

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

---

**STANDARD ECMA - 187**

**OFFICE DOCUMENT ARCHITECTURE (ODA)**

**APPLICATION PROGRAMMING INTERFACE**

**APPLICATION PROFILE INTERFACE FOR  
HANDLING COMPOUND DOCUMENTS**

**Volume 2**

- 3      **References**
- 3.4      **Functions**

June 1993

Free copies of this document are available from ECMA,  
European Computer Manufacturers Association,  
114 Rue du Rhône - CH-1204 Geneva (Switzerland)

Phone: +41 22 735 36 34 Fax: +41 22 786 52 31  
X.400: C=ch, A=ecma, P=ecma, O=gsmvnet,  
OU=ecma, S=helpdesk  
Internet: [helpdesk@ecma.ch](mailto:helpdesk@ecma.ch)

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

---

**STANDARD ECMA - 187**

**OFFICE DOCUMENT ARCHITECTURE (ODA)**

**APPLICATION PROGRAMMING INTERFACE**

**APPLICATION PROFILE INTERFACE FOR  
HANDLING COMPOUND DOCUMENTS**

**Volume 2**

<b>3</b>	<b>References</b>
<b>3.4</b>	<b>Functions</b>

June 1993

## Brief History

In 1985, ECMA/TC29 published Standard ECMA-101, Office Document Architecture, in order to facilitate the interchange of documents.

In the meantime, work had started in ISO and CCITT, resulting in the preparation of ISO 8613 "Office Document Architecture (ODA) and interchange format". Experts of TC29 participated to the work, and acted as editors of most of the parts of the ISO and CCITT documents.

The second edition of Standard ECMA-101 was prepared in 1988 by ECMA/TC29 in order to align ECMA-101 with the last ISO and CCITT publications (ISO 8613 and CCITT Recommendations in the T.410 series).

Work started in 1991 in a group of ECMA/TC29 to prepare application programming interfaces for use with Standard ECMA-101. Two API specifications were agreed for handling compound documents, one at constituent level and one at application profile level (this Standard). Both specifications are aligned with the current ECMA, ISO and CCITT standards, and with the ODA Document Application Profiles approved by ISO (FOD11, FOD26 and FOD36). Representatives of ISO/IEC JTC1/SC18 attended the meetings of ECMA-TC29.

---

This ECMA Standard is divided in three volumes, as follows:

**Volume 1**

- 1 General
- 2 Introduction
- 3 References
  - 3.1 Data Types
  - 3.2 Constants
  - 3.3 Error Handling

**Volume 2**

- 3.4 Functions

**Volume 3**

Annexes

This is volume 2.

## Table of contents

3.4	Functions	1
	dia_add_content	2
	dia_append_char_set	4
	dia_change_writing_mode	7
	dia_convert_content	9
	dia_create_document	11
	dia_create_generic_basic_font	14
	dia_create_generic_body_frame	16
	dia_create_generic_comm_entity	18
	dia_create_generic_front_area	20
	dia_create_generic_frame	22
	dia_create_generic_frame_seq	25
	dia_create_generic_head_footer	28
	dia_create_generic_pageset	31
	dia_create_generic_staking	33
	dia_create_generic_synchronized	36
	dia_create_gen_log_entity	38
	dia_create_gen_log_entity_seq	41
	dia_create_specific_entity	44
	dia_delete_document	48
	dia_duplicate_specific_entity	50
	dia_error_class	53
	dia_error_number	54
	dia_failed	55
	dia_find_first_entity	56
	dia_find_next_entity	58
	dia_find_profile	60
	dia_fix_properties	61
	dia_get_char_set_count	63
	dia_get_choice	65
	dia_get_construct_error	68
	dia_get_content	70
	dia_get_content_length	73
	dia_get_entity_handle	75
	dia_get_entity_type	78
	dia_get_fix_error	79
	dia_get_index_char_set	81
	dia_get_index_char_set_count	84
	dia_get_index_choice	86
	dia_get_index_entity	89
	dia_get_index_integer	92
	dia_get_index_length	95
	dia_get_index_string	98
	dia_get_index_structured	101
	dia_get_integer	104

<code>dia_get_interp_error</code>	107
<code>dia_get_length</code>	109
<code>dia_get_nth_char_set</code>	112
<code>dia_get_ola_read_error</code>	115
<code>dia_get_ola_write_error</code>	117
<code>dia_get_other_parent</code>	119
<code>dia_get_size</code>	121
<code>dia_get_spec_char_set_count</code>	124
<code>dia_get_spec_choice</code>	126
<code>dia_get_spec_content</code>	128
<code>dia_get_spec_content_length</code>	131
<code>dia_get_spec_entity_handle</code>	133
<code>dia_get_spec_index_choice</code>	135
<code>dia_get_spec_index_entity</code>	137
<code>dia_get_spec_index_integer</code>	139
<code>dia_get_spec_index_length</code>	141
<code>dia_get_spec_index_string</code>	143
<code>dia_get_spec_index_structure</code>	146
<code>dia_get_spec_integer</code>	148
<code>dia_get_spec_length</code>	150
<code>dia_get_spec_nth_char_set</code>	152
<code>dia_get_spec_size</code>	155
<code>dia_get_spec_status</code>	157
<code>dia_get_spec_string</code>	159
<code>dia_get_spec_tab_stop</code>	161
<code>dia_get_spec_tab_stop_strings</code>	163
<code>dia_get_status</code>	165
<code>dia_get_string</code>	168
<code>dia_get_tab_stop</code>	171
<code>dia_get_tab_stop_strings</code>	173
<code>dia_init_toolkit</code>	175
<code>dia_instantiate_spec_log</code>	177
<code>dia_link_generic_entity</code>	180
<code>dia_link_generic_entity_seq</code>	183
<code>dia_locate_gm_entity</code>	186
<code>dia_make_frame_equiv</code>	190
<code>dia_move_to_child_spec_entity</code>	193
<code>dia_move_to_parent_spec_entity</code>	195
<code>dia_move_to_root_entity</code>	197
<code>dia_move_to_sibling_spec_entity</code>	199
<code>dia_read_document</code>	201
<code>dia_read_generic_doc</code>	204
<code>dia_read_writing_mode</code>	207
<code>dia_register_callback_function</code>	209
<code>dia_set_entity_handle</code>	211
<code>dia_set_index_entity</code>	213
<code>dia_set_index_integer</code>	216

dla_set_index_null_value	219
dla_set_index_string	222
dla_set_index_structured	225
dla_set_integer	228
dla_set_null_value	230
dla_set_status	232
dla_set_string	235
dla_set_tab_stop	237
dla_term_toolkit	239
dla_unix_properties	240
dla_version	242
dla_write_document	243





## 3.4 Functions

This subclause describes the DAP API functions. They appear in alphabetical order. The description of each function contains the following information:

- The name of the function and a brief description of its purpose
- The C binding of the routine, expressed as a C function prototype
- A detailed specification of the function's behaviour, expressed in the description of the arguments and results
- A description of the arguments to the function
- A description of the value returned by the function
- A description of any errors that may be generated by the function

Arguments that receive a return value from the function are indicated in the Arguments section for each function by [write] following the argument name.

Descriptions of most parameters are included in the individual function descriptions.

In the subset of property getting functions that do not have the syllable *\_spec\_* in their names, that is those that perform the defaulting process to derive a value, two parameters are present that do not occur elsewhere. These parameters are *derived\_location* and *derived\_entity\_handle*. They correspond to the equivalent parameters in functions of the ODA API and return similar information. The values and meanings of *derived\_location* are as follows:

**DLA\_C\_DEF\_ENTITY**

Within the specified entity.

**DLA\_C\_DEF\_STYLE**

Within a style referenced by the specified entity.

**DLA\_C\_DEF\_CLASS**

Within the generic entity referenced by the specified entity, of which it is an instance.

