RDML function must provide a working list with an aggregate entry length of exactly 82 bytes. **Bytes Description** 1-4 Language code 5-82 Multilingual variable value.	82	82		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code (OK, ER)	2	2		

PUT_SYSTEM_VARIABLE

Category: Data Dictionary built-in functions

Description: Creates / amends a system variable. If the system variable name specified does not already exist the system variable is added, if it does exist the system variable definition is updated with the new details.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	System variable name A System variable name must begin with '*'.	5	20		
2	A	Req	Description	1	40		
3	A	Req	STATIC or DYNAMIC	6	7		
4	A	Req	Data type (ALPHA, NUMBER)	5	6		
5	N	Req	Length / Total digits	3	3	0	0
6	N	Req	Decimal positions	1	1	0	0
7	A	Req	Evaluation program	1	10		
8	A	Req	Ev. program type (FUN,	3	3		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			3GL)				
9	A	Req	Initial public access (ALL, NORMAL or NONE)	3	6		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code (OK, ER)	2	2		

Example

A small program to allow the creation / amendment of system variables. The user is requested to fill in the system variable details and a message based on the return code notifies if the operation was successful.

```
GROUP_BY     NAME(#SYSVAR) FIELDS(#SYSNAM #SYSDES #SYSSOD
             #SYSTYP #SYSLEN #SYSDEC #SYSPGM #PGMTYP #ACCESS)
*********    Set some defaults
CHANGE       #SYSVAR *NULL
CHANGE       #SYSSOD 'DYNAMIC'
CHANGE       #SYSTYP 'ALPHA '
CHANGE       #PGMTYP 'FUN'
CHANGE       #ACCESS 'NORMAL'
*********    Request System variable details
REQUEST      FIELDS(#SYSVAR)
*********
USE          BUILTIN(PUT_SYSTEM_VARIABLE)
             WITH_ARGS(#SYSNAM #SYSDES #SYSSOD #SYSTYP #SYSLEN
             #SYSDEC #SYSPGM #PGMTYP #ACCESS)
             TO_GET(#RETCOD)
```

```
**********  Inform user of success / failure
IF          '#RETCOD *EQ OK'
MESSAGE     MSGF(SYSMSGS) MSGID(SYS0023) MSGDTA(#SYSNAM)
ELSE
MESSAGE     MSGF(SYSMSGS) MSGID(SYS0024) MSGDTA(#SYSNAM)
< -------   Handle any errors  ------- >
ENDIF
```

Chapter 2. File Related Built-In Functions

ACCESS_RTE

Category: File related built in functions

Description: Specifies or re-specifies the attributes of an "access route" between the definition of the file being edited and another file defined within the LANSA system.

For details of what an "access route" is and how they are used by the LANSA system refer to the User Guide.

After using this built-in function to define the basic access route attributes, repetitively use the ACCESS_RTE_KEY built-in function to specify or re-specify the route key field(s) or value(s).

An edit session must be commenced by using the START_FILE_EDIT built-in function prior to using this built-in function.

Allowable argument values and adopted default values are as per the LANSA "Add Access Route" facility which is described in the User Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of access route First 3 letters of the name must be the same as the edit "source" nominated in the START_FILE_EDIT built-in function.	1	10		
2	A	Req	Description of access route	1	40		
3	A	Req	File to be accessed via	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			route				
4	A	Req	Library in which file resides *FIRST and *DEFAULT are allowable.	1	10		
5	N	Req	Maximum records expected Must be in range 1 - 9999.	1	4	0	0
6	A	Opt	Action to take if no records found via this route. Must be ABORT, IGNORE, N/AVAIL or DUMMY.	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = access route defined ER = error detected In case of "ER" return code error message(s) are issued automatically and the edit session continues. The access route that caused the error is ignored in this and all subsequent requests during the edit session.	2	2		

ACCESS_RTE_KEY

Category: File related built-in functions

Description: Specifies or re-specifies the name of a field or value that is to be used to access data via an access route previously defined via the ACCESS_RTE built-in function.

An edit session must be commenced by using the START_FILE_EDIT built-in function prior to using this built-in function.

Allowable argument values and adopted default values are as per the LANSA "Add Access Route" facility which is described in the User Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Access route name	1	10		
2	A	Req	Name of field from file being edited or a literal value that is to be used to form the key used to access data via the access route.	1	20		
3	N	Opt	Optional sequencing number. Used to sequence key fields. If not specified keys are sequenced in the same order as they are presented.	1	5	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = access key defined ER = error detected In the case of "ER" return code error message(s) are issued automatically and the edit session continues. The access route that caused the error is ignored in this and all subsequent requests during the edit session	2	2		

DLT_FILE

Category: File related built-in functions

Description: Submits a batch job to delete a file and its associated logical files and I/O module.

Argument values are almost exactly as per information input on the "Delete a file from the System" screen described in the User Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	File name	1	10		
2	A	Req	Library name	1	10		
3	A	Opt	Name of batch job Default: File name	1	10		
4	A	Opt	Name of job description Default: the job description from the requesting job's attributes.	1	21		
5	A	Opt	Name of job queue Default: the job queue	1	21		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			from the requesting job's attributes.				
6	A	Opt	Name of output queue Default: the output queue from the requesting job's attributes.	1	21		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = successful submission ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

Example

A user wants to control the deletion of files and associated logical views and I/O module using their own version of the "Delete a file from the System" facility.

```
*********   Define arguments and lists
DEFINE      FIELD(#FILNAM) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#LIBNAM) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2)
BEGIN_LOOP
```

```
*********    Request File and library name
REQUEST      FIELDS(#FILNAM #LIBNAM)
*********    Execute built-in-function - DLT_FILE
USE          BUILTIN(DLT_FILE) WITH_ARGS(#FILNAM #LIBNAM)
             TO_GET(#RETCOD)
*********    Check if submission was successful
IF           COND('#RETCOD *EQ ''OK''')
MESSAGE      MSGTXT('Delete of file submitted successfully')
CHANGE       FIELD(#FILNAM) TO(*BLANK)
ELSE
MESSAGE      MSGTXT('Delete submit failed with errors,
                     refer to additional messages')
ENDIF
END_LOOP
```

END_FILE_EDIT

Category: File related built-in functions

Description: Ends an "edit session" on the definition of a nominated LANSA file definition previously started by the START_FILE_EDIT built-in function.

The edit session may have been used to define a new file or alter an existing one.

The file definition is released by this built-in function so that it can be accessed by other users.

A number of checks that relate to prior actions via the LOGICAL_KEY and ACCESS_RTE_KEY built-in functions are performed via this function. These may result in the abandonment of the edit session, and an "ER" return code being returned.

Additionally, warning messages may be issued by this built-in function. In this case the return code will still be returned as "OK".

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Commit edited details flag Y = commit details N = drop edited details	1	1		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = edit session ended ER = fatal error detected In case of "ER" return code error message(s) are issued automatically and the edit session ended without commitment	2	2		

FILE_FIELD / VIRTUAL

Category: File related Built-In Functions

Description: FILE_FIELD specifies or re-specifies a field that is part of the record format of the file definition being edited.

FILE_FIELD_VIRTUAL specifies or re-specifies a virtual field that is part of the definition of the file being edited.

An edit session must be commenced by using the START_FILE_EDIT built-in function prior to using this built-in function.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

WARNING: The FILE_FIELD built-in function cannot be used for a file of type "OTHER".

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of real or virtual field. Must be defined in the LANSA data dictionary.	1	10		
2	N	Opt	Optional sequencing number. Used to order fields within the file record format. If not specified fields are sequenced in the same order as they are presented.	1	5	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = field added to file ER = fatal error detected In case of "ER" return code error message(s) are issued automatically and the edit session ended without commitment	2	2		

GET_FILE_INFO

Category: File related built-in functions

Description: Retrieves a list of file related information from the LANSA internal database and returns it to the calling RDML function in variable length working lists.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built-in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built-in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Physical file name	1	10		
2	A	Req	Physical file library	1	10		
3	A	Req	Type of file related information to retrieve. Valid types are: FIELDS- Fields in the file PHYKEYS- Fields used as keys to the file LGLVIEWS- Logical views for the file ACCROUTES- Access routes for the file	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			MLATTR- Multilingual attributes				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = list returned partially or completely filled. No more of this type of information exists for this file. OV = list returned completely filled, but more of this type of information than could fit in the list exists. NR = list was returned empty. Last entry in the list is returned as null. ER = File not found. Last entry in the list is returned as null.	2	2		
2	List	Req	Header working list to contain file related information. The calling RDML function must provide a working list with an aggregate entry length of exactly 100 bytes.	100	100		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Bytes Description 1-5 Number of the first entry in the detail list for this entry in character format. A value of '00000' denotes that there are no entries in the detail list for this entry. 6-10 Number of the last entry in the detail list for this entry in character format 11-100 Rest of information **<u>For type FIELDS:</u>** One header list entry is formatted as follows: **Bytes Description** 1-5 As above 6-10 as above **<u>For type PHYKEYS</u>**: One header list entry is formatted as follows: **Bytes Description** 1-5 As above 6-10 As above **<u>For Type LGLVIEWS:</u>** Each header list entry is formatted as follows:				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			1-5 As above 6-10 As above 11-20 Logical view name 21-60 Logical view description **For type ACCROUTES :** Each header list entry is formatted as follows: **Bytes Description** 1-5 as above 6-10 as above 11-20 Access route name 21-60 Access route description 61-70 File accessed **For type MLATTR:** Each header list entry is formatted as follows: **Bytes Description** 1-5 As above 6-10 As above 11-20 Logical view name or physical file name				
3	List	Req	Detail working list to	50	50		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			contain file related information. The calling RDML function must provide a working list with an aggregate entry length of exactly 50 bytes. **<u>For type FIELDS:</u>** Each detail list entry is formatted as follows: **Bytes Description** 1-10 Field name that is part of the file **<u>For type PHYKEYS:</u>** Each detail list entry is formatted as follows: **Bytes Description** 1-10 Field name that is part of the file key **<u>For type LGLVIEWS:</u>** Each detail list entry is formatted as follows: **Bytes Description** 1-10 Field name that is part of the logical view key **<u>For type ACCROUTES :</u>** Each detail list entry is formatted as follows:				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Bytes Description 1-20 Value that is used as a key in the access route. **For type MLATTR:** Each detail list entry is formatted as follows: **Bytes Description** 1-4 Language code 5-44 Logical view or physical file description				

GET_LOGICAL_LIST

Category: File related built-in functions

Description: Retrieves a list of physical files associated logical views and their descriptions from the data dictionary and returns them to the calling RDML function in a variable length working list.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Physical file name.	1	10		
2	A	Req	Physical file library.	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain logical file information. The calling RDML function must provide a working list with an	70	70		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			aggregate entry length of exactly 70 bytes. Each returned list entry is formatted as follows: **Bytes Description** 1-10 Logical file 11-20 Logical file library 21-60 Description 61-70 <<future expansion>>				
2	A	Opt	Return code. OK = list returned partially or completely filled with file details. No more logicals exist for this physical file. OV = list returned completely filled, but more files than could fit in the list exist. Typically used to indicate "more" fields in page at a time style list displays. NR = list was returned empty. Last file in the list is returned as blanks. ER = Physical file not found. Last file in the list is returned as blanks.	2	2		

GET_PHYSICAL_LIST

Category: File related built-in functions

Description: Retrieves a list of physical files and their descriptions from the data dictionary and returns them to calling RDML function in a variable length working list.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Positioning file value. The returned list starts with the first file from the dictionary whose name is greater than the value passed in this argument.	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain File information. The calling RDML function	70	70		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			must provide a working list with an aggregate entry length of exactly 70 bytes. Each returned list entry is formatted as follows: **Bytes Description** 1-10 Physical file name 11-20 Physical file library 21-60 Description 61-70 <<future expansion>>				
2	A	Opt	Last file in returned list Typically this value is used as the positioning argument on subsequent calls to this built-in function.	1	10		
3	A	Opt	Return code. OK = list returned partially or completely filled with file details. No more files exist beyond those returned in the list. OV = list returned completely filled, but more files than could fit in the list exist. Typically used to indicate "more" files in page at a time style list displays.	2	2		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			NR = list was returned empty. Last file in the list is returned as blanks.				

Example

This function could be used to write a program that allows a site to modify an existing LANSA database.

```
DEF_LIST      NAME(#FILLST) FIELDS(#FILNAM #FILLIB #FILDES #SPARE)
              TYPE(*WORKING) ENTRYS(10)
DEF_LIST      NAME(#FILDSP) FIELDS((#SELECTOR *SEL) #FILNAM #FILLIB
              #FILDES)
**********    -Clear lists-
CLR_LIST      NAMED(#FILLST)
CLR_LIST      NAMED(#FILDSP)
**********    -Request file to start from in list-
REQUEST       FIELDS(#STRTFL) TEXT(('File to start from' 5 5))
**********    -Get the list of files-
USE           BUILTIN(GET_PHYSICAL_LIST) WITH_ARGS(#STRTFL)
              TO_GET(#FILLST #LAST #RETCOD)
**********    -If records found-
IF            COND('(#RETCOD *EQ OK) *OR (#RETCOD *EQ OV)')
SELECTLIST    NAMED(#FILLST)
ADD_ENTRY     TO_LIST(#FILDSP)
ENDSELECT
**********
DISPLAY       BROWSELIST(#FILDSP)
**********    -Process selected records-
SELECTLIST    NAMED(#FILDSP) GET_ENTRYS(*SELECT)
EXECUTE       SUBROUTINE(FILE_EDIT)
```

```
ENDSELECT
ELSE
MESSAGE         MSGTXT('No files found .... Program ended')
RETURN
ENDIF
```

LOAD_OTHER_FILE

Category: File related built-in functions

Description: Loads the definition of an "OTHER" file.

An edit session must be commenced by using the START_FILE_EDIT built-in function prior to using this built-in function.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Maximum number of logical files to load Default: 5	1	2	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code (OK, ER) In case of "ER" return code error message(s) are issued automatically.	2	2		

Example

A user wants to control the load of an "OTHER" file definition using their own version of the 'Load an "OTHER" file' option. A maximum of 2 logicals to load has been specified in this example.

```
********** Define arguments and lists
DEFINE     FIELD(#FILNAM) TYPE(*CHAR) LENGTH(10)
DEFINE     FIELD(#LIBNAM) TYPE(*CHAR) LENGTH(10)
DEFINE     FIELD(#MAXLOG) TYPE(*DEC) LENGTH(2) DECIMALS(0)
DEFINE     FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2)
BEGIN_LOOP
********** Request File and library name and maximum number of
********** logicals to load
REQUEST    FIELDS(#FILNAM #LIBNAM #MAXLOG)
**********
USE        BUILTIN(START_FILE_EDIT)
           WITH_ARGS(#FILNAM #LIBNAM 'DEM')
           TO_GET(#RETCODE)
USE        BUILTIN(LOAD_OTHER_FILE) WITH_ARGS(2) TO_GET(#RETCOD)
USE        BUILTIN(END_FILE_EDIT) ('Y')
********** Submit job to make file operational
USE        BUILTIN(MAKE_FILE_OPERATIONL) WITH_ARGS(#FILNAM
           #LIBNAM) TO_GET(#RETCOD)
**********
END_LOOP
```

LOGICAL_KEY

Category: File related built-in functions

Description: Specifies or re-specifies the name of a field that is a key of a logical view / file previously defined by the LOGICAL_VIEW built-in function.