**DLA\_C\_DEF\_CLASS\_STYLE**

Within a style referenced by the generic entity that is referenced by the specified entity.

**DLA\_C\_DEF\_REF\_CLASS**

Within a generic entity in a resource document referenced by the generic entity that is referenced by the specified entity.

**DLA\_C\_DEF\_REF\_STYLE**

Within a style referenced by a generic entity in a resource document that is referenced by the generic entity that is referenced by the specified entity.

**DLA\_C\_DEF\_VAL\_LIST**

Within a default value list at a higher level of the hierarchical structure of the document.

**DLA\_C\_DEF\_DAP\_DEF**

Within a list of default values in the document profile, representing a default value permitted by a document application profile.

**DLA\_C\_DEF\_STD\_DEF**

Within ISO 8613.

Except for the last value (*DLA\_C\_DEF\_STD\_DEF*), the parameter *derived\_entity\_handle* returns the entity in which the value was found to be specified. In this exception case the value returned is *DLA\_C\_NULL\_HANDLE*. In a few exceptional cases (see 2.2.3.3), this combination may be returned together with a *property\_states* value of *DLA\_C\_PROP\_UNSPECIFIED*.

## dla\_add\_content

## dla\_add\_content

Adds the data in a buffer to the content information of an entity.

### C Synopsis

```

dla_status      dla_add_content(
    dla_document_handle    document_handle,
    dla_entity_handle      entity_handle,
    dla_length             content_offset,
    dla_length             buffer_length,
    dla_ptr               buffer_ptr);

```

### Description

The `dla_add_content` function adds the data in the specified buffer to the content information of an entity. The `buffer_length` and `buffer_ptr` arguments indicate the length and location of the supplied buffer. The `content_offset` argument indicates the starting offset within the content information at which the supplied data is to be stored. If content information already exists at the specified offset, it is overwritten with the new data.

The specified entity must be of one of the following types:

DLA_C_GEN_BODY_GEO	(mode-2)
DLA_C_GEN_BODY_RAS	(mode-2)
DLA_C_GEN_BODY_TEX	(mode-2)
DLA_C_GEN_COMMON_GEO	
DLA_C_GEN_COMMON_RAS	
DLA_C_GEN_COMMON_TEX	
DLA_C_GEN_ENTRY_GEOM	(mode-2)
DLA_C_GEN_ENTRY_RAS	(mode-2)
DLA_C_GEN_ENTRY_TEX	(mode-2)
DLA_C_GEN_FOOTNOTE_TEX	(mode-2)
DLA_C_SPEC_BODY_GEO	
DLA_C_SPEC_BODY_RAS	
DLA_C_SPEC_BODY_TEX	
DLA_C_SPEC_ENTRY_GEOM	
DLA_C_SPEC_ENTRY_RAS	
DLA_C_SPEC_ENTRY_TEX	
DLA_C_SPEC_FOOTNOTE_TEX	

No special action is required for different content architectures. The application must ensure that the data supplied is of the right type for the specified entity.

## Arguments

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**content\_offset**

The starting octet at which the supplied data is to be stored as the content information. A value of zero is specified when adding the first buffer of data to the content information.

**buffer\_length**

The length of the data in the specified buffer expressed as the number of octets.

**buffer\_ptr**

The buffer containing the data to be stored as content information.

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

**[DLA\_E\_INV\_OFFSET]**

The offset is not valid, that is, greater than the length of the content information.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## `dia_append_char_set`

## `dia_append_char_set`

Specifies an initial or an additional graphic character set to be used in a character content entity, or for another non-basic graphic character set for use in the profile. The function is for the values that can be specified in the entity, and for the properties: Graphic character sets (character content defaults), Profile character sets, Comments character sets and Alternative repr char sets, but not for Non-basic char presentation features, which are automatically set as necessary by this function on behalf of the application.

### C Synopsis

```

dia_status      dia_append_char_set (
    dia_document_handle      document_handle,
    dia_entity_handle        entity_handle,
    dia_property_code         property_code,
    dia_octet_string          character_set_id,
    dia_length                string_length,
    dia_integer               code_area,
    dia_integer               character_set_type);
  
```

### Description

The `dia_append_char_set` function specifies an initial or an additional graphic character set that can be designated for a basic component within the document. The function can also be used for the non-basic values specified in the profile.

The function is for the values of the property Graphic character sets that are specified in the entity, and for profile properties: Graphic character sets (character content defaults), Profile character sets, Comments character sets and Alternative repr char sets, but not for Char presentation features, which are set automatically.

The function is not for the profile property Non-basic char presentation features because this is set automatically when non-basic values of the property Graphic character sets are established using this function.

All of the properties that can be set by using this function can also be set as octet strings, but if they are set in this way, the Non-basic char presentation features property is not updated automatically.

The Graphic character set is specified by the final byte identifier, the character set type, and the code area. The final byte identifier is the first byte of the argument `character_set_id`. The character set type is the value of the argument `character_set_type` which specifies whether the character set contains 94 or 96 characters and is single-byte or multi-byte.

The argument `code_area` specifies one of the sets G0, G1, G2 or G3.



## Arguments

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**property\_code**

A symbolic constant representing the property being operated on.

**character\_set\_id**

The final character identifier of a graphic character set.

**string\_length**

An integer that identifies the length of the final character identifier string.

**code\_area**

An integer that identifies the code area for the graphic character set. The following values are applicable:

DLA\_C\_G0

DLA\_C\_G1

DLA\_C\_G2

DLA\_C\_G3

**character\_set\_type**

An integer whose value specifies whether the character set consists of 94 or 96 characters and is single-byte or multi-byte. The following values are defined for use:

DLA\_C\_SIN\_94

DLA\_C\_SIN\_96

DLA\_C\_MUL\_94

DLA\_C\_MUL\_96

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_CODE]**

The entity is not valid for this function.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

## `dla_append_char_set`

### **[DLA\_E\_INV\_CODE\_AREA]**

The combination of code area and character set descriptions is not permitted by the DAP or by ISO 2022.

### **[DLA\_E\_VAL\_OUT\_OF\_RNG]**

Either `code_area` or `character_set_type` is outside the range of values permitted by the DAP, or the final byte identifier is outside the range of values permitted by ISO 2022.

### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_change\_writing\_mode

Changes the mode in which a specific document is being written.

## C Synopsis

```
dia_status      dia_change_writing_mode(
    dia_document_handle    document_handle,
    dia_document_options    document_option);
```

## Description

The `dia_change_writing_mode` function changes the mode in which a specific document is being written.

The only change that is permitted is from mode-1 (DLA\_C\_AUTO\_GEN) to mode-2 (DLA\_C\_MODE\_2). This change is not permitted if any FOD36 clumps of generic logical entities are in an unfinished state. This is because the rules for finishing such clumps are different in the two states, and the transition would be too difficult to control.

## Arguments

**document\_handle**

The handle of the document being operated on.

**document\_option**

An enumerated value that allows the application to specify a new option to control the processing of the document. The only new value that is allowed is as follows:

Option	Meaning
DLA_C_MODE_2	Application creates generic logical structure

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## `dla_change_writing_mode`

### Errors

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_OLD\_MODE]**

The original mode of the document is not one that can be changed.

**[DLA\_E\_NEW\_MODE]**

The new mode is not one to which the document can be changed.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

**[DLA\_E\_CLUMPS\_UNFINISHED]**

One or more clumps in the generic logical structure are not finished. (FOO36 documents only.)

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## dia\_convert\_content

Converts the content information of an entity from one type of coding to another.

## C Synopsis

```
dia_status      dia_convert_content(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_conversion_type     type_of_conversion);
```

## Description

The dia\_convert\_content function converts the content data in the given basic entity. The content information is overwritten with the new data.

This is the only modification that is allowed with read only documents or external and resource documents that have been read from CDIF. There is no error status of DLA\_E\_DOC\_READ\_ONLY for this function.