An edit session must be commenced by using the START_FILE_EDIT built-in function prior to using this built-in function.

Allowable argument values and adopted default values are as per the LANSA "Create Logical View" facility which is described in the User Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

WARNING: This built-in function cannot be used for a file of type "OTHER".

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of logical view to which key is to belong.	1	10		
2	A	Req	Name of key field. Must have been previously specified as a field in the file by using the FILE_FIELD built-in function.	1	10		
3	N	Opt	Optional sequencing number. Used to	1	5	0	0

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			sequence key fields. If not specified keys are sequenced in the same order as they are presented.				
4	A	Opt	Ascending or Descending key. Must be A or D. Default is A.	1	1		
5	A	Opt	Signed, Unsigned or Absolute value ordering of numeric key. Must be S,U or A. Default is U for alphas and S for numerics.	1	1		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = key defined ER = fatal error detected In case of "ER" return code error message(s) are issued automatically and the edit session ended without commitment.	2	2		

LOGICAL_VIEW

Category: File related built-in functions

Description: Specifies or re-specifies the name and basic attributes of a logical view / file that is to base on the file definition being edited.

An edit session must be commenced by using the START_FILE_EDIT built-in function prior to using this built-in function.

After using this built-in function to define the basic logical view / file attributes, repetitively use the LOGICAL_KEY built-in function to specify or re-specify the key field name(s).

Allowable argument values and adopted default values are as per the LANSA "Create Logical View" facility which is described in the User Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

WARNING: This built-in function cannot be used for a file of type "OTHER".

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of logical view.	1	10		
2	A	Req	Description of logical view	1	40		
3	A	Opt	Access path maintenance option Must be IMMED or DELAY. Default is IMMED.	1	7		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
4	A	Opt	Uniquely keyed file / view Must be YES or NO. Default is NO.	1	3		
5	A	Opt	Dynamic record selection Must be Y or N. Default is N. Must be YES or NO. Default is NO.	1	3		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = logical view defined ER = fatal error detected In case of "ER" return code error message(s) are issued automatically and the edit session ended without commitment.	2	2		

MAKE_FILE_OPERATIONL

Category: File related built-in functions

Description: Submits a batch job to create or recreate a file plus associated logical files and I/O module.

Argument values are almost exactly as per information input on the "Create / Re-Create a File" screen described in the User Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	File name	1	10		
2	A	Req	Library name	1	10		
3	A	Opt	Recreate options if file is already created **Byte 1 - Recreate physical file** Y = recreate the physical file N = do not recreate the physical file Default: Y	1	3		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Byte 2 - Recreate logical files				
			Y = recreate logical files				
			N = do not recreate logical files				
			Default: Y				
			Byte 3 - Recreate I/O module				
			Y = recreate I/O module				
			N = do not recreate I/O module				
			Default: Y				
4	A	Opt	Name of batch job	1	10		
			Default: File name				
5	A	Opt	Name of job description	1	21		
			Default: the job description from the requesting job's attributes.				
6	A	Opt	Name of job queue	1	21		
			Default: the job queue from the requesting job's attributes.				
7	A	Opt	Name of output queue	1	21		
			Default: the output queue from the requesting job's attributes.				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
8	A	Opt	Produce file and I/O module source listings ? Y = produce listings N = do not produce listings Default :N (do not produce listings)	1	1		
9	A	Opt	Ignore decimal data error in associated I/O module? Y = ignore decimal data errors N = do not ignore errors Default: N (do not ignore errors)	1	1		
10	A	Opt	Strip debug data options in associated I/O module? Y = debugging information should be stripped. N = debugging information should not be stripped. Default: Y (debugging information should be stripped)	1	1		
11	A	Opt	User program to call Default: Blank	1	21		
12	A	Opt	Delete $$ File?	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Y = $$ version of file should be deleted. N = $$ version of file should not be deleted. Default: N ($$ version of file should not be deleted)				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = successful submission ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

Example

A user wants to control the compilation of files and associated logical views and I/O module using their own version of the "Create / Re-Create a File" facility.

```
*********   Define arguments and lists
DEFINE      FIELD(#FILNAM) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#LIBNAM) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2)
BEGIN_LOOP
*********   Request File and library name
REQUEST     FIELDS(#FILNAM #LIBNAM)
*********   Execute built-in-function - MAKE_FILE_OPERATIONL
USE         BUILTIN(MAKE_FILE_OPERATIONL) WITH_ARGS(#FILNAM
            #LIBNAM) TO_GET(#RETCOD)
*********   Check if submission was successful
IF          COND('#RETCOD *EQ ''OK''')
MESSAGE     MSGTXT('Create/recreate of file submitted successfully')
CHANGE      FIELD(#FILNAM) TO(*BLANK)
ELSE
MESSAGE     MSGTXT('Create/recreate submit failed with errors,
                      refer to additional messages')
ENDIF
END_LOOP
```

PHYSICAL_KEY

Category: File related built-in functions

Description: Specifies or re-specifies the name of a field that is a key of the physical file associated with the file definition being edited.

An edit session must be commenced by using the START_FILE_EDIT built-in function prior to using this built-in function.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

WARNING: This built-in function cannot be used for a file of type "OTHER".

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of key field. Must have been previously specified as a field in the file by using the FILE_FIELD built-in function.	1	10		
2	N	Opt	Optional sequencing number. Used to sequence key fields. If not specified keys are sequenced in the same order as they are presented.	1	5	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = key defined ER = fatal error detected In case of "ER" return code error message(s) are issued automatically and the edit session ended without commitment.	2	2		

PUT_FILE_ML

Category: File related built-in functions

Description: Puts/updates a list of file multilingual attributes in different languages.

An edit session must be commenced by using the START_FILE_EDIT built-in function prior to using this built-in function.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used **must** carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Physical file or logical view name.	1	10		
2	List	Req	Working list to contain language code and file multilingual attributes. The function must supply a working list with an aggregate entry length of exactly 44 bytes. Each list entry sent should be formatted as follows:	44	44		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Bytes Description 1-4 Language code 5-44 Physical file or logical view description				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = multilingual attributes added / updated to the database successfully. ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

SET_FILE_ATTRIBUTE

Category: File related built-in functions

Description: Sets a file's database attributes.

An edit session must be commenced by using the START_FILE_EDIT built-in function prior to using this built-in function.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built-in function can be used in conjunction with other data dictionary built-in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built-in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Attribute to set Valid attributes . . . I/O module: 'IOMODULE=YES' 'IOMODULE=NO ' OS/400 High Speed Table: 'OS400HST=YES' 'OS400HST=NO '	1	256		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code (OK, ER) In case of "ER" return code error message(s) are issued automatically.	2	2		

NOTE: Currently this built-in function can only be used to determine whether or not a file will have an I/O module. This facility will be extended at a later date to include other database attributes.

Example

A LANSA function to emulate the 'File definition Menu' has been written. When a certain option is taken the user can decide to set a file attribute. IE Do you want an I/O module (Yes/No) ?

```
********** Define arguments and lists
DEFINE     FIELD(#FILNAM) TYPE(*CHAR) LENGTH(10)
DEFINE     FIELD(#LIBNAM) TYPE(*CHAR) LENGTH(10)
DEFINE     FIELD(#YESNO) TYPE(*CHAR) LENGTH(32) LABEL('I/O Module')
DEFINE     FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2)
BEGIN_LOOP
********** Request File and library name and I/O module attribute
REQUEST    FIELDS(#FILNAM #LIBNAM #YESNO)
**********
USE        BUILTIN(START_FILE_EDIT) WITH_ARGS(#FILNAM #LIBNAM 'DEM')
           TO_GET(#RETCODE)
IF         COND('#YESNO *EQ YES')
USE        BUILTIN(SET_FILE_ATTRIBUTE) WITH_ARGS('''IOMODULE=YES''')
           TO_GET(#RETCOD)
ELSE
```

```
USE         BUILTIN(SET_FILE_ATTRIBUTE) WITH_ARGS('''IOMODULE=NO ''')
            TO_GET(#RETCOD)
ENDIF
USE         BUILTIN(END_FILE_EDIT) WITH_ARGS('Y') TO_GET(#RETCOD)
*********** Submit job to make file operational
USE         BUILTIN(MAKE_FILE_OPERATIONL) WITH_ARGS(#FILNAM #LIBNAM)
            TO_GET(#RETCOD)
**********
END_LOOP
```

START_FILE_EDIT

Category: File related built-in functions

Description: Starts an "edit session" on the definition of a nominated LANSA file definition.

The edit session can be used to define a new file or alter an existing one.

The file **definition** is locked for exclusive use throughout the edit session.

Only one file definition can be edited at one time (ie: it is not possible to concurrently edit 2 file definitions from within the same job).

Details of the new or amended file definition will be lost unless the END_FILE_EDIT built-in function is used to "commit" them.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of file to be edited	1	10		
2	A	Req	Library in which file resides *FIRST is acceptable when editing an existing file definition. *DEFAULT is acceptable when editing an existing file definition or creating a new one.	1	10		
3 **	A	Req	Source of edited details.	3	3		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Must not be blank, LAN or OTH.				
4	A	Opt	File description Must not be blank. Required for a new file.	1	40		
5	A	Opt	Initial public access. Required for a new file. ALL, NORMAL or NONE allowed.	1	6		
6	A	Opt	File component edit options Byte 1: Y or N indicates that fields are to be edited. Default value is Y. Byte 2: Y or N indicates that logical views/files are to be edited. Default value is Y. Byte 3: Y or N indicates that access route details are to be edited. Default value is Y. When a byte is passed as N, the associated component of the file definition remains unchanged during the edit session and is not flagged for pending deletion as described below. Any attempt to edit that component of the file definition will cause a fatal error.	3	3		

** The "source" of the edited details is vital. When an edit session is commenced all details of the file definition that have the same "source" as that passed to the START_FILE_EDIT built-in function are flagged for pending deletion. If the details are not "re-specified" by one of the FILE_FIELD, LOGICAL_VIEW, etc built-in functions, they are deleted from the file definition by the END_FILE_EDIT built-in function. The exception is those that have been specifically excluded from editing by using one or more of the byte positions in the 6th argument previously described. This allows for file details specified by other "sources" (such as direct input via the LANSA "Review or change a file definition" facilities) to remain intact during a file edit session.

Examples of this "source" code would be:

LDM	LANSA data modeling interface
IEW	Information engineering workbench interface
ACC	Accelerate data modeling interface

Once set, the source code used should never be changed within a particular type of interface.

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = edit session commenced ER = fatal error detected In case of "ER" return code error message(s) are issued automatically and the edit session ended without commitment.	2	2		

Example

The following example defines all details of a simple name and address file called NAMES to the LANSA system by using built-in functions rather than the conventional menu driven interfaces.

```
*** Define the fields into the data dictionary (no prompting)

USE     BUILTIN(PUT_FIELD) WITH_ARGS('N' 'CUSTNO' 'S' 007 0 ' '
        'Customer number') TO_GET(#RETCODE)
USE     BUILTIN(PUT_FIELD) WITH_ARGS('N' 'CUSNAME' 'A' 010 0 ' '
        'Customer name') TO_GET(#RETCODE)
USE     BUILTIN(PUT_FIELD) WITH_ARGS('N' 'ADDRESS1' 'A' 020 0 ' '
        'Address line 1') TO_GET(#RETCODE)
USE     BUILTIN(PUT_FIELD) WITH_ARGS('N' 'ADDRESS2' 'A' 020 0 ' '
        'Address line 2') TO_GET(#RETCODE)
USE     BUILTIN(PUT_FIELD)WITH_ARGS('N' 'ZIPCODE' 'S' 006 0 ' '
        'Zip code') TO_GET(#RETCODE)

*** Start an edit session on the new file NAMES in library QGPL

USE     BUILTIN(START_FILE_EDIT) WITH_ARGS('NAMES' 'QGPL' 'DEM'
          'Customer details' 'NORMAL') TO_GET(#RETCODE)

*** Define the fields in the file
USE     BUILTIN(FILE_FIELD) WITH_ARGS('CUSTNO') TO_GET(#RETCODE)
USE     BUILTIN(FILE_FIELD) WITH_ARGS('CUSNAME') TO_GET(#RETCODE)
USE     BUILTIN(FILE_FIELD) WITH_ARGS('ADDRESS1') TO_GET(#RETCODE)
USE     BUILTIN(FILE_FIELD) WITH_ARGS('ADDRESS2') TO_GET(#RETCODE)
USE     BUILTIN(FILE_FIELD) WITH_ARGS('ZIPCODE') TO_GET(#RETCODE)
*** Define the primary or relational file key

USE     BUILTIN(PHYSICAL_KEY) WITH_ARGS('CUSTNO') TO_GET(#RETCODE)

*** Define additional logical view in CUSNAME / ZIPCODE order
USE     BUILTIN(LOGICAL_VIEW) WITH_ARGS('NAMESV1' 'Customers in
        name order') TO_GET(#RETCODE)
```

```
*** Define keys of logical view NAMESV1
USE      BUILTIN(LOGICAL_KEY) WITH_ARGS('NAMESV1' 'CUSNAME')
         TO_GET(#RETCODE)
USE      BUILTIN(LOGICAL_KEY) WITH_ARGS('NAMESV1' 'ZIPCODE')
         TO_GET(#RETCODE)

*** Define "one to one" access route to ZIPTABLE by using key ZIPCODE
USE      BUILTIN(ACCESS_RTE) WITH_ARGS('DEM1' 'Zip details'
        'ZIPTABLE' '''*FIRST''' 1 'N/AVAIL') TO_GET(#RETCODE)
USE      BUILTIN(ACCESS_RTE_KEY) WITH_ARGS('DEM1' 'ZIPCODE')
         TO_GET(#RETCODE)

*** Define "one to many" access route to ORDHDRV2 using key CUSTNO
USE      BUILTIN(ACCESS_RTE) WITH_ARGS('DEM2' 'Order details'
         'ORDHDRV2' '''*FIRST''' 999 'IGNORE') TO_GET(#RETCODE)
USE      BUILTIN(ACCESS_RTE_KEY) WITH_ARGS('DEM2' 'CUSTNO')
         TO_GET(#RETCODE)

*** End the edit session and commit details
USE      BUILTIN(END_FILE_EDIT) WITH_ARGS('Y') TO_GET(#RETCODE)
```

Chapter 3. Rule/Trigger Related Built-In Functions

DELETE_CHECKS

Category: Validation related built in functions

Description: Deletes standard DICTIONARY or FILE level validation checks from a nominated field for subsequent replacement by PUT_XXXXXXX validation check built in functions.

When deleting FILE level validation checks from a field, the file involved must have been previously placed into an edit session by the START_FILE_EDIT built in function.

Normal authority and task tracking rules apply to the use of this built in function.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Level of validation checks that are to be deleted. D = Dictionary level F = File level	1	1		
2	A	Req	Name of field in dictionary or file from which validation rules are to be deleted.	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
3 *	N	Req	Sequence number to control deletion. Only checks with a sequence number greater than or equal to this value are deleted. If this argument is not specified, a value of zero (0) is assumed, so all checks will match this control value.	1	3	0	0
4 *	A	Req	Generic description of check used to control deletion. Only checks which have a description generically matching this value will be deleted. If this value is not specified, a default value of blanks is assumed, so all checks will match this control value.	1	30		

* The deletion control sequence number and description are related by an "AND" relationship. So if you pass values of 500 and 'IEW', only checks that have a sequence number greater than or equal to 500 and a description that starts with 'IEW' will be deleted.