The given basic entity must be the one that owns the content data and no reference to any other entity, such as one in a resource document, occurs.

The only attribute that is modified to correspond with the new content encoding is DLA\_C\_CP\_TY\_COD, which is changed if raster content is converted. For converted character content the application is responsible for altering the character attributes, or for interpreting them correctly in the case of read-only documents. In the case of formatted processable form documents the character content portions are, in general, shared between basic specific logical entities and specific layout entities, whereas conversion is performed only for the indicated basic object. Only if conversion is requested for all basic specific layout entities can the conversion of all such shared content portions be guaranteed.

The specified entity must be of one of the following types:

```
DLA_C_SPEC_BODY_TEXT
DLA_C_SPEC_BODY_RASTER
DLA_C_SPEC_FOOTNOTE_TEXT
DLA_C_SPEC_ENTRY_TEXT
DLA_C_SPEC_ENTRY_RASTER
DLA_C_GEN_ENTRY_TEXT
DLA_C_GEN_ENTRY_RASTER
DLA_C_GEN_BODY_TEXT
DLA_C_GEN_BODY_RASTER
DLA_C_GEN_FOOTNOTE_TEXT
DLA_C_GEN_COMMON_TEXT
DLA_C_GEN_COMMON_RASTER
DLA_C_GEN_GENERIC_BLOCK
DLA_C_SPEC_GENERIC_BLOCK
DLA_C_SPEC_SPECIFIC_BLOCK
```

No conversion is available for geometric content architecture.

## **dla\_convert\_content**

### **Arguments**

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**type\_of\_conversion**

The conversion to be performed, which should be one of the following:

DLA_C_CHAR_6937_TO_8859	ISO 6937 to ISO 8859-1 character sets
DLA_C_CHAR_8859_TO_6937	ISO 8859-1 to ISO 6937 character sets
DLA_C_RAS_TO_T4_1D	raster to T4 one dimensional encoding
DLA_C_RAS_TO_T4_2D	raster to T4 two dimensional encoding
DLA_C_RAS_TO_T6	raster to T6 encoding
DLA_C_RAS_TO_BIT	raster to bitmap encoding

### **Results**

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### **Errors**

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_INV\_CONVERSION]**

Raster conversion is being attempted on character content or character conversion on raster content.

**[DLA\_E\_INV\_ENCODING]**

This error code corresponds to the ODA API error code OLA\_E\_INV\_ENCODING.

**[DLA\_E\_CANT\_COMP]**

This error code corresponds to the ODA API error code OLA\_E\_CANT\_COMP.

**[DLA\_E\_INV\_CONV]**

This error code corresponds to the ODA API error code OLA\_E\_INV\_CONV.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_create\_document

Creates an instance of a document that conforms to a specified DAP level.

## C Synopsis

```

dia_status      dia_create_document (
    dia_toolkit_handle    toolkit_handle,
    dia_dap_level         dap_level,
    dia_document_options  document_options,
    char                  *store_file,
    dia_length            file_spec_length,
    dia_document_handle   *document_handle_return);

```

## Description

The `dia_create_document` function creates an instance of a document that conforms to a specified DAP level.

The document processing option that is chosen influences the operations that can be performed during document creation.

One entity of the following type is created automatically:

DLA\_C\_PROFILE

Additionally, in mode-1, one entity of each of the following types is also created automatically:

DLA\_C\_SPEC\_LOG\_ROOT

DLA\_C\_GEN\_LOG\_ROOT

DLA\_C\_GEN\_LAY\_ROOT

The toolkit assists the application in creating a document that conforms to the specified DAP level. Errors are returned by subsequent function calls, if the application attempts to specify entities or associated properties not allowed at the specified DAP level. Errors are also returned for attempts to specify nonconformant property values or links.

**dla\_create\_document****Arguments****toolkit\_handle**

The handle of the toolkit being operated on.

**dap\_level**

An enumerated value that indicates the required DAP level.

Dap_level	Meaning
DLA_C_DAP_LEVEL_1	DAP level 1
DLA_C_DAP_LEVEL_2	DAP level 2
DLA_C_DAP_LEVEL_3	DAP level 3

**document\_options**

An enumerated value that allows the application to specify options to control the processing of the document. The values are as follows:

Option	Meaning
DLA_C_NO_OPTIONS	Equivalent to DLA_C_MODE_1
DLA_C_MODE_1	Toolkit creates generic logical structure
DLA_C_AUTO_GEN	Toolkit creates generic logical structure (Equivalent to DLA_C_MODE_1)
DLA_C_MODE_2	Application creates generic logical structure
DLA_C_WITH_EXTERNAL	Application will read generic document from ODIF
DLA_C_FOR_EXTERNAL	Application will supply ODIF identifiers for all entities
DLA_C_FOR_RESOURCE	Application will supply ODIF identifiers for all entities and the resources property

**store\_file**

A string containing the file specification for the temporary files used by the memory management scheme. If this argument is not required by the toolkit implementation, DLA\_C\_NULL\_STRING must be specified.

**file\_spec\_length**

The length of the string containing the file specification for the temporary files used by the memory management scheme.

**document\_handle\_return [write]**

The handle of the new document.

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_INV\_TOOLKIT]

The toolkit handle is not valid.

### [DLA\_E\_DAP\_LEVEL]

The DAP level specified is not supported.

### [DLA\_E\_DAP\_MODE]

The given dap\_level (FOD11) does not permit a generic document.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## **dia\_create\_generic\_basic\_float**

## **dia\_create\_generic\_basic\_float**

Creates a generic basic float entity.

### **C Synopsis**

```
dia_status      dia_create_generic_basic_float(  
    dia_document_handle    document_handle,  
    dia_entity_handle      generic_parent_handle,  
    dia_entity_handle      *generic_basic_float_handle);
```

### **Description**

The `dia_create_generic_basic_float` function creates a `DLA_C_GEN_BASIC_FLOAT` entity, and updates the Generator for subordinates of the parent generic frame entity specified.

The handle of the new generic basic float is returned.

A unique category name is created and the value of the Permitted categories attribute of the ODA constituent corresponding to the generic basic float entity is set to this category name.

This function is not applicable at DAP Level 1.

### **Arguments**

**document\_handle**

The handle of the document being operated on.

**generic\_parent\_handle**

The handle of the parent generic frame entity which is the superior of the new generic basic float.

**generic\_basic\_float\_handle [write]**

The handle of the new basic float entity.

### **Results**

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_MODE]**

The writing mode of this document is not appropriate for this function.

**[DLA\_E\_INV\_OP\_AT\_DAP\_LEVEL]**

The requested operation is not valid at the current DAP level.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_TOO\_MANY\_CATEGORIES]**

The number of categories would have exceeded the limit (99 999 999) with the creation of this frame.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dla\_create\_generic\_body\_frame**

## **dla\_create\_generic\_body\_frame**

Creates a body frame of type DLA\_C\_GEN\_COMP\_BODY\_VAR, DLA\_C\_GEN\_COMP\_BODY\_FIX or DLA\_C\_GEN\_BASIC\_BODY, depending on the value of an argument.

### **C Synopsis**

```
dla_status      dla_create_generic_body_frame(  
    dla_document_handle      document_handle,  
    dla_entity_handle        generic_parent_handle,  
    dla_entity_type          body_entity_type,  
    dla_entity_handle        *generic_body_handle);
```

### **Description**

The `dla_create_generic_body_frame` function creates a body frame within the generic parent page or frame specified.

The value of the argument determines the DAP type of the frame that is created. This is the only function that can be used to create a body frame subordinate to a generic page.

The property Page layout type must be set before any immediate subordinates of a generic page are created or linked, and cannot be changed thereafter.

### **Arguments**

#### **document\_handle**

The handle of the document being operated on.

#### **generic\_parent\_handle**

The handle of the generic page or frame within which the body frame is to be created.

#### **body\_entity\_type**

The type of the entity required. It must be one of the following:

DLA\_C\_GEN\_COMP\_BODY\_VAR

DLA\_C\_GEN\_COMP\_BODY\_FIX

DLA\_C\_GEN\_BASIC\_BODY

#### **generic\_body\_handle [write]**

The handle of the new body frame.



## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_DOC\_READ\_ONLY]

The document is read only.

### [DLA\_E\_INV\_MODE]

The writing mode of this document is not appropriate for this function.

### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

### [DLA\_E\_INV\_ENTITY\_OPER]

The requested operation is not valid for an entity of this type.

### [DLA\_E\_BODY\_EXISTS]

A body frame is already linked to the given page.

### [DLA\_E\_PAG\_LAY\_NOTSPEC]

The Page Layout Type of the generic page has not been specified.

### [DLA\_E\_INV\_OP\_AT\_DAP\_LEVEL]

The requested operation is not valid at the current DAP level.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_create\_generic\_comm\_entity****dia\_create\_generic\_comm\_entity**

Creates an entity of type CommonText, PageNumber, CommonRaster, CommonGeometric, CommonReference, CommonNumber, CurrentInstance or TableNumber.

**C Synopsis**

```
dia_status      dia_create_generic_comm_entity(
    dia_document_handle    document_handle,
    dia_entity_handle      frame_entity_handle,
    dia_entity_type        common_entity_type,
    dia_entity_handle      *entity_handle_return);
```

**Description**

The function `create_generic_comm_entity` creates one of the common entities corresponding to sourced content, and updates the specified frame to reference the new generic common entity. The frame is specified by the entity whose handle is `frame_entity_handle`. The type of the frame entity must be one of the following:

```
DLA_C_GEN_BASIC_HEADER
DLA_C_GEN_BASIC_FOOTER
DLA_C_GEN_SRCD_CONT_VAR
DLA_C_GEN_SRCD_CONT_FIXED
DLA_C_GEN_TABLE_LABEL_CONTENT
```

The type of the requested new entity is specified in the argument `common_entity_type`.

The function returns the handle of the new entity in the argument `entity_handle`, so that presentation and layout directive properties and, for example, in the case of page or table number entities, the format of the number can be set on that entity.

The first call of this function for a frame causes an ODA constituent corresponding to the constraint `CommonContent` to be created and the logical source attribute of the frame is set to point to it.

Successive calls to this function for a frame add the resulting underlying ODA constituent to the Generator for subordinates of the ODA constituent corresponding to the constraint `CommonContent`.

Sourced content can be placed in entities of types `CommonText`, `CommonRaster` or `CommonGeometric`, using the normal `dia_add_content()` function.

This function can be used to create entities of types `CommonText`, `CommonRaster` or `CommonGeometric` in a resource document. In this case there can be no frames and the value of the argument `frame_entity_handle` must be `DLA_C_NULL_HANDLE`. The actions relating to the frame indicated by this argument, (described earlier in this section) do not take place.

## Arguments

**document\_handle**

The handle of the document being operated on.

**frame\_entity\_handle**

The handle of the frame entity that is to refer to the sourced content to be contained in the common\_entity. In mode DLA\_C\_FOR\_RESOURCE the value of this argument must be DLA\_C\_NULL\_HANDLE.

**common\_entity\_type**

The type of the entity required. It must be one of the following:

DLA\_C\_GEN\_PAGE\_NUMBER  
DLA\_C\_GEN\_COMMON\_TEXT  
DLA\_C\_GEN\_COMMON\_GEO  
DLA\_C\_GEN\_COMMON\_RASTER  
DLA\_C\_GEN\_COMMON\_REFERENCE  
DLA\_C\_GEN\_COMMON\_NUMBER  
DLA\_C\_GEN\_CURRENT\_INSTANCE  
DLA\_C\_GEN\_TABLE\_NUMBER

**entity\_handle\_return [write]**

The handle of the new entity created by this function.

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_MODE]**

The writing mode of this document is not appropriate for this function.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_INV\_OP\_AT\_DAP\_LEVEL]**

The common entity type is not permitted at this DAP Level.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dia\_create\_generic\_fnote\_area**

## **dia\_create\_generic\_fnote\_area**

Creates a generic footnote area entity.

### **C Synopsis**

```
dia_status      dia_create_generic_fnote_area(  
    dia_document_handle      document_handle,  
    dia_entity_handle        generic_composite_frame_handle,  
    dia_entity_handle        *generic_footnote_area_handle);
```

### **Description**

The `dia_create_generic_fnote_area` function creates a `DLA_C_GEN_FOOTNOTE_AREA` entity, and updates the Generator for subordinates of the frame entity specified.

The type of the `generic_composite_frame_handle` can be one of:

`DLA_C_GEN_COMP_BODY_VAR`  
`DLA_C_GEN_COMP_FIXTURE_VAR`  
`DLA_C_GEN_COMP_FIXTURE_FIXED`  
`DLA_C_GEN_COMP_COL_FIXED`  
`DLA_C_GEN_COMP_COL_VARIABLE`

The handle of the new generic footnote area is returned.

### **Arguments**

**document\_handle**

The handle of the document being operated on.

**generic\_composite\_frame\_handle**

The handle of the generic composite frame entity which identifies the superior of the new footnote area.

**generic\_footnote\_area\_handle [write]**

The handle of the new footnote area entity.

### **Results**

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_FNOTE\_AREA\_EXISTS]**

There can be only one footnote area within a frame and it is already present. (Restriction additional to level 3 DAP constraint.)

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_MODE]**

The writing mode of this document is not appropriate for this function.

**[DLA\_E\_INV\_OP\_AT\_DAP\_LEVEL]**

The requested operation is not valid at the current DAP level.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## **dia\_create\_generic\_frame**

## **dia\_create\_generic\_frame**

Creates a generic frame entity of one of a number of types, depending on the value of an argument.

### **C Synopsis**

```
dia_status      dia_create_generic_frame(  
    dia_document_handle    document_handle,  
    dia_entity_type        frame_entity_type,  
    dia_entity_handle      generic_parent_handle,  
    dia_entity_handle      *generic_frame_handle);
```

### **Description**

The `dia_create_generic_frame` function creates a new generic frame as a potential subordinate of the generic frame specified. To create a generic body frame as a subordinate to a page, the function `dia_create_generic_body_frame` must be used. Note that there are a number of frame types that are not included and for which special functions must be used.

The value of the argument determines the DAP type of the frame that is created.

### **Arguments**

#### **document\_handle**

The handle of the document being operated on.

#### **frame\_entity\_type**

The type of the entity required. It must be one of the following:

DLA\_C\_GEN\_COMP\_BODY\_VAR  
DLA\_C\_GEN\_SRCD\_CONT\_FIXED  
DLA\_C\_GEN\_BASIC\_FLOAT  
DLA\_C\_GEN\_COMP\_FLOAT  
DLA\_C\_GEN\_BASIC\_COLUMN  
DLA\_C\_GEN\_COMP\_FIXTURE\_VAR  
DLA\_C\_GEN\_COMP\_FIXTURE\_FIXED  
DLA\_C\_GEN\_BASIC\_FIXTURE  
DLA\_C\_GEN\_COMP\_COL\_FIXED  
DLA\_C\_GEN\_COMP\_COL\_VARIABLE  
DLA\_C\_GEN\_COMP\_ARTWORK  
DLA\_C\_GEN\_FORM\_AREA  
DLA\_C\_GEN\_TABLE\_HEADER

## dia\_create\_generic\_frame

DLA\_C\_GEN\_ENTRY\_GROUP\_AREA  
 DLA\_C\_GEN\_TABLE\_AREA  
 DLA\_C\_GEN\_TABLE\_LABEL  
 DLA\_C\_GEN\_COMP\_TABLE\_LABEL  
 DLA\_C\_GEN\_LABEL\_COMPONENT  
 DLA\_C\_GEN\_ROW\_AREA  
 DLA\_C\_GEN\_CELL  
 DLA\_C\_GEN\_SUBROW\_GROUP  
 DLA\_C\_GEN\_SUBROW  
 DLA\_C\_GEN\_TABLE\_LABEL\_CONTENT  
 DLA\_C\_GEN\_FORM\_ENTRY\_AREA

**generic\_parent\_handle**

The handle of the generic frame to which the new generic frame is to be a potential subordinate.