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = validation check defined ER = fatal error detected	2	2		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			NR = no records found eligible for deletion In case of "ER" return code error message(s) are issued automatically. When a file edit session is involved it is ended automatically without commitment.				

Example

A user wants to delete validation checks for a specific field, without going through the LANSA options provided on the "Field Control Menu" that enables the user to delete validation checks.

```
********* Define arguments and lists
DEFINE    FIELD(#LEVEL) TYPE(*CHAR) LENGTH(1) LABEL('Level')
DEFINE    FIELD(#FIELD) TYPE(*CHAR) LENGTH(10) LABEL('Field')
DEFINE    FIELD(#SEQNUM) TYPE(*DEC) LENGTH(3) DECIMALS(0)
          LABEL('Sequence #')
DEFINE    FIELD(#DESCR) TYPE(*CHAR) LENGTH(30) LABEL('Description')
DEFINE    FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2) LABEL('Return code')
GROUP_BY  NAME(#VALCHK) FIELDS((#LEVEL) (#FIELD) (#SEQNUM) (#DESCR))
********* Request Validation check details
BEGIN_LOOP
REQUEST   FIELDS(#VALCHK)
********* Execute built-in-function - DELETE_CHECKS
USE       BUILTIN(DELETE_CHECKS) WITH_ARGS(#LEVEL #FIELD #SEQNUM
          #DESCR) TO_GET(#RETCOD)
********* Deletion of validation checks was successful
IF        COND('#RETCOD *EQ ''OK''')
MESSAGE   MSGTXT('Deletion of validation check(s) was successful')
********* Deletion of validation checks failed
```

```
ELSE
IF         COND('#RETCOD *EQ ''ER''')
MESSAGE    MSGTXT('Deletion of validation check(s) failed')
********** No records found eligible for deletion
ELSE
IF         COND('#RETCOD *EQ ''NR''')
MESSAGE    MSGTXT('No Records found eligible for deletion')
ENDIF
ENDIF
ENDIF
END_LOOP
```

DELETE_TRIGGERS

Category: Validation related built in functions

Description: Deletes standard DICTIONARY or FILE level triggers from a nominated field for subsequent replacement by the PUT_TRIGGER built in function.

When deleting FILE level triggers from a field, the file involved must have been previously placed into an edit session by the START_FILE_EDIT built in function.

Normal authority and task tracking rules apply to the use of this built in function.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Level of triggers that are to be deleted. D = Dictionary level F = File level	1	1		
2	A	Req	Name of field in dictionary or file from which triggers are to be deleted.	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
3 *	N	Req	Sequence number to control deletion. Only triggers with a sequence number greater than or equal to this value are deleted. If this argument is not specified, a value of zero (0) is assumed, so all triggers will match this control value.	1	3	0	0
4 *	A	Req	Generic description of trigger used to control deletion. Only checks which have a description generically matching this value will be deleted. If this value is not specified, a default value of blanks is assumed, so all checks will match this control value.	1	30		

* The deletion control sequence number and description are related by an "AND" relationship. So if you pass values of 500 and 'IEW', only triggers that have a sequence number greater than or equal to 500 and a description that starts with 'IEW' will be deleted.

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = validation check defined ER = fatal error detected NR = no records found eligible for deletion In case of "ER" return code error message(s) are issued automatically. When a file edit session is involved it is ended automatically without commitment.	2	2		

PUT_COND_CHECK

Category: Validation related built in functions

Description: Create/amend a "simple conditional logic" DICTIONARY or FILE level validation check into the data dictionary or file definition of the nominated field.

When adding a FILE level validation check to a field, the file involved must have been previously placed into an edit session by the START_FILE_EDIT built in function.

All argument values passed to this built in function are validated exactly as if they had been entered through the online validation check definition screen panels.

Normal authority and task tracking rules apply to the use of this built in function.

For more information on "field validation checks", refer to the "field validation checks" section within the "Field Control" chapter in the LANSA Users Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Level of validation check. D = Dictionary level	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			F = File level				
2	A	Req	Name of field in dictionary to which validation rule is to be applied.	1	10		
3	N	Req	Sequence number of check.	1	3	0	0
4	A	Req	Description of check.	1	30		
5	A	Req	Enable check for ADD.	1	1		
			Y = Check performed on ADD				
			U = Check performed on ADDUSE				
			N = Check not performed on ADD				
6	A	Req	Enable check for CHANGE.	1	1		
			Y = Check performed on CHG				
			U = Check performed on CHGUSE				
			N = Check not performed on CHG				
7	A	Req	Enable check for DELETE.	1	1		
			Y = Check performed on DLT				
			N = Check not performed on DLT				
8	A	Req	Action if check is true.	4	6		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			NEXT = Perform next check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.				
9	A	Req	Action if check is false. NEXT = Perform next check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.	4	6		
10	A	Req	Message file details. Details of error message to be issued from a message file. Message file details should be formatted as follows: **Bytes Description** 1-7 Error Message Number 8-17 Message File Name 18-27 Message File Library If message text is used, pass this argument as blanks.	27	27		
11	A	Req	Message text.	1	80		
12	List	Req	Working list to contain the condition that is to be	1	20		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			evaluated for the simple logic check. The calling RDML function must provide a working list with an aggregate entry length of exactly 79 bytes and exactly 5 condition line entries. Each list entry sent should be formatted as follows: **Bytes Description** 1-79 Condition line				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = validation check defined ER = fatal error detected In case of "ER" return code error message(s) are issued automatically. When a file edit session is involved it is ended automatically without commitment.	2	2		

Example

A user wants to put a "simple conditional logic" validation check for a specific field, without going through the LANSA options provided on the "Field Control Menu" that enables the user to put a "simple conditional logic" validation check.

```
*********  Define arguments and lists
DEFINE     FIELD(#LEVEL) TYPE(*CHAR) LENGTH(1) LABEL('Level')
DEFINE     FIELD(#FIELD) TYPE(*CHAR) LENGTH(10) LABEL('Field')
DEFINE     FIELD(#SEQNUM) TYPE(*DEC) LENGTH(3) DECIMALS(0)
           LABEL('Sequence #')
DEFINE     FIELD(#DESCR) TYPE(*CHAR) LENGTH(30) LABEL('Description')
DEFINE     FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2) LABEL('Return code')
DEFINE     FIELD(#ENBADD) TYPE(*CHAR) LENGTH(1) LABEL('Enable ADD')
DEFINE     FIELD(#ENBCHG) TYPE(*CHAR) LENGTH(1) LABEL('Enable CHG')
DEFINE     FIELD(#ENBDLT) TYPE(*CHAR) LENGTH(1) LABEL('Enable DLT')
DEFINE     FIELD(#TRUE) TYPE(*CHAR) LENGTH(6) LABEL('Action if True')
DEFINE     FIELD(#FALSE) TYPE(*CHAR) LENGTH(6) LABEL('Action if False')
DEFINE     FIELD(#MSGDET) TYPE(*CHAR) LENGTH(27) LABEL('Message Detail')
DEFINE     FIELD(#MSGTXT) TYPE(*CHAR) LENGTH(80) LABEL('Message Text')
DEFINE     FIELD(#CONLIN) TYPE(*CHAR) LENGTH(79) LABEL('Condition line')
DEF_LIST   NAME(#CONWRK) FIELDS((#CONLIN)) TYPE(*WORKING) ENTRYS(5)
DEF_LIST   NAME(#CONBRW) FIELDS((#CONLIN)) ENTRYS(5)
GROUP_BY   NAME(#VALCHK) FIELDS((#LEVEL) (#FIELD) (#SEQNUM) (#DESCR)
           (#ENBADD) (#ENBCHG) (#ENBDLT) (#TRUE)
           (#FALSE) (#MSGDET) (#MSGTXT))
*********  Initialize Browse list
CLR_LIST   NAMED(#CONBRW)
INZ_LIST   NAMED(#CONBRW) NUM_ENTRYS(5) WITH_MODE(*CHANGE)
*********  Clear Working lists
BEGIN_LOOP
CLR_LIST   NAMED(#CONWRK)
*********  Request Validation check details
REQUEST    FIELDS((#VALCHK)) BROWSELIST(#CONBRW)
*********  Load key field working list
SELECTLIST NAMED(#CONBRW)
```

```
ADD_ENTRY  TO_LIST(#CONWRK)
ENDSELECT
*********  Execute built-in-function - PUT_COND_CHECK
USE        BUILTIN(PUT_COND_CHECK) WITH_ARGS(#LEVEL #FIELD #SEQNUM
           #DESCR #ENBADD #ENBCHG #ENBDLT #TRUE #FALSE #MSGDET
           #MSGTXT #CONWRK) TO_GET(#RETCOD)
*********  Put "simple conditional logic" successful
IF         COND('#RETCOD *EQ ''OK''')
MESSAGE    MSGTXT('Put "simple conditional logic" validation
           check(s) was successful')
*********  Put "simple conditional logic" failed
ELSE
IF         COND('#RETCOD *EQ ''ER''')
MESSAGE    MSGTXT('Put "simple conditional logic" validation
           check(s) failed')
ENDIF
ENDIF
END_LOOP
```

PUT_DATE_CHECK

Category: Validation related built in functions

Description: Create/amend a "date range / date format" DICTIONARY or FILE level validation check into the data dictionary or file definition of the nominated field.

When adding a FILE level validation check to a field, the file involved must have been previously placed into an edit session by the START_FILE_EDIT built in function.

All argument values passed to this built in function are validated exactly as if they had been entered through the online validation check definition screen panels.

Normal authority and task tracking rules apply to the use of this built in function.

For more information on "field validation checks", refer to the "field validation checks" section within the "Field Control" chapter in the LANSA Users Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Level of validation check.	1	1		
			D = Dictionary level				
			F = File level				
2	A	Req	Name of field in dictionary to which validation rule is to be applied.	1	10		
3	N	Req	Sequence number of check.	1	3	0	0
4	A	Req	Description of check.	1	30		
5	A	Req	Enable check for ADD.	1	1		
			Y = Check performed on ADD				
			U = Check performed on ADDUSE				
			N = Check not performed on ADD				
6	A	Req	Enable check for CHANGE.	1	1		
			Y = Check performed on CHG				
			U = Check performed on CHGUSE				
			N = Check not performed on CHG				
7	A	Req	Enable check for DELETE.	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Y = Enable check. N = Do not enable check.				
8	A	Req	Action if check is true. NEXT = Perform next check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.	4	6		
9	A	Req	Action if check is false. NEXT = Perform next check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.	4	6		
10	A	Req	Message file details . Details of error message to be issued from a message file. Message file details should be formatted as follows: **Bytes Description** 1-7 Error Message Number 8-17 Message File Name 18-27 Message File Library	27	27		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			If message text is used, pass this argument as blanks.				
11	A	Req	Message text.	1	80		
12	A	Req	Format that date is to be validated in.	1	8		
13	N	Opt	Number of days allowed into the past for specified date. If not specified, a value of 9999999 is assumed.	1	7	0	0
14	N	Opt	Number of days allowed into the future for specified date. If not specified, a value of 9999999 is assumed.	1	7	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = validation check defined ER = fatal error detected In case of "ER" return code error message(s) are issued automatically. When a file edit session is involved it is ended automatically without commitment.	2	2		

Example

A user wants to put a "date range / date format" validation check for a specific field, without going through the LANSA options provided on the "Field Control Menu", that enables the user to put a "date range / date format" validation check.

```
********* Define arguments and lists
DEFINE    FIELD(#LEVEL) TYPE(*CHAR) LENGTH(1) LABEL('Level')
DEFINE    FIELD(#FIELD) TYPE(*CHAR) LENGTH(10) LABEL('Field')
DEFINE    FIELD(#SEQNUM) TYPE(*DEC) LENGTH(3) DECIMALS(0)
          LABEL('Sequence #')
DEFINE    FIELD(#DESCR) TYPE(*CHAR) LENGTH(30) LABEL('Description')
DEFINE    FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2) LABEL('Return code')
DEFINE    FIELD(#ENBADD) TYPE(*CHAR) LENGTH(1) LABEL('Enable ADD')
DEFINE    FIELD(#ENBCHG) TYPE(*CHAR) LENGTH(1) LABEL('Enable CHG')
DEFINE    FIELD(#ENBDLT) TYPE(*CHAR) LENGTH(1) LABEL('Enable DLT')
DEFINE    FIELD(#TRUE ) TYPE(*CHAR) LENGTH(6) LABEL('Action if True')
DEFINE    FIELD(#FALSE) TYPE(*CHAR) LENGTH(6) LABEL('Action if False')
DEFINE    FIELD(#MSGDET) TYPE(*CHAR) LENGTH(27) LABEL('Message Detail')
DEFINE    FIELD(#MSGTXT) TYPE(*CHAR) LENGTH(80) LABEL('Message Text')
DEFINE    FIELD(#DATFMT) TYPE(*CHAR) LENGTH(8) LABEL('Date format')
DEFINE    FIELD(#DAYPST) TYPE(*DEC) LENGTH(7) DECIMALS(0)
          LABEL('Days Past')
DEFINE    FIELD(#DAYFUT) TYPE(*DEC) LENGTH(7) DECIMALS(0)
          LABEL('Days Future')
GROUP_BY  NAME(#VALCHK) FIELDS((#LEVEL) (#FIELD) (#SEQNUM)
           (#DESCR) (#ENBADD) (#ENBCHG) (#ENBDLT) (#TRUE)
           (#FALSE) (#MSGDET) (#MSGTXT) (#DATFMT)
           (#DAYPST) (#DAYFUT))
********* Request Validation check details
BEGIN_LOOP
REQUEST   FIELDS((#VALCHK))
********* Execute built-in-function - PUT_DATE_CHECK
USE       BUILTIN(PUT_DATE_CHECK) WITH_ARGS(#LEVEL #FIELD #SEQNUM
          #DESCR #ENBADD #ENBCHG #ENBDLT #TRUE #FALSE #MSGDET
          #MSGTXT #DATFMT #DAYPST #DAYFUT) TO_GET(#RETCOD)
```

```
********* Put "date range/format" validation check was successful
IF        COND('#RETCOD *EQ ''OK''')
MESSAGE   MSGTXT('Put "date range/format" validation check(s) was
          successful')
********* Put "date range/format" failed
ELSE
IF        COND('#RETCOD *EQ ''ER''')
MESSAGE   MSGTXT('Put "date range/format" validation check(s)
                failed')
ENDIF
ENDIF
END_LOOP
```

PUT_FILE_CHECK

Category: Validation related built in functions

Description: Create/amend a "code/table file lookup" DICTIONARY or FILE level validation check into the data dictionary or file definition of the nominated field.

When adding a FILE level validation check to a field, the file involved must have been previously placed into an edit session by the START_FILE_EDIT built in function.

All argument values passed to this built in function are validated exactly as if they had been entered through the online validation check definition screen panels.

Normal authority and task tracking rules apply to the use of this built in function.

For more information on "field validation checks", refer to the "field validation checks" section within the "Field Control" chapter in the LANSA Users Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Level of validation check. D = Dictionary level	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			F = File level				
2	A	Req	Name of field in dictionary to which validation rule is to be applied.	1	10		
3	N	Req	Sequence number of check.	1	3	0	0
4	A	Req	Description of check.	1	30		
5	A	Req	Enable check for ADD.	1	1		
			Y = Check performed on ADD				
			U = Check performed on ADDUSE				
			N = Check not performed on ADD				
6	A	Req	Enable check for CHANGE.	1	1		
			Y = Check performed on CHG				
			U = Check performed on CHGUSE				
			N = Check not performed on CHG				
7	A	Req	Enable check for DELETE.	1	1		
			Y = Enable check.				
			N = Do not enable check.				
8	A	Req	Action if check is true.	4	6		
			NEXT = Perform next				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.				
9	A	Req	Action if check is false. NEXT = Perform next check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.	4	6		
10	A	Req	Message file details. Details of error message to be issued from a message file. Message file details should be formatted as follows: **Bytes Description** 1-7 Error Message Number 8-17 Message File Name 18-27 Message File Library If message text is used, pass this argument as blanks.	27	27		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
11	A	Req	Message text.	1	80		
12	A	Req	Name of file that check is to be performed against.	1	10		
13	List	Req	Working list to contain key fields/values to use when checking in the file. The calling RDML function must provide a working list with an aggregate entry length of exactly 20 bytes and at most 10 key fields/values entries may be specified. Each list entry sent should be formatted as follows : **Bytes Description** 1-20 Key fields/values	1	20		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = validation check defined ER = fatal error detected In case of "ER" return code error message(s) are issued automatically. When a file edit session is	2	2		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			involved it is ended automatically without commitment.				

Example

A user wants to put a "code/table file lookup" validation check for a specific field, without going through the LANSA options provided on the "Field Control Menu" that enables the user to put a "code / table file lookup" validation check.

```
********* Define arguments and lists
DEFINE      FIELD(#FILNAM) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#LIBNAM) TYPE(*CHAR) LENGTH(10)
DEFINE      FIELD(#LEVEL) TYPE(*CHAR) LENGTH(1) LABEL('Level')
DEFINE      FIELD(#FIELD) TYPE(*CHAR) LENGTH(10) LABEL('Field')
DEFINE      FIELD(#SEQNUM) TYPE(*DEC) LENGTH(3) DECIMALS(0)
            LABEL('Sequence #')
DEFINE      FIELD(#DESCR) TYPE(*CHAR) LENGTH(30) LABEL('Description')
DEFINE      FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2) LABEL('Return code')
DEFINE      FIELD(#ENBADD) TYPE(*CHAR) LENGTH(1) LABEL('Enable ADD')
DEFINE      FIELD(#ENBCHG) TYPE(*CHAR) LENGTH(1) LABEL('Enable CHG')
DEFINE      FIELD(#ENBDLT) TYPE(*CHAR) LENGTH(1) LABEL('Enable DLT')
DEFINE      FIELD(#TRUE) TYPE(*CHAR) LENGTH(6) LABEL('Action if True')
DEFINE      FIELD(#FALSE) TYPE(*CHAR) LENGTH(6) LABEL('Action if False')
DEFINE      FIELD(#MSGDET) TYPE(*CHAR) LENGTH(27) LABEL('Message Detail')
DEFINE      FIELD(#MSGTXT) TYPE(*CHAR) LENGTH(80) LABEL('Message Text')
DEFINE      FIELD(#CODFIL) TYPE(*CHAR) LENGTH(10) LABEL('File name')
DEFINE      FIELD(#KEYFLD) TYPE(*CHAR) LENGTH(20) LABEL('Key field')
DEF_LIST    NAME(#KEYWRK) FIELDS((#KEYFLD)) TYPE(*WORKING) ENTRYS(10)
DEF_LIST    NAME(#KEYBRW) FIELDS((#KEYFLD)) ENTRYS(10)
GROUP_BY    NAME(#VALCHK) FIELDS((#LEVEL) (#FIELD) (#SEQNUM) (#DESCR)
            (#ENBADD) (#ENBCHG) (#ENBDLT) (#TRUE)
            (#FALSE) (#MSGDET) (#MSGTXT) (#CODFIL))
*********   Initialize Browse list
```

```
CLR_LIST    NAMED(#KEYBRW)
INZ_LIST    NAMED(#KEYBRW) NUM_ENTRYS(10) WITH_MODE(*CHANGE)
*********   Start file edit
REQUEST     FIELDS(#FILNAM #LIBNAM)
**********
USE         BUILTIN(START_FILE_EDIT)
            WITH_ARGS(#FILNAM #LIBNAM 'DEM')
            TO_GET(#RETCOD)
*********   Clear Working lists
BEGIN_LOOP
CLR_LIST    NAMED(#KEYWRK)
*********   Request Validation check details
REQUEST     FIELDS((#VALCHK)) BROWSELIST(#KEYBRW)
*********   Load key field working list
SELECTLIST NAMED(#KEYBRW)
ADD_ENTRY   TO_LIST(#KEYWRK)
ENDSELECT
*********   Execute built-in-function - PUT_FILE_CHECK
USE         BUILTIN(PUT_FILE_CHECK) WITH_ARGS(#LEVEL #FIELD #SEQNUM
            #DESCR #ENBADD #ENBCHG #ENBDLT #TRUE #FALSE #MSGDET
            #MSGTXT #CODFIL #KEYWRK) TO_GET(#RETCOD)
*********   Put "code/table file lookup" validation successful
IF          COND('#RETCOD *EQ ''OK''')
MESSAGE     MSGTXT('Put "code/table file lookup" validation
            check(s) was successful')
*********   Put "code/table file lookup" failed
ELSE
IF          COND('#RETCOD *EQ ''ER''')
MESSAGE     MSGTXT('Put "code/table file lookup" validation
            check(s) failed')
ENDIF
ENDIF
END_LOOP
USE         BUILTIN(END_FILE_EDIT) ('Y')
```

PUT_PROGRAM_CHECK

Category: Validation related built in functions

Description: Create/amend a "call user program" DICTIONARY or FILE level validation check into the data dictionary or file definition of the nominated field.

When adding a FILE level validation check to a field, the file involved must have been previously placed into an edit session by the START_FILE_EDIT built in function.

All argument values passed to this built in function are validated exactly as if they had been entered through the online validation check definition screen panels.

Normal authority and task tracking rules apply to the use of this built in function.

For more information on "field validation checks", refer to the "field validation checks" section within the "Field Control" chapter in the LANSA Users Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Level of validation check. D = Dictionary level	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			F = File level				
2	A	Req	Name of field in dictionary to which validation rule is to be applied.	1	10		
3	N	Req	Sequence number of check.	1	3	0	0
4	A	Req	Description of check.	1	30		
5	A	Req	Enable check for ADD.	1	1		
			Y = Check performed on ADD				
			U = Check performed on ADDUSE				
			N = Check not performed on ADD				
6	A	Req	Enable check for CHANGE.	1	1		
			Y = Check performed on CHG				
			U = Check performed on CHGUSE				
			N = Check not performed on CHG				
7	A	Req	Enable check for DELETE.	1	1		
			Y = Enable check.				
			N = Do not enable check.				
8	A	Req	Action if check is true.	4	6		
			NEXT = Perform next				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.				
9	A	Req	Action if check is false. NEXT = Perform next check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.	4	6		
10	A	Req	Message file details Details of error message to be issued from a message file. Message file details should be formatted as follows: **Bytes Description** 1-7 Error Message Number 8-17 Message File Name 18-27 Message File Library If message text is used, pass this argument as blanks.	27	27		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
11	A	Req	Message text.	1	80		
12	A	Req	Name of program that is to be called to perform this check. Prefix name by "LF=" if a function is to be called. Note - additional parameters are not allowed if a function is performing the check.	1	10		
13	List	Req	Working list to contain the additional parameters that should be passed to the nominated program. The calling RDML function must provide a working list with an aggregate entry length of exactly 20 bytes and exactly 10 parameter entries. Each list entry sent should be formatted as follows: **Bytes Description** 1-20 Additional parameter	1	20		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = validation check defined ER = fatal error detected In case of "ER" return code error message(s) are issued automatically. When a file edit session is involved it is ended automatically without commitment.	2	2		

Example

A user wants to put a "call user program" validation check for a specific field, without going through the LANSA options provided on the "Field Control Menu" that enables the user to put a "call user program" validation check.

```
*********  Define arguments and lists
DEFINE     FIELD(#LEVEL) TYPE(*CHAR) LENGTH(1) LABEL('Level')
DEFINE     FIELD(#FIELD) TYPE(*CHAR) LENGTH(10) LABEL('Field')
DEFINE     FIELD(#SEQNUM) TYPE(*DEC) LENGTH(3) DECIMALS(0)
           LABEL('Sequence #')
DEFINE     FIELD(#DESCR) TYPE(*CHAR) LENGTH(30) LABEL('Description')
DEFINE     FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2) LABEL('Return code')
DEFINE     FIELD(#ENBADD) TYPE(*CHAR) LENGTH(1) LABEL('Enable ADD')
DEFINE     FIELD(#ENBCHG) TYPE(*CHAR) LENGTH(1) LABEL('Enable CHG')
DEFINE     FIELD(#ENBDLT) TYPE(*CHAR) LENGTH(1) LABEL('Enable DLT')
DEFINE     FIELD(#TRUE) TYPE(*CHAR) LENGTH(6) LABEL('Action if True')
DEFINE     FIELD(#FALSE) TYPE(*CHAR) LENGTH(6) LABEL('Action if False' )
DEFINE     FIELD(#MSGDET) TYPE(*CHAR) LENGTH(27) LABEL('Message Detail')
DEFINE      FIELD(#MSGTXT) TYPE(*CHAR) LENGTH(80) LABEL('Message Text')
DEFINE      FIELD(#USRPGM) TYPE(*CHAR) LENGTH(10) LABEL('User program')
DEFINE      FIELD(#PGMPRM) TYPE(*CHAR) LENGTH(20) LABEL('program parms')
DEF_LIST    NAME(#PRMWRK) FIELDS((#PGMPRM)) TYPE(*WORKING) ENTRYS(10)
DEF_LIST    NAME(#PRMBRW) FIELDS((#PGMPRM)) ENTRYS(10)
GROUP_BY    NAME(#VALCHK) FIELDS((#LEVEL) (#FIELD) (#SEQNUM) (#DESCR)
            (#ENBADD) (#ENBCHG) (#ENBDLT) (#TRUE)
            (#FALSE) (#MSGDET) (#MSGTXT) (#USRPGM))
*********   Initialize Browse list
CLR_LIST    NAMED(#PRMBRW)
INZ_LIST    NAMED(#PRMBRW) NUM_ENTRYS(10) WITH_MODE(*CHANGE)
*********   Clear Working lists
BEGIN_LOOP
CLR_LIST    NAMED(#PRMWRK)
*********   Request Validation check details
REQUEST     FIELDS((#VALCHK)) BROWSELIST(#PRMBRW)
*********   Load key field working list
SELECTLIST NAMED(#PRMBRW)
ADD_ENTRY   TO_LIST(#PRMWRK)
ENDSELECT
*********   Execute built-in-function - PUT_PROGRAM_CHECK
USE         BUILTIN(PUT_PROGRAM_CHECK) WITH_ARGS(#LEVEL #FIELD
            #SEQNUM #DESCR #ENBADD #ENBCHG #ENBDLT #TRUE #FALSE
            #MSGDET #MSGTXT #USRPGM #PRMWRK) TO_GET(#RETCOD)
*********   Put "call user program" validation successful
IF          COND('#RETCOD *EQ ''OK''')
MESSAGE     MSGTXT('Put "call user program" validation check(s) was
            successful')
*********   Put "call user program" failed
ELSE
IF          COND('#RETCOD *EQ ''ER''')
MESSAGE     MSGTXT('Put "call user program" validation check(s) failed')
ENDIF
ENDIF
END_LOOP
```

PUT_RANGE_CHECK

Category: Validation related built in functions

Description: Create/amend a "range of values" DICTIONARY or FILE level validation check into the data dictionary or file definition of the nominated field.

When adding a FILE level validation check to a field, the file involved must have been previously placed into an edit session by the START_FILE_EDIT built in function.

All argument values passed to this built in function are validated exactly as if they had been entered through the online validation check definition screen panels.

Normal authority and task tracking rules apply to the use of this built in function.

For more information on "field validation checks", refer to the "field validation checks" section within the "Field Control" chapter in the LANSA Users Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Level of validation check.	1	1		
			D = Dictionary level				
			F = File level				
2	A	Req	Name of field in dictionary to which validation rule is to be applied.	1	10		
3	N	Req	Sequence number of check.	1	3	0	0
4	A	Req	Description of check.	1	30		
5	A	Req	Enable check for ADD.	1	1		
			Y = Check performed on ADD				
			U = Check performed on ADDUSE				
			N = Check not performed on ADD				
6	A	Req	Enable check for CHANGE.	1	1		
			Y = Check performed on CHG				
			U = Check performed on CHGUSE				
			N = Check not performed on CHG				
7	A	Req	Enable check for DELETE.	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Y = Enable check. N = Do not enable check.				
8	A	Req	Action if check is true. NEXT = Perform next check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.	4	6		
9	A	Req	Action if check is false. NEXT = Perform next check ERROR = Issue fatal error ACCEPT = Accept value and do no more checking.	4	6		
10	A	Req	Message file details Details of error message to be issued from a message file. Message file details should be formatted as follows: **Bytes Description** 1-7 Error Message Number 8-17 Message File Name 18-27 Message File	27	27		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Library If message text is used, pass this argument as blanks.				
11	A	Req	Message text.	1	80		
12	List	Req	Working list to contain "from" range values. The calling RDML function must provide a working list with an aggregate entry length of exactly 20 bytes and at most 20 "from" range value entries may be specified. Each "from" range entry passed must have a matching "to" value entry or unpredictable results may occur. Each list entry sent should be formatted as follows: **Bytes Description** 1-20 "From" range value	1	20		
13	List	Req	Working list to contain "to" range values. The calling RDML function must provide a working list with an aggregate entry length of exactly 20 bytes and at most 20 "to" range value	1	20		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			entries may be specified. Each "to" range entry passed must have a matching "from" value entry or unpredictable results may occur. Each list entry sent should be formatted as follows: **Bytes Description** 1-20 "To" range value				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = validation check defined ER = fatal error detected In case of "ER" return code error message(s) are issued automatically. When a file edit session is involved it is ended automatically without commitment.	2	2		