**generic\_frame\_handle [write]**

The handle of the new generic frame.

## Results

**Status**

Normally success, indicating that the operation was successful, otherwise one of the following errors is returned.

## Errors

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_MODE]**

The writing mode of this document is not appropriate for this function.

**[DLA\_E\_INV\_PARENT\_ENTITY]**

The parent entity is not in the document.

**[DLA\_E\_NOT\_PERM\_AT\_DAP\_LEVEL]**

The link requested is not valid at the DAP level of the document.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_GFS\_NOT\_PERM]**

The pair of entities of these types cannot be linked in this manner because the resulting Generator for subordinates is not permitted for the current DAP level.

**[DLA\_E\_SNAKING\_REP]**

A CompositeColumn/Variable frame cannot be added to a Snaking Column frame where the earlier call to `dia_create_generic_snaking` specified `DLA_C_SNAKING_REP` as the value of `snaking_type` and a subordinate generic frame is already established.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

`dia_create_generic_frame`

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_create\_generic\_frame\_seq**

Creates a generic frame entity of one of a number of types, depending on the value of an argument.

This function permits the creation of an entity at a particular position in the generic layout structure.

**C Synopsis**

```
dia_status      dia_create_generic_frame_seq(
    dia_document_handle    document_handle,
    dia_entity_type        frame_entity_type,
    dia_entity_handle      generic_parent_handle,
    dia_integer            index,
    dia_entity_handle      *generic_frame_handle);
```

**Description**

The `dia_create_generic_frame_seq` function creates a new generic frame as a potential subordinate of the generic frame specified. To create a generic body frame as a subordinate to a page, the function `dia_create_generic_body_frame` must be used. Note that there are a number of frame types that are not included and for which special functions must be used.

The value of the argument determines the DAP type of the frame that is created.

This function should be used rather than `dia_create_generic_frame` when the new entity should not be appended to the sequence of entities in the Generator for subordinates. This may apply because that position is not conformant with DAP constraints or because it would not produce the desired effect.

The `generic_parent_handle` parameter specifies the superior entity to which the new frame entity is to be linked. The type of superior must be one for which the new entity can be added as a subordinate. That is, the Generator for subordinates of the superior class must be capable of being expanded, within DAP constraints, to reference it.

The linking is done by index and not by link entity, because a link entity is generally not unique in a Generator for subordinates (unlike the case for a specific structure).

The index argument must have a valid value for the required position (starting at one) of the entity. This means that the new entry precedes an existing one at that position.

## **dla\_create\_generic\_frame\_seq**

### **Arguments**

**document\_handle**

The handle of the document being operated on.

**frame\_entity\_type**

The type of the entity required. It must be one of the following:

DLA\_C\_GEN\_COMP\_BODY\_VAR  
DLA\_C\_GEN\_SRCD\_CONT\_FIXED  
DLA\_C\_GEN\_BASIC\_FLOAT  
DLA\_C\_GEN\_COMP\_FLOAT  
DLA\_C\_GEN\_BASIC\_COLUMN  
DLA\_C\_GEN\_COMP\_FIXTURE\_VAR  
DLA\_C\_GEN\_COMP\_FIXTURE\_FIXED  
DLA\_C\_GEN\_BASIC\_FIXTURE  
DLA\_C\_GEN\_COMP\_COL\_FIXED  
DLA\_C\_GEN\_COMP\_COL\_VARIABLE  
DLA\_C\_GEN\_COMP\_ARTWORK  
DLA\_C\_GEN\_FORM\_AREA  
DLA\_C\_GEN\_TABLE\_HEADER  
DLA\_C\_GEN\_ENTRY\_GROUP\_AREA  
DLA\_C\_GEN\_TABLE\_AREA  
DLA\_C\_GEN\_TABLE\_LABEL  
DLA\_C\_GEN\_COMP\_TABLE\_LABEL  
DLA\_C\_GEN\_LABEL\_COMPONENT  
DLA\_C\_GEN\_ROW\_AREA  
DLA\_C\_GEN\_CELL  
DLA\_C\_GEN\_SUBROW\_GROUP  
DLA\_C\_GEN\_SUBROW  
DLA\_C\_GEN\_TABLE\_LABEL\_CONTENT  
DLA\_C\_GEN\_FORM\_ENTRY\_AREA

**generic\_parent\_handle**

The handle of the generic frame to which the new generic frame is to be a potential subordinate.

**index**

The index of the entry at which the value is to be stored. The first entry in a sequence is at index 1.

**generic\_frame\_handle [write]**

The handle of the new generic frame.

### **Results**

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_MODE]**

The writing mode of this document is not appropriate for this function.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_INV\_PARENT\_ENTITY]**

The parent entity is not in the document.

**[DLA\_E\_NOT\_PERM\_AT\_DAP\_LEVEL]**

The link requested is not valid at the DAP level of the document.

**[DLA\_E\_GFS\_NOT\_PERM]**

The pair of entities of these types cannot be linked in this manner because the resulting Generator for subordinates is not permitted for the current DAP level.

**[DLA\_E\_SNAKING\_REP]**

A CompositeColumn/Variable frame cannot be added to a Snaking Column frame where the earlier call to `dla_create_generic_snaking` specified `DLA_C_SNAKING_REP` as the value of `snaking_type` and a subordinate generic frame is already established.

**[DLA\_E\_INDEX\_OUT\_OF\_RNG]**

The index is out of range for valid indexes for the entries in the Generator for subordinates.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_create\_generic\_head\_footer****dia\_create\_generic\_head\_footer**

Creates a cluster of generic layout entities associated with a generic header or footer.

**C Synopsis**

```
dia_status      dia_create_generic_head_footer(
    dia_document_handle    document_handle,
    dia_entity_handle      generic_page_handle,
    dia_header_footer_type header_footer_type,
    dia_integer            num_content_entities,
    dia_entity_handle      *generic_head_foot_handle);
```

**Description**

The `dia_create_generic_head_footer` function creates a cluster of generic layout entities associated with a generic header or footer created on the specified generic page. The Generator for subordinates of the specified generic page entity is updated.

The value of the `header_footer_type` argument specifies whether a header or footer is required, and for a composite header or footer specifies whether the subordinate entities are to be of type `DLA_C_GEN_SRC_D_CONT_VAR` or `DLA_C_GEN_SRC_D_CONT_FIXED`.

The `num_content_entities` argument specifies the required number of `DLA_C_GEN_SRC_D_CONT_VAR` or `DLA_C_GEN_SRC_D_CONT_FIXED` entities for a composite header or footer.

The property Page layout type must be set before any immediate subordinates of a generic page are created or linked, and cannot be changed thereafter.

The property Header footer frames (`DLA_C_HED_FOT`) consists of the handles of the SourcedContentFixed or SourcedContentVariable frames created by this function. These handles can be retrieved to set properties of entities within the cluster.

**Arguments**

**document\_handle**  
The handle of the document being operated on.

**generic\_page\_handle**  
The handle of the generic page being operated on.

## dia\_create\_generic\_head\_footer

**header\_footer\_type**

The type of header or footer. An enumerated value, that is one of the following:

Header_footer_type	Meaning
DLA_C_COMP_HEAD_FIXED	A composite header with a sequence of subordinate entities of type DLA_C_GEN_SRCD_CONT_FIXED
DLA_C_COMP_HEAD_VAR	A composite header with a sequence of subordinate entities of type DLA_C_GEN_SRCD_CONT_VAR
DLA_C_COMP_FOOT_FIXED	A composite footer with a sequence of subordinate entities of type DLA_C_GEN_SRCD_CONT_FIXED
DLA_C_COMP_FOOT_VAR	A composite footer with a sequence of subordinate entities of type DLA_C_GEN_SRCD_CONT_VAR
DLA_C_BASIC_HEAD	A basic header
DLA_C_BASIC_FOOT	A basic footer

**num\_content\_entities**

An integer value specifying the number of entities of type DLA\_C\_GEN\_SRCD\_CONT\_FIXED or DLA\_C\_GEN\_SRCD\_CONT\_VAR required in a composite header or footer.