Example

A user wants to put a "range of values" validation check for a specific field, without going through the LANSA options provided on the "Field Control Menu" that enables the user to put a "range of values" validation check.

```
*********  Define arguments and lists
DEFINE     FIELD(#LEVEL) TYPE(*CHAR) LENGTH(1) LABEL('Level')
DEFINE     FIELD(#FIELD) TYPE(*CHAR) LENGTH(10) LABEL('Field')
DEFINE     FIELD(#SEQNUM) TYPE(*DEC) LENGTH(3) DECIMALS(0)
           LABEL('Sequence #')
DEFINE     FIELD(#DESCR) TYPE(*CHAR) LENGTH(30) LABEL('Description')
DEFINE     FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2) LABEL('Return code')
DEFINE     FIELD(#ENBADD) TYPE(*CHAR) LENGTH(1) LABEL('Enable ADD')
DEFINE     FIELD(#ENBCHG) TYPE(*CHAR) LENGTH(1) LABEL('Enable CHG')
DEFINE     FIELD(#ENBDLT) TYPE(*CHAR) LENGTH(1) LABEL('Enable DLT')
DEFINE     FIELD(#TRUE) TYPE(*CHAR) LENGTH(6) LABEL('Action if True')
DEFINE     FIELD(#FALSE) TYPE(*CHAR) LENGTH(6) LABEL('Action if False')
DEFINE     FIELD(#MSGDET) TYPE(*CHAR) LENGTH(27) LABEL('Message Detail')
DEFINE     FIELD(#MSGTXT) TYPE(*CHAR) LENGTH(80) LABEL('Message Text')
DEFINE     FIELD(#FRMRNG) TYPE(*CHAR) LENGTH(20) LABEL('From range')
DEFINE     FIELD(#TORNG) TYPE(*CHAR) LENGTH(20) LABEL('To range')
DEF_LIST   NAME(#FRMWRK) FIELDS((#FRMRNG)) TYPE(*WORKING) ENTRYS(20)
DEF_LIST   NAME(#TOWRK) FIELDS((#TORNG)) TYPE(*WORKING) ENTRYS(20)
DEF_LIST   NAME(#RNGBRW) FIELDS((#FRMRNG) (#TORNG)) ENTRYS(20)
GROUP_BY   NAME(#VALCHK) FIELDS((#LEVEL) (#FIELD) (#SEQNUM) (#DESCR)
           (#ENBADD) (#ENBCHG) (#ENBDLT) (#TRUE)
           (#FALSE) (#MSGDET) (#MSGTXT))
*********  Initialize Browse list
CLR_LIST   NAMED(#RNGBRW)
INZ_LIST   NAMED(#RNGBRW) NUM_ENTRYS(20) WITH_MODE(*CHANGE)
*********  Clear Working lists
BEGIN_LOOP
CLR_LIST   NAMED(#FRMWRK)
CLR_LIST   NAMED(#TOWRK)
*********  Request Validation check details
REQUEST    FIELDS((#VALCHK)) BROWSELIST(#RNGBRW)
```

```
*********  Load From and To range value working lists
SELECTLIST NAMED(#RNGBRW)
ADD_ENTRY  TO_LIST(#FRMWRK)
ADD_ENTRY  TO_LIST(#TOWRK)
ENDSELECT
*********  Execute built-in-function - PUT_RANGE_CHECK
USE        BUILTIN(PUT_RANGE_CHECK) WITH_ARGS(#LEVEL #FIELD #SEQNUM
           #DESCR #ENBADD #ENBCHG #ENBDLT #TRUE #FALSE #MSGDET
           #MSGTXT #FRMWRK #TOWRK) TO_GET(#RETCOD)
*********  Put "range of values" validation check was successful
IF         COND('#RETCOD *EQ ''OK''')
MESSAGE    MSGTXT('Put "range of values" validation check(s) was
           successful')
*********  Put "range of values" failed
ELSE
IF         COND('#RETCOD *EQ ''ER''')
MESSAGE    MSGTXT('Put "range of values" validation check(s) failed')
ENDIF
ENDIF
END_LOOP
```

PUT_TRIGGER

Category: Validation related built in functions

Description: Create/amend a DICTIONARY or FILE level trigger into the data dictionary or file definition of the nominated field.

When adding a FILE level trigger to a field, the file involved must have been previously placed into an edit session by the START_FILE_EDIT built in function.

All argument values passed to this built in function are validated exactly as if they had been entered through the online validation check definition screen panels.

Normal authority and task tracking rules apply to the use of this built in function.

For more information on "field rules/triggers", refer to the "field rules/triggers" section within the "Field Control" chapter in the LANSA Users Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Level of validation check. D = Dictionary level	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			F = File level				
2	A	Req	Name of field in dictionary to which trigger rule is to be applied.	1	10		
3	N	Req	Sequence number of trigger.	1	3	0	0
4	A	Req	Description of trigger.	1	30		
5	A	Req	Name of trigger function.	1	7		
6	List	Req	Working list of trigger points. The calling RDML function must provide a working list with an aggregate entry length of exactly 5 bytes and at most 6 trigger point value entries may be specified. Each trigger point is associated with a "before" and an "after" entry. At least one trigger point must have one of these set to "Y". The trigger point must be specified in 3 characters as one of: OPN for Open CLS for Close RED for Read INS for Insert	5	5		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			UPD for Update DLT for Delete Each list entry sent should be formatted as follows: **Bytes** **Description** 1-3 Trigger position 4-4 Trigger before 5-5 Trigger after				
7	List	Req	Working list of trigger conditions. The calling RDML function may provide a working list with an aggregate entry length of exactly 36 bytes and at most 20 trigger conditions entries may be specified. Each list entry sent should be formatted as follows: **Bytes** **Description** 1-3 AND / OR 4-13 Field name 14-16 Operation code 17-36 Value	36	36		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = field details returned ER = field not accessible In case of "ER" return code error message(s) are issued automatically.	2	2		

PUT_VALUE_CHECK

Category: Validation related built in functions

Description: Create/amend a "list of values" DICTIONARY or FILE level validation check into the data dictionary or file definition of the nominated field.

When adding a FILE level validation check to a field, the file involved must have been previously placed into an edit session by the START_FILE_EDIT built in function.

All argument values passed to this built in function are validated exactly as if they had been entered through the online validation check definition screen panels.

Normal authority and task tracking rules apply to the use of this built in function.

For more information on "field validation checks", refer to the "field validation checks" section within the "Field Control" chapter in the LANSA Users Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Level of validation check.	1	1		
			D = Dictionary level				
			F = File level				
2	A	Req	Name of field in dictionary to which validation rule is to be applied.	1	10		
3	N	Req	Sequence number of check.	1	3	0	0
4	A	Req	Description of check.	1	30		
5	A	Req	Enable check for ADD.	1	1		
			Y = Check performed on ADD				
			U = Check performed on ADDUSE				
			N = Check not performed on ADD				
6	A	Req	Enable check for CHANGE.	1	1		
			Y = Check performed on CHG				
			U = Check performed on CHGUSE				
			N = Check not performed on CHG				
7	A	Req	Enable check for DELETE.	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Y = Enable check.				
			N = Do not enable check.				
8	A	Req	Action if check is true.	4	6		
			NEXT Perform next check				
			ERROR Issue fatal error				
			ACCEPT = Accept value and do no more checking.				
9	A	Req	Action if check is false.	4	6		
			NEXT Perform next check				
			ERROR Issue fatal error				
			ACCEPT = Accept value and do no more checking.				
10	A	Req	Message file details	27	27		
			Details of error message to be issued from a message file. Message file details should be formatted as follows:				
			Bytes Description				
			1-7 Error Message Number				
			8-17 Message File Name				
			18-27 Message File Library If message text is used, pass this argument as blanks.				
11	A	Req	Message text.	1	80		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
12	List	Req	Working list to contain list values. The calling RDML function must provide a working list with an aggregate entry length of exactly 20 bytes and at most 50 list value entries may be specified. Each list entry sent should be formatted as follows: **Bytes** **Description** 1-20 List value	20	20		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = validation check defined ER = fatal error detected In case of "ER" return code error message(s) are issued automatically. When a file edit session is involved it is ended automatically without commitment.	2	2		

Example

A user wants to put a "list of values" validation check for a specific field, without going through the LANSA options provided on the "Field Control Menu" that enables the user to put a "list of values" validation check.

```
*********  Define arguments and lists
DEFINE     FIELD(#LEVEL) TYPE(*CHAR) LENGTH(1) LABEL('Level')
DEFINE     FIELD(#FIELD) TYPE(*CHAR) LENGTH(10) LABEL('Field')
DEFINE     FIELD(#SEQNUM) TYPE(*DEC) LENGTH(3) DECIMALS(0)
           LABEL('Sequence #')
DEFINE     FIELD(#DESCR) TYPE(*CHAR) LENGTH(30) LABEL('Description')
DEFINE     FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2) LABEL('Return code')
DEFINE     FIELD(#ENBADD) TYPE(*CHAR) LENGTH(1) LABEL('Enable ADD')
DEFINE     FIELD(#ENBCHG) TYPE(*CHAR) LENGTH(1) LABEL('Enable CHG')
DEFINE     FIELD(#ENBDLT) TYPE(*CHAR) LENGTH(1) LABEL('Enable DLT')
DEFINE     FIELD(#TRUE) TYPE(*CHAR) LENGTH(6) LABEL('Action if True')
DEFINE     FIELD(#FALSE) TYPE(*CHAR) LENGTH(6) LABEL('Action if False')
DEFINE     FIELD(#MSGDET) TYPE(*CHAR) LENGTH(27) LABEL('Message Detail')
DEFINE     FIELD(#MSGTXT) TYPE(*CHAR) LENGTH(80) LABEL('Message Text')
DEFINE     FIELD(#LSTVAL) TYPE(*CHAR) LENGTH(20) LABEL('List value')
DEF_LIST   NAME(#VALWRK) FIELDS((#LSTVAL)) TYPE(*WORKING) ENTRYS(50)
DEF_LIST   NAME(#VALBRW) FIELDS((#LSTVAL)) ENTRYS(50)
GROUP_BY   NAME(#VALCHK) FIELDS((#LEVEL) (#FIELD) (#SEQNUM) (#DESCR)
           (#ENBADD) (#ENBCHG) (#ENBDLT) (#TRUE)
           (#FALSE) (#MSGDET) (#MSGTXT))
*********  Initialize Browse list
CLR_LIST   NAMED(#VALBRW)
INZ_LIST   NAMED(#VALBRW) NUM_ENTRYS(50) WITH_MODE(*CHANGE)
*********  Clear Working list
BEGIN_LOOP
CLR_LIST   NAMED(#VALWRK)
*********  Request Validation check details
REQUEST    FIELDS((#VALCHK)) BROWSELIST(#VALBRW)
*********  Load list of values working list
SELECTLIST NAMED(#VALBRW)
ADD_ENTRY  TO_LIST(#VALWRK)
```

```
ENDSELECT
*********  Execute built-in-function - PUT_VALUE_CHECK
USE        BUILTIN(PUT_VALUE_CHECK) WITH_ARGS(#LEVEL #FIELD #SEQNUM
           #DESCR #ENBADD #ENBCHG #ENBDLT #TRUE #FALSE #MSGDET
           #MSGTXT #VALWRK) TO_GET(#RETCOD)
*********  Put "list of values" validation check was successful
IF         COND('#RETCOD *EQ ''OK''')
MESSAGE    MSGTXT('Put "list of values" validation check(s) was
           successful')
*********  Put "list of values" failed
ELSE
IF         COND('#RETCOD *EQ ''ER''')
MESSAGE    MSGTXT('Put "list of values" validation check(s) failed')
ENDIF
ENDIF
END_LOOP
```

Chapter 4. Process Related Built-In Functions

COMPILE_PROCESS

Category: Process related built in functions

Description: Submits a batch job to compile a process and all selected functions

Argument values are exactly as per information input on the "Compile / Re-Compile a Process" screen described in the User Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other process related built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Process name	1	10		
2	List	Req	Working list to contain function names. The calling RDML function must provide a working list with an aggregate entry length of exactly 7 bytes. If you do not wish to specify any functions for compilation then you must pass an empty working list.	1	7		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Each returned list entry is formatted as follows: **Bytes Description** 1-7 Function name				
3	A	Opt	Name of batch job Default: Process name	1	10		
4	A	Opt	Name of job description Default: the job description from the requesting job's attributes.	1	21		
5	A	Opt	Name of job queue Default: the job queue from the requesting job's attributes.	1	21		
6	A	Opt	Name of output queue Default: the output queue from the requesting job's attributes.	1	21		
7	A	Opt	Compile process as well as functions ? YES = compile process NO = do not compile process Default: the "compile process default" value at position 461 in the system definition data area DC@A01.	1	3		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
8	A	Opt	Produce RDML source listing? YES = produce RDML listing NO = do not produce listing Default: the "source listing default" value at position 146 in the system definition data area DC@A01.	1	3		
9	A	Opt	Produce RPG & DDS source listings? YES = produce RPG & DDS listings NO = do not produce listings Default: the "source listing default" value at position 146 in the system definition data area DC@A01.	1	3		
10	A	Opt	Optimize compiled program(s)? YES = optimize program(s) NO = do not optimize Default: the "optimize compile default" value at position 147 in the system definition data area DC@A01.	1	3		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
11	A	Opt	Ignore decimal data errors in program(s)? YES = ignore decimal data errors NO = do not ignore errors Default: the "decimal data error default" value at position 148 in the system definition data area DC@A01.	1	3		
12	A	Opt	Allow debug / Remove observability? YESYES = Allow program(s) to be used in debug and do not remove observability. NO NO = Do not allow debug and remove program observability NO YES = Do not allow debug but do not remove the programs observability. Default: the "enable debug default" value at position 400 in the system definition data area DC@A01. Warning: Do not specify YESNO for this parameter. The DEBUG facility cannot work if a program is not	1	6		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			observable.				
13	A	Opt	Dump code generator work areas? YES = Dump work areas NO = Do not dump work areas Default: YES	1	3		
14	A	Opt	Produce Documentor details? YES = Produce Documentor details NO = Do not produce Documentor details Default: YES if Documentor is enabled at the partition level, otherwise NO.	1	3		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = successful submission. ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

Example

A user wants to control the compilation of processes and functions using their own version of the "Compile / Re-Compile a Process" facility.

```
*********  Define arguments and lists
DEFINE     FIELD(#PROCES) TYPE(*CHAR) LENGTH(10)
DEFINE     FIELD(#FUNCTN) TYPE(*CHAR) LENGTH(7)
DEFINE     FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2)
DEF_LIST   NAME(#WKFUNL) FIELDS((#FUNCTN)) TYPE(*WORKING)
DEF_LIST   NAME(#BWFUNL) FIELDS((#FUNCTN))
*********  Clear working and browse lists
BEGIN_LOOP
CLR_LIST   NAMED(#WKFUNL)
INZ_LIST   NAMED(#BWFUNL) NUM_ENTRYS(10) WITH_MODE(*CHANGE)
*********  Request Process and Functions
REQUEST    FIELDS(#PROCES) BROWSELIST(#BWFUNL)
*********  Move Functions from the browselist to the working list
SELECTLIST NAMED(#BWFUNL)
ADD_ENTRY  TO_LIST(#WKFUNL)
ENDSELECT
*********  Execute built-in-function - COMPILE_PROCESS
USE        BUILTIN(COMPILE_PROCESS) WITH_ARGS(#PROCES #WKFUNL)
           TO_GET(#RETCOD)
*********  Check if submission was successful
IF         COND('#RETCOD *EQ ''OK''')
MESSAGE    MSGTXT('Compile Process submitted successfully')
CHANGE     FIELD(#PROCES) TO(*BLANK)
ELSE
MESSAGE    MSGTXT('Compile Process submit failed with errors,
           refer to additional messages')
ENDIF
END_LOOP
```

DELETE_PROCESS

Category: Process related built in functions

Description: Submits a batch job to delete a process and all of its functions

Argument values are exactly as per information input on the "Delete a Process" screen described in the User Guide.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other process related built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Process name	1	10		
2	A	Opt	Name of batch job Default: Process name	1	10		
3	A	Opt	Name of job description Default: the job description from the requesting job's attributes.	1	21		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
4	A	Opt	Name of job queue Default: the job queue from the requesting job's attributes.	1	21		
5	A	Opt	Name of output queue Default: the output queue from the requesting job's attributes.	1	21		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = successful submission ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

Example

A user wants to control the deletion of processes using their own version of the "Delete a Process" facility.

```
*********  Define arguments
DEFINE     FIELD(#PROCES) TYPE(*CHAR) LENGTH(10)
DEFINE     FIELD(#RETCOD) TYPE(*CHAR) LENGTH(2)
*********  Request Process
BEGIN_LOOP
REQUEST    FIELDS(#PROCES)
*********  Execute built-in-function - DELETE_PROCESS
USE        BUILTIN(DELETE_PROCESS) WITH_ARGS(#PROCES) TO_GET(#RETCOD)
*********  Check if submission was successful
IF         COND('#RETCOD *EQ ''OK''')
MESSAGE    MSGTXT('Delete Process submitted successfully')
CHANGE     FIELD(#PROCES) TO(*BLANK)
ELSE
MESSAGE    MSGTXT('Delete Process submit failed with errors,
           refer to additional messages')
ENDIF
END_LOOP
```

DLT_PROCESS_ATTACH

Category: Process related built in functions

Description: Deletes all attached processes and/or functions from the definition of the process definition currently being edited by the START_PROCESS_EDIT built in function.