**generic\_head\_foot\_handle [write]**

The handle of the new generic header or footer.

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_MODE]**

The writing mode of this document is not appropriate for this function.

**[DLA\_E\_INV\_TYPE\_AT\_DAP\_LEVEL]**

The requested header or footer type is not valid at the current DAP level.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.



## `dia_create_generic_head_footer`

### **[DLA\_E\_HEADER\_EXISTS]**

A header frame is already linked to the given page.

### **[DLA\_E\_FOOTER\_EXISTS]**

A footer frame is already linked to the given page.

### **[DLA\_E\_PAG\_LAY\_NOTSPEC]**

The Page Layout Type of the generic page has not been specified.

### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_create\_generic\_pageset**

Creates a cluster of generic layout entities associated with the generic pageset.

**C Synopsis**

```
dia_status      dia_create_generic_pageset (
    dia_document_handle    document_handle,
    dia_pageset_type       pageset_type,
    dia_entity_handle      *generic_pageset_handle);
```

**Description**

The `dia_create_generic_pageset` function creates a cluster of generic layout entities associated with the generic pageset. The value of the `pageset_type` argument determines the composition of the cluster.

In all cases, a `DLA_C_GEN_PAGESET` entity is created and the Generator for subordinates of the `DLA_C_GEN_LAY_ROOT` entity is updated. If there is no `DLA_C_GEN_LAY_ROOT` at the time of the call, then it is created.

The following subordinates of `DLA_C_GEN_PAGESET` are created, depending on the value of the `pageset_type` argument:

```
DLA_C_GEN_PAGE
DLA_C_GEN_RECTO_PAGE
DLA_C_GEN_VERSO_PAGE
```

The `pageset_type` argument also determines the value of the Generator for subordinates of the created pageset.

The property Generic page (`DLA_C_GEN_PGE`) consists of the handles of the generic page entities created by this function. These handles can be retrieved to set properties of entities within the cluster.

**Arguments**

**document\_handle**

The handle of the document being operated on.

## dlc\_create\_generic\_pageset

### pageset\_type

A description of the set of page entity subordinates required for the pageset entity, expressed as a selection from the following enumeration:

Pageset_type	Meaning
DLA_C_TITLE_PAGE	Create a DLA_C_GEN_PAGE entity representing a title page.
DLA_C_SINGLE_PAGE	Create a DLA_C_GEN_PAGE entity representing a repetition of single pages.
DLA_C_RECTO_VERSO	Create DLA_C_GEN_RECTO_PAGE and DLA_C_GEN_VERSO_PAGE entities representing a repetition of recto and verso pages.
DLA_C_TITLE_SINGLE_PAGE	Create two DLA_C_GEN_PAGE entities representing a title page and a repetition of single pages.
DLA_C_TITLE_VERSO_RECTO	Create DLA_C_GEN_PAGE, DLA_C_GEN_VERSO_PAGE and DLA_C_GEN_RECTO_PAGE entities representing a title page and a repetition of verso and recto pages.

### generic\_pageset\_handle [write]

The handle of the new generic pageset.

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_DOC\_READ\_ONLY]

The document is read only.

### [DLA\_E\_INV\_MODE]

The writing mode of this document is not appropriate for this function.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## dla\_create\_generic\_snaking

Creates a cluster of generic layout entities associated with a generic snaking column.

## C Synopsis

```
dla_status      dla_create_generic_snaking(
    dla_document_handle    document_handle,
    dla_entity_handle      generic_composite_body_handle,
    dla_snaking_type        snaking_type,
    dla_integer            num_col_var,
    dla_entity_handle      *generic_snaking_handle);
```

## Description

The `dla_create_generic_snaking` function creates a cluster of generic layout entities associated with a generic snaking column in the specified generic composite body frame.

A `DLA_C_GEN_SNAKING_COLUMN` entity is created and the Generator for subordinates of the `DLA_C_GEN_COMP_BODY_VAR` entity is updated.

The value of the `snaking_type` argument determines the type of the Generator for subordinates created on the new `DLA_C_GEN_SNAKING_COLUMN` entity.

The `num_col_var` argument specifies the required number of `DLA_C_GEN_COL_VARIABLE` entities if the snaking type is `DLA_C_SNAKING_SEQ`. It must be 1 in the case of FOD26 documents if the snaking type is `DLA_C_SNAKING_REP`. (See below for the case of FOD36 documents.)

A unique category name is created for the set of `DLA_C_GEN_COL_VARIABLE` entities and the value of the Permitted categories attribute of each member is set to this category name.

In FOD36 documents a snaking column frame can have Composite Column Variable frames as subordinates as well as, or instead of, lowest level Column Variable frames. Depending on the value of the argument `snaking_type`, either a single or several such generic frames can be linked or created as subordinates to a snaking columns frame. It is permitted (and required) to set `num_col_var` to zero if the subordinates are to consist only of Composite Column Variable frames.

If multiple generic Composite Column Variable frames are used, it is the responsibility of the application to ensure that all corresponding lowest level frames share their category name by using the function `dla_make_frame_equiv` to modify the value of the Permitted categories attribute of some of the frames as necessary.

The property Snaking column (`DLA_C_SNA_COL`) consists of the handles of the ColumnVariable frames created by this function. These handles can be retrieved to set properties of entities within the cluster.

## dla\_create\_generic\_snaking

### Arguments

**document\_handle**

The handle of the document being operated on.

**generic\_composite\_body\_handle**

The handle of the entity of type generic variable composite body within which the snaking column is required.

**snaking\_type**

The type of snaking column. An enumerated value that is one of the following:

Snaking_type	Meaning
DLA_C_SNAKING_REP	A repetition of a single (composite) column variable entity is required.
DLA_C_SNAKING_SEQ	A sequence of different (composite) column variable entities is required.

**num\_col\_var**

The number of DLA\_C\_GEN\_COL\_VARIABLE entities required. It must have a value of 0 or 1 if snaking\_type is DLA\_C\_SNAKING\_REP.

**generic\_snaking\_handle [write]**

The handle of the new generic snaking column.

### Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned.

### Errors

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_MODE]**

The writing mode of this document is not appropriate for this function.

**[DLA\_E\_INV\_OP\_AT\_DAP\_LEVEL]**

The requested operation is not valid at the current DAP level.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_INV\_COL\_COUNT]**

The number of columns specified is not allowed for the given snaking\_type and DAP level.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_TOO\_MANY\_CATEGORIES]

The number of categories would have exceeded the limit (99 999 999) with the creation of this group of frames.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_create\_generic\_synchronized****dia\_create\_generic\_synchronized**

Creates the cluster of generic layout entities associated with a generic synchronized column.

**C Synopsis**

```
dia_status      dia_create_generic_synchronized(
    dia_document_handle    document_handle,
    dia_entity_handle      generic_composite_body_handle,
    dia_integer            num_col_fix,
    dia_entity_handle      *generic_synch_handle);
```

**Description**

The `dia_create_generic_synchronized` function creates the cluster of generic layout entities associated with a generic synchronized column created in the specified generic variable composite body frame.

A `DLA_C_GEN_SYNC_COLUMN` entity is created and the Generator for subordinates of the `DLA_C_GEN_COMP_BODY_VAR` entity is updated.

The `num_col_fix` argument specifies the number of `DLA_C_GEN_COL_FIXED` entities required.

A unique category name is created for each `DLA_C_GEN_COL_FIXED` entity and the value of the Permitted categories attribute of each is set to this name.

In FOD36 documents a synchronized columns frame can have Composite Column Fixed frames as subordinates as well as lowest level Column Fixed frames. Either a single or several such generic frames can be linked or created as subordinates to a synchronized columns frame. It is permitted (and required) to set `num_col_fix` to zero if the subordinates are to consist only of Composite Column Fixed frames.

The property Synchronized columns (`DLA_C_SYN_COL`) consists of the handles of the ColumnFixed frames created by this function. These handles can be retrieved to set properties of entities within the cluster.

**Arguments**

**document\_handle**

The handle of the document being operated on.

**generic\_composite\_body\_handle**

The handle of the generic variable composite body frame in which the synchronized columns are required.

**num\_col\_fix**

The number of `DLA_C_GEN_COL_FIXED` entities required.

**generic\_synch\_handle [write]**

The handle of the new generic synchronized column.

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_DOC\_READ\_ONLY]

The document is read only.

### [DLA\_E\_INV\_MODE]

The writing mode of this document is not appropriate for this function.

### [DLA\_E\_INV\_OP\_AT\_DAP\_LEVEL]

The requested operation is not valid at the current DAP level.

### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

### [DLA\_E\_INV\_ENTITY\_OPER]

The requested operation is not valid for an entity of this type.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_TOO\_MANY\_CATEGORIES]

The number of categories would have exceeded the limit (99 999 999) with the creation of this group of frames.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## **dia\_create\_gen\_log\_entity**

## **dia\_create\_gen\_log\_entity**

Creates a generic logical entity and links it into the generic logical structure. This function does not create common entities.

### **C Synopsis**

```
dia_status      dia_create_gen_log_entity(  
    dia_document_handle    document_handle,  
    dia_entity_type        entity_type,  
    dia_entity_handle       parent_entity,  
    dia_entity_handle       *entity_handle_return)
```

### **Description**

The `dia_create_gen_log_entity` function creates a generic entity of the type specified. The list of entity types does not include common entities or mandatory entities that are created as part of a cluster.

This function cannot be used if the writing mode of the document is mode-1 or `DLA_C_WITH_EXTERNAL`.

The `parent_entity` parameter specifies the superior entity to which the new logical entity is to be linked. The type of superior must be one for which the new entity can be added as a subordinate; that is, the Generator for subordinates of the superior class must be capable of being expanded, within DAP constraints, to reference it.

When the entity type is `DLA_C_GEN_LOG_ROOT`, the value of the `parent_entity` parameter must be `DLA_C_NULL_HANDLE`.

When the generic entity type is `NumberedSegment`, `Footnote` or `Reference`, a cluster is created.

### **Arguments**

**document\_handle**

The handle of the document being operated on.



## dla\_create\_gen\_log\_entity

**entity\_type**

Indicates the type of DAP entity required. The value specified must be one of the following:

Entity type	Meaning
DLA_C_GEN_LOG_ROOT	DocumentLogicalRoot
DLA_C_GEN_PASSAGE	Passage
DLA_C_GEN_NUM_SEG	NumberedSegment
DLA_C_GEN_NUMBER	Number
DLA_C_GEN_TITLE	Title
DLA_C_GEN_CAPTION	Caption
DLA_C_GEN_PARAGRAPH	Paragraph
DLA_C_GEN_PHRASE	Phrase
DLA_C_GEN_FOOTNOTE	Footnote
DLA_C_GEN_FOOTNOTE_TEXT	FootnoteText
DLA_C_GEN_FIGURE	Figure
DLA_C_GEN_BODY_TEXT	BodyText
DLA_C_GEN_REFERENCE	Reference
DLA_C_GEN_BODY_RASTER	BodyRaster
DLA_C_GEN_BODY_GEOM	BodyGeometric
DLA_C_GEN_DESCRIPTION	Description
DLA_C_GEN_ARTWORK	Artwork
DLA_C_GEN_NUMBERED_LIST	NumberedList
DLA_C_GEN_UNNUMBERED_LIST	UnNumberedList
DLA_C_GEN_DEFINITION_LIST	DefinitionList
DLA_C_GEN_LIST_ITEM	ListItem
DLA_C_GEN_LIST_TERM	ListTerm
DLA_C_GEN_TABLE	Table
DLA_C_GEN_ROW	Row
DLA_C_GEN_TABLE_COMPONENT	TableComponent
DLA_C_GEN_ROW_COMPONENT	RowComponent
DLA_C_GEN_FORM	Form
DLA_C_GEN_ENTRY_ELEMENT	EntryElement
DLA_C_GEN_ENTRY_GROUP	EntryGroup
DLA_C_GEN_ENTRY_TEXT	EntryText
DLA_C_GEN_ENTRY_RASTER	EntryRaster
DLA_C_GEN_ENTRY_GEOMETRIC	EntryGeometric

**parent\_entity**

The handle of the entity that specifies the position for the new entity in the generic logical structure. In mode DLA\_C\_FOR\_RESOURCE the value of this argument must be DLA\_C\_NULL\_HANDLE.

## **dla\_create\_gen\_log\_entity**

**entity\_handle\_return** [write]  
The handle of the new generic logical entity.

### **Results**

**Status**  
Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### **Errors**

**[DLA\_E\_NOT\_PERM\_AT\_DAP\_LEVEL]**  
The link requested is not valid at the DAP level of the document.

**[DLA\_E\_DOC\_READ\_ONLY]**  
The document is read only.

**[DLA\_E\_INV\_MODE]**  
The writing mode of this document is not appropriate for this function.

**[DLA\_E\_DUP\_ROOT]**  
A generic logical root already exists.

**[DLA\_E\_INV\_PARENT\_ENTITY]**  
The parent entity is not in the document.

**[DLA\_E\_GFS\_NOT\_PERM]**  
The pair of entities of these types cannot be linked in this manner because the resulting Generator for subordinates is not permitted for the current DAP level.

**[DLA\_E\_CLUMP\_FINISHED]**  
The operation is not permitted because it would require that the Generator for subordinates of a finished clump be updated.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**  
Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**  
An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_create\_gen\_log\_entity\_seq

Creates a generic logical entity and links it into the generic logical structure. This function does not create common entities.

This function permits the creation of an entity at a particular position in the generic logical structure.

## C Synopsis

```
dia_status      dia_create_gen_log_entity_seq(
    dia_document_handle    document_handle,
    dia_entity_type        entity_type,
    dia_entity_handle      parent_entity,
    dia_integer            index,
    dia_entity_handle      *entity_handle_return);
```

## Description

The `dia_create_gen_log_entity_seq` function creates a generic entity of the type specified. The list of entity types does not include common entities or mandatory entities that are created as part of a cluster.

This function cannot be used if the writing mode of the document is mode-1, `DLA_C_WITH_EXTERNAL` or `DLA_C_FOR_RESOURCE`.

This function should be used rather than `dia_create_gen_log_entity` when the entity should not be appended to the sequence of entities in the Generator for subordinates. This may apply because that position is not conformant with DAP constraints or because it would not produce the desired effect.

The `parent_entity` parameter specifies the superior entity to which the new logical entity is to be linked. The type of superior must be one for which the new entity can be added as a subordinate; that is, the Generator for subordinates of the superior class must be capable of being expanded, within DAP constraints, to reference it.

The linking is done by index and not by link entity because a link entity is generally not unique in a Generator for subordinates (unlike the case for a specific structure).

The `index` argument must have a valid value for the required position (starting at one) of the entity. Thus the new entry precedes an existing one at that position.

When the generic entity type is `NumberedSegment`, `Footnote` or `Reference`, a cluster is created.

## Arguments

`document_handle`

The handle of the document being operated on.