Information passed into this built in function is subjected to the same editing and validation rules as the equivalent online facility provided in a full LANSA development environment.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Delete Test Sequence Number. All attached processes / functions with a sequence number greater than or equal to this value are to be deleted.	1	3	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

END_PROCESS_EDIT

Category: Process related built in functions

Description: Ends an active edit session on a LANSA process definition.

An edit session is commenced by using the built in function START_PROCESS_EDIT.

A process edit session should be terminated by using the END_PROCESS_EDIT built in function to ensure all locks/etc are released/shutdown in an orderly manner.

Any process edit session that receives a fatal error will have an END_PROCESS_EDIT command automatically issued.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No argument values.

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = edit session ended ER = fatal error detected	2	2		

GET_PROCESS_ATTR

Category: Process related built in functions

Description: Get attributes of a process definition that is being edited within an edit session previously started by using the START_PROCESS_EDIT built in function.

Attributes set or returned by this built in function have the same editing and validation rules as the equivalent online facility provided in a full LANSA development environment.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of attribute to get Valid attribute names are: DESC- Process Description TYPE- Process Type OPTCOM - Optimize for remote comms ENAGUI - Enable for GUI	1	50		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			TOTFUN - Total associated functions EXISTS - Checks for existence of function name specified in bytes 7 to 13 of argument (directly following the EXISTS string in bytes 1 to 6).				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		
2	A	Req	Returned process attribute **For attribute DESC:** The process description as A(40) **For attribute TYPE:** SAA/CUA ACT/BAR **For attribute OPTCOMM:** Y N **For attribute ENAGUI:** Y N **For attribute TOTFUN:** Character 3 value containing a number in the range 000 - 990.	1	256		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			For attribute EXISTS:				
			Y				
			N				

GET_PROCESS_INFO

Category: Process related built in functions

Description: Retrieves a list of process related information from the LANSA internal database and returns it to the calling RDML function in a variable length working list.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

: **Special Note:** This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	**Type A/N**	**Req/ Opt**	**Description**	**Min Len**	**Max Len**	**Min Dec**	**Max Dec**
1	A	Req	Process name.	1	10		
2	A	Req	Type of process related information to retrieve. Valid types are: PROCATTACH - Attached processes/ functions MLATTR- Multilingual attributes	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = list returned partially or completely filled. No more of this type of information exists for this process. OV = list returned completely filled, but more of this type of information than could fit in the list exists. NR = list was returned empty. Last entry in the list is returned as null. ER = Process not found. Last entry in the list is returned as null.	2	2		
2	List	Req	Working list to contain process related information. The calling RDML function must provide a working list with an aggregate entry length of exactly 100 bytes. **For type PROCATTACH:** Each returned list entry is formatted as follows: **Bytes Description** 1-10 Attached process name	100	100		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			11-17 Attached function name (*ALL if process is attached) 18-100 <<future expansion>> **For type MLATTR:** Each returned list entry is formatted as follows: **Bytes Description** 1-4 Language code 5-44 Process description 45-100 <<future expansion>>				

GET_PROCESS_LIST

Category: Process related built in functions

Description: Retrieves a list of processes and their descriptions from the LANSA internal database and returns them to the calling RDML function in a variable length working list.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Positioning process value. The returned list starts with the first process from the dictionary whose name is greater than the value passed in this argument.	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain Process information. The calling RDML function must provide a working list with an aggregate entry length of exactly 60 bytes. Each returned list entry is formatted as follows: **Bytes Description** 1-10 Process name 11-50 Description 51-60 <<future expansion>>	60	60		
2	A	Opt	Last process in returned list Typically this value is used as the positioning argument on a subsequent calls to this built in function.	1	10		
3	A	Opt	Return code. OK = list returned partially or completely filled with process details. No more processes exist beyond those returned in the list. OV = list returned completely filled, but more processes than	2	2		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			could fit in the list exist. Typically used to indicate "more" processes in page at a time style list displays. NR = list was returned empty. Last process in the list is returned as blanks.				

Example

A program can be created, using this function, to compile a series of processes in an overnight job.

```
DEFINE      NAME (#PRONAM) TYPE(*CHAR) LENGTH(10)
DEFINE      NAME (#PRODES) TYPE(*CHAR) LENGTH(40)
DEFINE      NAME (#SPARE) TYPE(*CHAR) LENGTH(10)
DEF_LIST    NAME(#PROLST) FIELDS(#PRONAM #PRODES #SPARE)
            TYPE(*WORKING) ENTRYS(10)
**********  If interactive mode
IF          COND('MODE *EQ I')
**********  -Clear list-
CLR_LIST    NAMED(#PROLST)
**********  -Request process to start from in list-
REQUEST     FIELDS(#STRPRO) TEXT(('Process to start from' 5 5))
SUBMIT      PROCESS(#PROCESS) FUNCTION(#PROCESS) EXCHANGE(#STRPRO)
**********
ELSE        /* If batch mode */
**********  -Get the list of processes-
USE         BUILTIN(GET_PROCESS_LIST) WITH_ARGS(#STRPRO)
            TO_GET(#PROLST #LAST #RETCOD)
**********  -If records found-
IF          COND('(#RETCOD *EQ OK) *OR (#RETCOD *EQ OV)')
USE         BUILTIN(COMPILE_PROCESS) WITH_ARGS(< etc >......
ELSE
MESSAGE      SGTXT('No files found .... Program ended')
RETURN
ENDIF
ENDIF
```

PUT_PROCESS_ACTIONS

Category: Process related built in functions

Description: Puts the definition of an action bar layout into the definition of the process definition currently being edited by the START_PROCESS_EDIT built in function.

Information passed into this built in function is subjected to the same editing and validation rules as the equivalent online facility provided in a full LANSA development environment.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	L	Req	Action Bar Definition List 1. This working list must contain at least 1 entry and may contain at most 18. For each entry in this list there must also be an entry in action bar definition list number 2.				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Lists 1 and 2 are conceptually just one list that must be passed as two real lists to get around the 256 byte list entry length limit that LANSA imposes.				

Each working list entry must have an aggregate length of 200 bytes and be formatted exactly as follows:

Bytes Fmt Ln/Dec

1-10 Alpha 10

Description: Action Bar Keyword

11-13 Alpha 3

Description: AB$OPT Code

14-15 Packed 3,0

Description: Number of Pull Down Options define in following array structures.

16-195 Alpha 9*20

Description: Array of 9 x alpha 20 Pull Down Option Descriptions

196-200 Alpha 5

Description: Spare area

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			for future expansion of function				
2	L	Req	Action Bar Definition List 2 This working list must contain at least 1 entry and may contain at most 18. For each entry in this list there must also be an entry in action bar definition list number 1. Lists 1 and 2 are conceptually just one list that must be passed as two real lists to get around the 256 byte list entry length limit that LANSA imposes. Each working list entry must have an aggregate length of 200 bytes and be formatted exactly as follows: **Bytes Frm Ln/Dec** 1-18 Alpha 9*2 Description: Array of 9 x alpha 2 Accelerator Function Key Numbers. 19-45 Alpha 9*3 Description: Array of 9 x alpha 3 PD$OPT identification values.				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			46-54 Alpha 9*1 Description: Array of 9 x alpha 1 initial availability flags. 55-144 Alpha 9*10 Description: Array of 9 x alpha 10 function names. Used to indicate name of function within this process that is to be invoked. 145-171 Alpha 9*3 Array of 9 x alpha 3 process attachment sequence numbers. Used to specify the sequence number of an "attached" process or function that is to be invoked. 172-200 Alpha 29 Description: Spare area for future expansion of function.				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

PUT_PROCESS_ATTACH

Category: Process related built in functions

Description: Puts a process and/or function "attachment" into the definition of the process definition currently being edited by the START_PROCESS_EDIT built in function.

Information passed into this built in function is subjected to the same editing and validation rules as the equivalent online facility provided in a full LANSA development environment.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Sequencing Number.	1	3	0	0
2	A	Req	Name of process to attach.	1	10		
3	A	Req	Name of function to attach.	1	7		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

PUT_PROCESS_ATTR

Category: Process related built-in functions

Description: Sets an attribute of a process definition that is being edited within an edit session previously started by using the START_PROCESS_EDIT built-in function.

Attributes set or returned by this built-in function have the same editing and validation rules as the equivalent online facility provided in a full LANSA development environment.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of attribute to change Valid attribute names are: DESC- Process Description TYPE- Process Type OPTCOM - Optimize for remote comms ENAGUI - Enable for GUI	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
2	A	Req	Value of that attribute is to be changed to. Allowable values are as follows: **For attribute DESC:** Any valid new process description up to 40 characters in length. **For attribute TYPE:** SAA/CUA ACT/BAR **For attribute OPTCOMM:** Y N **For attribute ENAGUI:** Y N	1	256		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

PUT_PROCESS_ML

Category: Process related built in functions

Description: Puts/updates a list of process multilingual attributes in different languages.

An edit session must be commenced by using the START_PROCESS_EDIT built in function prior to using this built in function.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain language code and process multilingual attributes. The function must supply a working list with an aggregate entry length of exactly 44 bytes. Each list entry sent should be formatted as follows:	44	44		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Bytes Description				
			1-4 Language code				
			5-44 Process description				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = multilingual attributes added / updated to the database successfully. ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

START_PROCESS_EDIT

Category: Process related built in functions

Description: Starts an "edit session" on the definition of a nominated LANSA process definition.

The edit session can be used to define a new process or alter an existing one.

The process definition is locked for exclusive usage throughout the edit session.

Only one process definition can be edited at one time (ie: it is not possible to concurrently edit two or more process definitions within the same job).

A process edit session should be terminated by using the END_PROCESS_EDIT built in function to ensure all locks/etc are released/shutdown in an orderly manner.

Any process edit session that receives a fatal error will have an END_PROCESS_EDIT command automatically issued.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of process to be edited	1	10		
2	A	Opt	Process description. Used for a new process only. Default value is PROCESS.	1	40		
3	A	Opt	Process menu type / style. Used for a new process only. Must be SAA/CUA or ACT/BAR. Default value is SAA/CUA.	7	7		
4	A	Opt	Initial public access. Used for a new process only. ALL, NORMAL or NONE allowed. Default value is NORMAL.	3	6		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = edit session commenced ER = fatal error detected	2	2		

Chapter 5. Function Related Built-In Functions

DELETE_FUNCTION

Category: Function related built in functions

Description: Deletes all details of the function currently being edited and ends the edit session against the function.

An edit session is commenced by using the built in function START_FUNCTION_EDIT.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No argument values.

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = function deleted and edit session was ended ER = fatal error detected	2	2		

END_FUNCTION_EDIT

Category: Function related built in functions

Description: Ends an active edit session on a LANSA function definition.

An edit session is commenced by using the built in function START_FUNCTION_EDIT.

A function edit session should be terminated by using the END_FUNCTION_EDIT built in function to ensure all locks/etc are released/shutdown in an orderly manner.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No argument values.

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code	2	2		
			OK = edit session ended				
			ER = fatal error detected				

GET_FUNCTION_ATTR

Category: Function related built in functions

Description: Gets an attribute of a function definition that is being edited within an edit session previously started by using the START_FUNCTION_EDIT built in function.

Attributes set or returned by this built in function have the same editing and validation rules as the equivalent online facility provided in a full LANSA development environment.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of attribute to get Valid attribute names are: DESC- Function description ONMENU - Display on Menu MENUSQ - Menu sequence Number TOTCMD - Total RDML	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			commands (including comments)				
			EDTSRC - Associated editing source				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code	2	2		
			OK = operation completed				
			ER = fatal error detected				
2	A	Req	Area in which to return the attribute. Allowable values are as follows	1	256		
			For attribute DESC:				
			The function description as A(40)				
			For attribute ONMENU:				
			Y				
			N				
			For attribute MENUSQ:				
			Valid number represented as 5 characters.				
			Range 00001 to 99999.				
			For attribute TOTCMD:				
			Valid number represented				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			as 4 characters. Range 0000 to 9999. **For attribute EDTSRC:** Character 3 editing source as the "source" field on the START_PROCESS_EDIT built in. Value LAN or blanks indicates last editor was standard online RDML editor				

GET_FUNCTION_INFO

Category: Function related built in functions

Description: Retrieves a list of function related information from the LANSA internal database and returns it to the calling RDML function in a variable length working list.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Process name.	1	10		
2	A	Req	Function name.	1	7		
3	A	Req	Type of function related information to retrieve. Valid types are FIELDS- Fields used by the function FILES- Files used by the function FUNCPANEL - LANSA Documentor panel layouts	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			FUNSTEXT- LANSA Documentor function MSL Diagram				
			FUNTNOTE- LANSA Documentor function MSL technical notes				
			FUNTABLE- LANSA Documentor function MSL tables				
			FUNXR3GL- LANSA Documentor called 3GL programs				
			FUNXRPRO- LANSA Documentor called processes				
			FUNXRFUN- LANSA Documentor called functions				
			FUNXRBIF- LANSA Documentor called built in functions				
			FUNCREP- LANSA Documentor report layouts functions				
			FUNXRSYV- LANSA Documentor system variables used				
			FUNXRMST- LANSA Documentor message text used				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			FUNXRMSI- LANSA Documentor predefined messages used				
			MLATTR- Multilingual attributes				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code	2	2		
			OK = list returned partially or completely filled. No more of this type of information exists for this function.				
			OV = list returned completely filled, but more of this type of information than could fit in the list exists.				
			NR = list was returned empty. Last entry in the list is returned as null.				
			ER = Function not found. Last entry in the list is returned as null.				
2	List	Req	Working list to contain process related information.	132	132		
			The calling RDML function must provide a				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			working list with an aggregate entry length of exactly 132 bytes. **For type FIELDS:** Each returned list entry is formatted as follows **Bytes Description** 1-10 Field name. 11-132 <<future expansion>> **For type FILES:** Each returned list entry is formatted as follows **Bytes Description** 1-10 Physical file name 11-20 Physical file library 21-30 Logical view name (blank if the physical file used) 31-132 <<future expansion>> **For type FUNXXXXXXXXXX:** (ex LANSA Documentor) Each returned list entry is formatted as follows				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Bytes Description 1-132 LANSA Documentor line. **For type MLATTR:** Each returned list entry is formatted as follows **Bytes Description** 1-4 Language code 5-44 Function description 45-132 <<future expansion>>				

GET_FUNCTION_LIST

Category: Function related built in functions

Description: Retrieves a list of processes associated functions and their descriptions from the LANSA internal database and returns them to the calling RDML function in a variable length working list.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Process name.	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain function information. The calling RDML function must provide a working list with an aggregate entry length of	60	60		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			exactly 60 bytes. Each returned list entry is formatted as follows: **Bytes Description** 1-7 Function name 8-47 Function description 48-60 <<future expansion>>				
2	A	Opt	Return code. OK = list returned partially or completely filled with function details. No more functions exist for this process. OV = list returned completely filled, but more functions than could fit in the list exist. Typically used to indicate "more" functions in page at a time style list displays. NR = list was returned empty. Last function in the list is returned as blanks. ER = Process not found. Last function in the list is returned as blanks.	2	2		

GET_FUNCTION_RDML

Category: Function related built in functions

Description: Returns the RDML code associated with a function into a working list.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No argument values.

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		
2	List	Req	Working list Name The working list must				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			have an aggregate entry length of 72 bytes where each entry is composed of - bytes 1-4, format Signed, length 4, decimal 0, description: Command sequence number - bytes 5-7, format Alpha, length 3, description: Command Label - bytes 8-17, format Alpha, length 10, description: Command - bytes19-72, format Alpha, length 55, description: Command Parameters				

Technical Notes

Commands that have more than 55 bytes of parameters are returned in multiple entries like this example

```
Seq  Lab Command       Parameters
0001     **********    This is a comment line
0002     SET_MODE      TO(*CHANGE)
0003 L32 GROUP_BY      NAME(#GROUP) FIELDS(#FIELD001 #FIELD002
0003                   #FIELD003 #FIELD004 #FIELD005 #FIELD006)
0004     DISPLAY       FIELDS(#GROUP)
0005     MENU
0006     **********    This is a comment line
```

PUT_FUNCTION_ATTR

Category: Function related built in functions

Description: Sets an attribute of a function definition that is being edited within an edit session previously started by using the START_FUNCTION_EDIT built in function.

Attributes set or returned by this built in function have the same editing and validation rules as the equivalent online facility provided in a full LANSA development environment.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of attribute to change Valid attribute names are: DESC- Function description ONMENU - Display on Menu MENUSQ - Menu sequence Number	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			EDTSRC - Identifier of Editor				
2	A	Req	Value of that attribute is to be changed to. Allowable values are as follows **For attribute DESC:** Any valid new function description up to 40 characters in length **For attribute ONMENU:** Y N **For attribute MENUSQ:** Valid number represented as 5 characters. Range 00001 to 99999. **For attribute EDTSRC:** 3 character editing "source" identifier. Must not be blank or LAN.	1	256		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

PUT_FUNCTION_ML

Category: Function related built in functions

Description: Puts/updates a list of function multilingual attributes in different languages.

An edit session must be commenced by using the START_FUNCTION_EDIT built in function prior to using this built in function.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Resale Notice: This built in function can be used in conjunction with other data dictionary built in functions to emulate or extend some of the facilities supplied in shipped LANSA systems. If this built in function is used in software intended for resale, then all screen panels and reports used must carry the name of the vendor organization, making clear that the software involved was not provided by ASPECT as part of the shipped LANSA system.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list to contain language code and function multilingual attributes. The function must supply a working list with an aggregate entry length of exactly 44 bytes. Each list entry sent should be formatted as follows **Bytes Description**	44	44		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			1-4 Language code 5-44 Function description				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = multilingual attributes added / updated to the database successfully. ER = argument details are invalid or an authority problem has occurred. In case of "ER" return code error message(s) are issued automatically.	2	2		

PUT_FUNCTION_RDML

Category: Function related built in functions

Description: Stores the RDML code associated with a function from a working list.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Working list Name The working list must have an aggregate entry length of 72 bytes where each entry is composed of - Bytes 1-4, format Signed, length 4, decimal 0, description: Command sequence number - Bytes 5- 7, format Alpha				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			3, description: Command Label - Bytes 8-17, format Alpha 10, description: Command -Bytes 19-72, format Alpha, length 55, description: Command Parameters				
2	A	Req	Nominated Editing Source Must not be blank or LAN. Used to "tag" edited RDML with last editor identifier.	3	3		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

Technical Notes

Commands that have more than 55 bytes of parameters must be formatted across multiple entries like this example

```
Seq  Lab Command  Parameters
0001     **********  This is a comment line
0002     SET_MODE TO(*CHANGE)
0003 L32 GROUP_BY NAME(#GROUP) FIELDS(#FIELD001 #FIELD002
0003              #FIELD003 #FIELD004 #FIELD005 #FIELD006)
0004     DISPLAY  FIELDS(#GROUP)
0005     MENU
0006     ********** This is a comment line
```

START_FUNCTION_EDIT

Category: Function related built in functions

Description: Starts an "edit session" on the definition of a nominated LANSA function definition.

The edit session can be used to define a new function or alter an existing one.

A function edit session must be started and ended within a process edit session on the parent (i.e.: owning) process. Multiple edit sessions on functions within the same process may be conducted serially (but not concurrently) within the same process edit session.

For example:

```
START_PROCESS_EDIT
      START_FUNCTION_EDIT
      << work with function A >>
       END_FUNCTION_EDIT
END_PROCESS_EDIT
```

or:

```
START_PROCESS_EDIT
      START_FUNCTION_EDIT
      << work with function A >>
      END_FUNCTION_EDIT

      START_FUNCTION_EDIT
      << work with function B >>
      END_FUNCTION_EDIT
END_PROCESS_EDIT
```

The function definition is locked for exclusive usage throughout the function edit session.

Only one function definition can be edited at one time (ie: it is not possible to concurrently edit two or more function definitions within the same job).

A function edit session should be terminated by using the END_FUNCTION_EDIT built in function to ensure all locks/etc are released/shutdown in an orderly manner.

Any function edit session that receives a fatal error will have an END_FUNCTION_EDIT and an END_PROCESS_EDIT operation automatically issued.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of function to be edited	1	7		
2	A	Opt	Function Description. Required for a new function only. Must not be blank. Default value is FUNCTION.	1	40		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
3	A	Opt	Initial public access. Required for a new function only. ALL, NORMAL or NONE allowed. Default value is NORMAL.	3	6		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = edit session commenced ER = fatal error detected	2	2		

Chapter 6. Template Related Built-In Functions

EXECUTE_TEMPLATE

Category: Template related built in functions

Description: Executes an application template to generate RDML function code into a working list.

Generated RDML code is appended to the END of the working list, so the list may need to be cleared before (via the CLR_LIST command) before invoking the application template.

Alternatively, multiple templates may be executed serially to progressively build up the resulting RDML function code.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of Application Template to be executed. This template must have been previously defined by the LANSA Application Template facilities.	1	10		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		
2	List	Req	Working list Name. The working list must be formatted as per the description contained in the GET_FUNCTION_RDML built in function description.				

TEMPLATE_@@ADD_LST

Category: Template related built in functions

Description: Allows a new field to be added to an application template list. The application template list is not cleared by this operation. If the field is already in the list it is replaced, otherwise it is added to the list.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of application template that will be used later via the EXECUTE_TEMPLATE built in.	1	10		
2	N	Req	Number of list that field is to be added to.	1	2	0	0
3	A	Req	Name of field to be added to application template list. Must be a valid field	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			in the LANSA data dictionary.				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

TEMPLATE_@@CANSNNN

Category: Template related built in functions

Description: Allows an application template character reply (@@CANSnnn) variable to be set before executing an application template via the EXECUTE_TEMPLATE built in function.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of application template that will be used later via the EXECUTE_TEMPLATE built in.	1	10		
2	N	Req	Number of @@CANSnnn variable that is to be set.	1	2	0	0
3	A	Req	Value that @@CANSnnn variable is to be set to.	1	74		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

TEMPLATE_@@CLR_LST

Category: Template related built in functions

Description: Clears an application template list.

Application template lists are widely used by application templates to control and organize the generation of RDML code. They should not be confused with working lists or browse lists which are RDML level constructs used in normal RDML application

This built in function allows access to an application template list, thus providing a means by which information can be set up for (and thus communicated to) an application template that will be later executed via the EXECUTE_TEMPLATE built in function to generate RDML code.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of application template that will be used later via the EXECUTE_TEMPLATE	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			built in.				
2	N	Req	Number of application template list to be cleared.	1	2	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

TEMPLATE_@@GET_FILS

Category: Template related built in functions

Description: From a nominated base file name this facility returns to the caller a list of all related files.

Functionally this built in function acts like the first step of the application template command @@GET_FILS in that from a nominated base file name it returns a list of files.

In a template, the command displays the resulting list to the user for selection. However, the built in function version returns to the list the calling RDML function.

In an application template the user selects files by entering a non-blank value beside them at the workstation. To perform this action from an RDML function use the TEMPLATE_@@SET_FILS built in.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

Please Note: This built in function has considerably more power than its online template equivalent @@GET_FILS.

The basic difference is in the ability of this function to extract a much more comprehensive file access route list. The online version prevents the extraction of the same underlying physical file more than once in the complete file list. However this built in functions relaxes this rule so that the same underlying physical file cannot be used more than once in any single access route "chain" or "path" starting from, and including, the base file.

Obviously this limit must be imposed to prevent "closed circuits" or "infinite loops" within the access route "chain" or "path".

It is strongly recommended that any developer who plans to use this function design a simple test function using this built in to extract and display the resulting file list from a nominated base file. This way the characteristics of this function can be much more easily examined and understood.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of application template that will be used later via the EXECUTE_TEMPLATE built in.	1	10		
2	A	Req	Primary or base file name.	1	10		
3	N	Opt	From file number. Default value is 1.	1	2		
4	N	Opt	To file number. Default value is 50.	1	2		
5	A	Opt	Physical Files Only. Must be Y or N. Default value is Y.	1	1		
6	A	Opt	1:1 Relationships Only. Must be Y or N. Default value is Y.	1	1		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		
2	List	Req	Returned list of related files First entry will be the base file. Working list must have an aggregate length of 40 bytes and is formatted as follows: - Bytes 1-1, format Alpha, length 1, description: Selection Flag. First entry returned as X. Others returned as blanks. - Bytes 2-11, format Alpha, length 10, description: File Name - Bytes 12-21, format Alpha, length 10, description: File Library. - Bytes 22-22, format Alpha, length 1, description: File Type. P = Physical. L = Logical. - Bytes 23-32, format Alpha, length 10, description: Underlying Physical File. If this file is a physical file then this name will be the same as				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			the entry file name. - Bytes 33-35, format Signed, length 3, decimal 0, description: Related file entry number. 1 = the file is directly related to the base file or is the base file. other = file is indirectly related to the base file via the file specified by this entry number. - Bytes 36-36, format Alpha, length 1, description: Nature of relationship between this file and the related file. P = Primary or Base File. O = One to One. M = Many. - Bytes 40-40, format Alpha, length 4, description: Future expansion.				

TEMPLATE_@@NANSNNN

Category: Template related built in functions

Description: Allows an application template numeric reply (@@NANSnnn) variable to be set before executing an application template via the EXECUTE_TEMPLATE built in function.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of application template that will be used later via the EXECUTE_TEMPLATE built in.	1	10		
2	N	Req	Number of @@NANSnnn variable that is to be set.	1	2	0	0
3	A	Req	Value that @@NANSnnn variable is to be set to.	1	15	0	5

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

TEMPLATE_@@SET_FILS

Category: Template related built in functions

Description: Allows file(s) from a list of files previously built by the TEMPLATE_@@GET_FILS built in to be "selected" for use within an application template that will be executed later.

Functionally this built in function acts like the second step of the application template command @@GET_FILS in that a "selection" of files is made and set up for use by the application template.

In an application template the user selects files by entering a non-blank value beside them at the workstation. To perform this action from an RDML function use this built in function.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of application template that will be used later via the EXECUTE_TEMPLATE built in.	1	10		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
2	List	Req	List of related files previously built by the TEMPLATE_ @@GET_FILS command with the "selection" flag set to a non-blank value to indicate a file that is to be selected. The first entry in the list is the base file and it must always be selected. Working list must have an aggregate length of 40 bytes and is formatted as follows: - Bytes 1-1, format Alpha, length 1, description: Selection Flag. First entry returned as X. Others returned as blanks. - Bytes 2-11, format Alpha, length 10, description: File Name - Bytes 12- 21, format Alpha, length 10, description: File Library - Bytes 22-22, format Alpha, length 1, description: File Type. P = Physical. L = Logical. - Bytes 23-32, format Alpha, length 10, description: Underlying Physical File. If this file is a physical file then this name will be the same as				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			the entry file name. - Bytes 33-35, format Signed, length 3, decimal 0, description: Related file entry number. 1 = the file is directly related to the base file or is the base file. other = file is indirectly related to the base file via the file specified by this entry number. - Bytes 36-36, format Alpha, length 1, description: Nature of relationship between this file and the related file. P = Primary or Base File. O = One to One. M = Many. - Bytes 40-40, format Alpha, length 4, description: Future expansion.				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

TEMPLATE_@@SET_IDX

Category: Template related built in functions

Description: Allows an application template index variable to be set to a nominated value before executing an application template via the EXECUTE_TEMPLATE built in function.

This built in can only be used against a function that has been previously placed into an edit session by using the START_FUNCTION_EDIT built in.

SAA Note: Only use this built-in function in applications that are to fully execute on one of the LANSA development platforms.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML functions.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Name of application template that will be used later via the EXECUTE_TEMPLATE built in.	1	10		
2	A	Req	Identifier of index variable that is to be set.		2		
3	N	Req	Value that index variable is to be set to.	1	2	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = operation completed ER = fatal error detected	2	2		

Chapter 7. Workfolder Application Facility/400 Built-In Functions

WAF_CRTOBJ

Category: Workfolder Application Facility/400

Description: This built in function creates a document record and returns to the caller the object path in which the new document must reside.

It is then your responsibility to place an object in the specified location.

This built in function uses the Workfolder Application Facility API called "CRTOBJ". Refer to the *Workfolder Application Facility Application Programming Interfaces* guide for more information about this API. *(Document Number GC38-3034).*

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Opt	System ID	1	1		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	System ID	1	1		
2	A	Req	Document ID	12	12		
3	A	Req	AS/400 folder	9	9		

No	**Type A/N**	**Req/ Opt**	**Description**	**Min Len**	**Max Len**	**Min Dec**	**Max Dec**
4	A	Req	AS/400 Subdirectory to place the object	12	12		
5	N	Req	Return code	4	6	0	0

WAF_DEFFOLD

Category: Workfolder Application Facility/400

Description: This built in function creates a new empty case.

This built in function uses the Workfolder Application Facility API called "DEFFOLD". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	**Type A/N**	**Req/ Opt**	**Description**	**Min Len**	**Max Len**	**Min Dec**	**Max Dec**
1	A	Req	External case ID	1	40		
2	A	Req	Case description	1	26		
3	A	Opt	Queue in which case is placed	1	10		
4	A	Opt	Priority when queued	1	1		

Return Values

No	**Type A/N**	**Req/ Opt**	**Description**	**Min Len**	**Max Len**	**Min Dec**	**Max Dec**
1	A	Req	Case ID	9	9		
2	N	Req	Return code	4	6	0	0

WAF_DEFSTOBJ

Category: Workfolder Application Facility/400

Description: This built in function adds a document to a case.

This built in function uses the Workfolder Application Facility API called "DEFSTOBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Document ID	1	12		
2	A	Req	Case ID	1	9		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_DLTEOBJ

Category: Workfolder Application Facility/400

Description: This built in function places a delete in the delete request file.

The Workfolder Application Facility must be activated to actually delete the document from DASD.

This built in function uses the Workfolder Application Facility API called "DLTEOBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Document ID	12	12		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_FETCHOBJ

Category: Workfolder Application Facility/400

Description: This built in function writes a retrieve request to the optical request file.

The Workfolder Application Facility must be activated to actually retrieve the document from optical storage.

This built in function uses the Workfolder Application Facility API called "FETCHOBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Document ID	12	12		
2	A	Opt	Priority for retrieve request	1	1		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_FINDOBJ

Category: Workfolder Application Facility/400

Description: This built in function locates all documents within a specified case.

The maximum document Identifiers returned with any one use of this function is 10.

The maximum number of documents in a case that can be retrieved is 300.

This built in function uses the Workfolder Application Facility API called "FINDOBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Case ID	1	9		
2	A	Opt	Starting document	1	12		
3	N	Opt	Number of documents to retrieve (up to 10)	2	2	0	0

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	List	List of documents	12	12		
2	A	Req	Last document in list	12	12		
3	N	Req	Documents found	2	2	0	0
4	N	Req	Return code	4	6	0	0

WAF_INDEXOBJ

Category: Workfolder Application Facility/400

Description: This built in function completes the following information to a document:

- Document description
- Document type

The information is added (first time) or changed.

If the length of the document description field is 0, the field is not changed.

If the length is less than 40, the description is padded with trailing blanks.

Document type must be a valid document type.

Although both 'Document description' and 'Document type' are optional at least one must be chosen.

This built in function uses the Workfolder Application Facility API called "INDEXOBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Document ID	1	12		
2	A	Opt	Document type	1	8		
3	A	Opt	Document description	1	40		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_PPRTOBJ

Category: Workfolder Application Facility/400

Description: This built in function sends the formatted print command to the image workstation.

Attributes set or returned by this built in function have the same editing and validation rules as the equivalent online facility provided in a full LANSA development environment.

This built in function uses the Workfolder Application Facility API called "PPRTOBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	**Type A/N**	**Req/ Opt**	**Description**	**Min Len**	**Max Len**	**Min Dec**	**Max Dec**
1	A	Req	Document ID	1	12		
2	A	Opt	User name to print on the separator page	1	8		
3	A	Opt	Terminal name to print on the separator page	1	8		
4	A	Opt	End of print parameter Values: Y - Yes N - No Default 'N'	1	1		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_PRINTOBJ

Category: Workfolder Application Facility/400

Description: This built in function queues print requests to an image workstation printer.

To print documents you must start 'Demand Print Server'.

This built in function uses the Workfolder Application Facility API called "PRINTOBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	**Type A/N**	**Req/ Opt**	**Description**	**Min Len**	**Max Len**	**Min Dec**	**Max Dec**
1	A	Req	Document ID	12	12		
2	A	Req	Print queue name	1	8		
3	A	Req	Terminal name to print on the separator page	1	8		
4	A	Req	User name to print on the separator page	1	8		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_RESET

Category: Workfolder Application Facility/400

Description: This built in function resets the image workstation.

This built in function does not reset the values set by the previous WAF_SETC built in function.

This built in function must be run from the workstation session configured to run the 'AS/400 Workfolder Application Facility'.

This built in function uses the Workfolder Application Facility API called "RESET". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Opt	Reset terminal Values: Y - Yes N - No Default 'Y'	1	1		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_RMVOBJ

Category: Workfolder Application Facility/400

Description: This built in function removes a document from a case.

The document is removed from the case but is not physically removed from DASD.

This built in function uses the Workfolder Application Facility API called "RMVOBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Document ID	1	12		
2	N	Req	Case ID to be removed from.	1	9		
3	A	Opt	Reject queue Values: 0 - Remove from queue 1 - On queue (re-index) 2 - On queue (delete) Default '0'	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
4	A	Opt	User ID to be placed in reject queue record	1	8		
5	A	Opt	Compatibility mode Values: Y - Do not use N - Use Default '0'	1	1		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_SCAN

Category: Workfolder Application Facility/400

Description: This built in function provides the following functions:

- Generates unique name for scanned document,
- Determines the DASD path for the next document using Workfolder Application Facility balancing,
- Sends the formatted scan command to the image workstation and waits for the workstation return code,
- Returns the return code to the calling program,
- Returns the index value to the calling program if one was requested,
- Adds a document record to the Workfolder Application Facility database file upon normal completion.

This built in function uses the Workfolder Application Facility API called "SCAN". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Description of document to be scanned	1	40		
2	A	Opt	Scan overlap indicator Values: Y - Yes	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			N - No E - End of scan overlap Default 'N'				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0
2	A	Opt	Index value returned with the scanned image.	50	50		
3	N	Opt	Number of pages in the document scanned.	3	3	0	0
4	N	Opt	Size of the document in bytes	7	7	0	0
5	A	Opt	Document ID	12	12		

WAF_SENDOBJ

Category: Workfolder Application Facility/400

Description: This built in function displays an image document or list of image documents on an image workstation.

The document to display must exist in the Workfolder Application Folder database.

This built in function uses the Workfolder Application Facility API called "SENDOBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	List	Req	Table of documents to display Up to 10 entries can be specified	12	12		
2	A	Opt	Reset terminal before the document is displayed Values: Y - Yes N - No Default 'N'	1	1		
3	A	Opt	Specifies the session in which the workstation is left after the document is	1	1		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			displayed Values: 0 - Image session 1 - Session one 2 - Session two Default '0'				
4	A	Opt	Controls the creation and presentation of annotations and masks. 8 bytes can be specified for each document specified in argument 1, with the following significance: - Byte 1. Value Y = Ignore Masks, other = Do not ignore masks - Byte 2. Value Y = Document to be printed without masks, - Byte 3. Value Y = Document can be hidden from display. - Byte 4. Value Y = Document can be masked. - Byte 5. Value Y = Document can be modified. - Byte 6. Value Y = Document can be printed with annotations.				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			- Byte 7. Value Y = Document can be displayed with annotations. - Byte 8. Value Y = Document can be annotated. Default is *BLANK.				

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0
2	A	Req	List entry in error	6	6		

WAF_SETC

Category: Workfolder Application Facility/400

Description: This function is used to send the set configuration command to the ImagePlus Workstation Program.

It is used to set the workstations scanning parameters and forms path.

Notes: This built in function must be used before any forms can be viewed or printed.

This built in function must be run from the workstation session configured to run the 'AS/400 Workfolder Application Facility'.

This built in function uses the Workfolder Application Facility API called "SETC". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

The 'Annotations and masks' (WPO) and the 'Level of support' (NTL) arguments are available for Version 2 of WAF/400 only. They are ignored for previous versions.

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	**Type A/N**	**Req/ Opt**	**Description**	**Min Len**	**Max Len**	**Min Dec**	**Max Dec**
1	A	Req	Parameter to set Valid values are: SFP - Folder path SOS - Scan overlap selector	3	3		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			CDR - Check digit routine				
			ACI - Alphabetic character index				
			SCI - Special character index				
			IVK - Index value keying				
			RTI - Reset terminal indicator				
			MAX - Maximum length if index				
			MIN - Minimum length if index				
			WPO - Annotations and masks (V2 only)				
			NTL - Level of support (V2 only)				
2	A	Req	Value to set	1	44		

Argument 2 - Value to Set

Parameter	Possible Value	Length
Scan overlap selector (SOS)	Y, N	1
Check digit routine(CDR)	Y, N	1
Alphabetic character index option (ACI)	Y, N	1
Special character index option (SCI)	Y, N	1
Index value keying option (IVK)	Y, N	1
Reset terminal indicator (RTI)	Y, N	1

Parameter	Possible Value	Length
Maximum length if index field (MAX)	0 to 50	2
Minimum length if index field (MIN)	0 to 50	2
Annotations and masks (WPO)	A, M, B	1
Level of Support (NTL)	Y, N (9 flags)	9

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_STOROBJ

Category: Workfolder Application Facility/400

Description: This built in function puts the formatted store request into the optical request file, increments the active request count, and returns to caller.

The Workfolder Application Facility optical storage processor must be activated to store the documents to optical.

This built in function uses the Workfolder Application Facility API called "STOROBJ". Refer to the 'Workfolder Application Facility Application Programming interfaces' guide for more information about this API. (Document Number GC38-3034).

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Document ID	12	12		
2	A	Opt	System ID	1	1		
3	A	Opt	Priority field for store requests.	1	1		

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	N	Req	Return code	4	6	0	0

WAF_USE

Category: Workfolder Application Facility/400

Description: This built in function tells LANSA that you are going to use the AS/400 Workfolder Application Facility.

It is a trigger for LANSA at compile time to include the necessary data definitions into the resulting program object.

Notes: This built in function must be used or the Workfolder Application Facility will not work as expected.

SAA Note: Only use this built in function in applications that are to fully execute under the control of the AS/400 operating system OS/400 and that have the IBM supplied product 'Workfolder Application Facility'.

Arguments

No argument values.

Return Values

No return values.

Chapter 8. Authority-Related Built-In Functions

GET_AUTHORITIES

Category: Authority related built in functions

Description: Retrieves a list of authorities to LANSA objects and returns it to the calling RDML function in a variable length working list.

Special Note: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML applications.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Object name.	1	10		
2	A	Req	Object extension.	1	10		
3	A	Req	Object type. Valid types are: AT - Application template DF - Field FD - File PD - Process PF - Function P# - Partition SV - System variable	2	2		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			MT - Multilingual variable				
4	A	Opt	User name.	1	10		

Dependencies

If any of Object type, Object name or Object extension are specified, then all three must be specified according to the following table.

Object Type	Object Name	Object Extension
AT	template name	*blank
DF	field name	*blank
FD	file name	*blank, *LIBL, library name
PD	process name	*blank
PF	process name	function name
P#	partition name	*blank
SV	positions 1-10 of system variable name	positions 11-20 of system variable name
MT	positions 1-10 of multilingual variable name	positions 11-20 of multilingual variable name

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = list returned partially or completely filled. No more authorities that match the arguments exist OV = list returned completely filled, but more authorities that match the arguments than could fit in the list exist. NR = No authorities that match the arguments exist. Last entry in the list is returned as null. ER = Error in the arguments passed. Last entry in the list is returned as null.	2	2		
2	List	Req	Working list of authorities. If Object type, name and extension are specified but not User, then as many authorities of users to the object as fit in the list will be returned. If User is specified but not Object type, name and extension, then as many authorities of the user to different objects as fit in the list will be returned.	70	70		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			If Object type, name, extension and User are specified, and the user is specifically authorized to the object then the authority of the user to the object will be returned in the list. If the user is not specifically authorized to the object no authorities will be returned. The calling RDML function must provide a working list with an aggregate entry length of exactly 70 bytes. **Bytes Description** 1-10 Object name 11-20 Object extension 21-22 Object type (see above for object type explanation) 23-32 User name 33-52 Access rights This is a string of 2 character codes representing the different access rights that the user has to the object. The individual access rights are:				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			UD - Use Definition MD - Manage Definition DD - Existence of Definition DS - Data - Display AD - Data - Add CH - Data - Change DL - Data - Delete If the entire string is blank then the user has had their access rights to the object revoked. 53-70 <<future expansion>>				

SET_AUTHORITY

Category: Authority related built in functions

Description: Sets the authority of a user to a LANSA object. The caller of this built in function must have management rights (MD) to the LANSA object.

Special Notes: This built in function provides access to very advanced facilities that basically allow RDML functions to construct new RDML applications.

This is a very specialized area that requires very good knowledge of the LANSA product. Use of this built in function in a normal "commercial" application (e.g.: Order Entry) is not normal and should not be attempted

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.

This condition also applies to the caller of SET_AUTHORITY changing their own authorities to objects.

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 the Technical Guide for details of the system definition data area DC@A01, its layout and how to change flags within it.

Arguments

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Object name.	1	10		
2	A	Req	Object extension.	1	10		
3	A	Req	Object type.	2	2		

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			Valid types are:				
			AT - Application template				
			DF - Field				
			FD - File				
			PD - Process				
			PF - Function				
			P# - Partition				
			SV - System variable				
			MT - Multilingual variable				
4	A	Req	User name.	1	10		
5	A	Req	Access rights	1	20		
			This is a string of 2 character codes representing the different access rights that the user is to have.				
			The individual access rights are:				
			UD - Use Definition				
			MD - Manage Definition				
			DD - Existence of Definition				
			DS - Data - Display				
			AD - Data - Add				
			CH - Data - Change				
			DL - Data - Delete				

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
			If the entire string is blank then the user is to have their access rights to the object revoked. If the string has the special value '*DELETE' then the user's authority is to be deleted, thus causing their rights to revert back to their associated group profile or *PUBLIC.				

Dependencies

Object Type	Object Name	Object Extension
AT	template name	*blank
DF	field name	*blank
FD	file name	*blank, *LIBL, library name
PD	process name	*blank
PF	process name	function name
P#	partition name	*blank
SV	positions 1-10 of system variable name .	positions 11-20 of system variable name
MT	positions 1-10 of multilingual variable name	positions 11-20 of multilingual variable name

Return Values

No	Type A/N	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Return code OK = The authority of the user to the LANSA object has been set. ER = Error occurred in setting the authority of the user to the object.	2	2		

Index

Tell us what you think of this guide

To: **LANSA Development Manager**
Fax: **+61 (2) 9957 2657**
Email: ***lansamarketing@aspect.com.au***

LANSA/AD Specialized Built-In Functions Guide

We hope you found this guide useful and informative. If you like what we've done, please let us know, if not, please tell us why, so that can make the guide better.

✓	Yes	No	No opinion
Does this guide meet your needs?			
Have you found the information accurate?			
Do you find the contents well organised?			
Is it easy to understand?			

What do you think we should do to improve this guide?

____

____

____

____

____

____

____

Your Name:____

Company:____

Tel: ____________________ Fax: ____________________

Email: ____

Thank you for taking the time to fill out this response.

Fold Under

If you would rather send your reponse by post,
or have your local LANSA distributor send it for you,
please fold on the dotted line and staple where indicated.

....

LANSA Development Manager
Aspect Computing Pty Ltd
Level 11,
122 Arthur Street
North Sydney
Australia
2060

....

Back - fold over