## dia\_create\_gen\_log\_entity\_seq

**entity\_type**

Indicates the type of DAP entity required. The value specified must be one of the following:

Entity type	Meaning
DLA_C_GEN_PASSAGE	Passage
DLA_C_GEN_NUM_SEG	NumberedSegment
DLA_C_GEN_NUMBER	Number
DLA_C_GEN_TITLE	Title
DLA_C_GEN_CAPTION	Caption
DLA_C_GEN_PARAGRAPH	Paragraph
DLA_C_GEN_PHRASE	Phrase
DLA_C_GEN_FOOTNOTE	Footnote
DLA_C_GEN_FOOTNOTE_TEXT	FootnoteText
DLA_C_GEN_FIGURE	Figure
DLA_C_GEN_BODY_TEXT	BodyText
DLA_C_GEN_REFERENCE	Reference
DLA_C_GEN_BODY_RASTER	BodyRaster
DLA_C_GEN_BODY_GECM	BodyGeometric
DLA_C_GEN_DESCRIPTION	Description
DLA_C_GEN_ARTWORK	Artwork
DLA_C_GEN_NUMBERED_LIST	NumberedList
DLA_C_GEN_UNNUMBERED_LIST	UnNumberedList
DLA_C_GEN_DEFINITION_LIST	DefinitionList
DLA_C_GEN_LIST_ITEM	ListItem
DLA_C_GEN_LIST_TERM	ListTerm
DLA_C_GEN_TABLE	Table
DLA_C_GEN_ROW	Row
DLA_C_GEN_TABLE_COMPONENT	TableComponent
DLA_C_GEN_ROW_COMPONENT	RowComponent
DLA_C_GEN_FORM	Form
DLA_C_GEN_ENTRY_ELEMENT	EntryElement
DLA_C_GEN_ENTRY_GROUP	EntryGroup
DLA_C_GEN_ENTRY_TEXT	EntryText
DLA_C_GEN_ENTRY_RASTER	EntryRaster
DLA_C_GEN_ENTRY_GEOMETRIC	EntryGeometric

**parent\_entity**

The handle of the entity that specifies the position for the new entity in the generic logical structure.

## **dla\_create\_gen\_log\_entity\_seq**

### **index**

The index of the entry at which the value is to be stored. The first entry in a sequence is at index 1.

### **entity\_handle\_return [write]**

The handle of the new generic logical entity.

## **Results**

### **Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## **Errors**

### **[DLA\_E\_NOT\_PERM\_AT\_DAP\_LEVEL]**

The link requested is not valid at the DAP level of the document.

### **[DLA\_E\_INV\_PARENT\_ENTITY]**

The parent entity is not in the document.

### **[DLA\_E\_INV\_ENTITY\_OPER]**

The generic entity\_type cannot be created with this function.

### **[DLA\_E\_GFS\_NOT\_PERM]**

The pair of entities of these types cannot be linked in this manner because the resulting Generator for subordinates is not permitted for the current DAP level.

### **[DLA\_E\_INDEX\_OUT\_OF\_RNG]**

The index is out of range for valid indexes for the entries in the Generator for subordinates.

### **[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

### **[DLA\_E\_INV\_MODE]**

The writing mode of this document is not appropriate for this function.

### **[DLA\_E\_CLUMP\_FINISHED]**

The operation is not permitted because it would require that the Generator for subordinates of a finished clump be updated.

### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dla\_create\_specific\_entity****dla\_create\_specific\_entity**

Creates a specific logical entity and links it into the specific logical structure.

**C Synopsis**

```
dla_status      dla_create_specific_entity(
    dla_document_handle    document_handle,
    dla_entity_type        entity_type,
    dla_entity_handle      link_entity,
    dla_link_option        link_option,
    dla_entity_handle      *entity_handle_return);
```

**Description**

The `dla_create_specific_entity` function creates a specific entity of the type specified.

The `link_entity` and `link_option` parameters together specify the superior entity to which the new logical entity is to be linked and its position in the sequence of subordinates of that entity. The superior must be one to which the new entity can be added as a subordinate. That is, it must be within the DAP constraint for subordinates, and the Generator for subordinates of the ODA class of the superior must either reference the ODA class of the new entity, or, in mode-1 only, must be capable of being expanded to reference it.

When the specific entity type is `NumberedSegment`, `Footnote` or `Reference`, a cluster is created.

**Arguments**

**document\_handle**

The handle of the document being operated on.



**entity\_type**

Indicates the type of DAP entity required. The value specified must be one of the following:

Entity type	Meaning
DLA_C_SPEC_PASSAGE	Specific Passage
DLA_C_SPEC_NUM_SEG	Specific NumberedSegment
DLA_C_SPEC_NUMBER	Specific Number
DLA_C_SPEC_TITLE	Specific Title
DLA_C_SPEC_CAPTION	Specific Caption
DLA_C_SPEC_PARAGRAPH	Specific Paragraph
DLA_C_SPEC_PHRASE	Specific Phrase
DLA_C_SPEC_FOOTNOTE	Specific Footnote
DLA_C_SPEC_FOOTNOTE_TEXT	Specific FootnoteText
DLA_C_SPEC_FIGURE	Specific Figure
DLA_C_SPEC_BODY_TEXT	Specific BodyText
DLA_C_SPEC_REFERENCE	Specific Reference
DLA_C_SPEC_BODY_RASTER	Specific BodyRaster
DLA_C_SPEC_BODY_GEOM	Specific BodyGeometric
DLA_C_SPEC_DESCRIPTION	Specific Description
DLA_C_SPEC_ARTWORK	Specific Artwork
DLA_C_SPEC_NUMBERED_LIST	Specific NumberedList
DLA_C_SPEC_UNNUMBERED_LIST	Specific UnNumberedList
DLA_C_SPEC_DEFINITION_LIST	Specific DefinitionList
DLA_C_SPEC_LIST_ITEM	Specific ListItem
DLA_C_SPEC_LIST_TERM	Specific ListTerm
DLA_C_SPEC_TABLE	Specific Table
DLA_C_SPEC_ROW	Specific Row
DLA_C_SPEC_TABLE_COMPONENT	Specific TableComponent
DLA_C_SPEC_ROW_COMPONENT	Specific RowComponent
DLA_C_SPEC_FORM	Specific Form
DLA_C_SPEC_ENTRY_ELEMENT	Specific EntryElement
DLA_C_SPEC_ENTRY_GROUP	Specific EntryGroup
DLA_C_SPEC_ENTRY_TEXT	Specific EntryText
DLA_C_SPEC_ENTRY_RASTER	Specific EntryRaster
DLA_C_SPEC_ENTRY_GEOMETRIC	Specific EntryGeometric

**link\_entity**

The handle of the entity which, in combination with the argument link\_option, specifies the position for the new entity in the specific logical structure.

## dia\_create\_specific\_entity

### link\_option

One of the values in the following table which, in combination with the argument `link_entity`, specifies the position for the new entity in the specific logical structure.

Option	Meaning
DLA_C_LINK_BEFORE	The new entity will be linked to the parent of <code>link_entity</code> and will precede it.
DLA_C_LINK_AFTER	The new entity will be linked to the parent of <code>link_entity</code> and will succeed it.
DLA_C_LINK_AS_FIRST	The new entity will be linked to the <code>link_entity</code> , becoming its first subordinate.
DLA_C_LINK_AS_LAST	The new entity will be linked to the <code>link_entity</code> , becoming its last subordinate.

### entity\_handle\_return [write]

The handle of the new specific logical entity.

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_ILLEGAL\_SUBORDINATE]

The link entity type prohibits this operation.

### [DLA\_E\_ILLEGAL\_POSITION]

The position requested for the new entity is not allowed. Applies especially to `NumberedSegment` vis a vis `Paragraph`, `BodyText`, `BodyRaster` or `BodyGeometric`.

### [DLA\_E\_NOT\_PERM\_AT\_DAP\_LEVEL]

The link requested is not valid at the DAP level of the document.

### [DLA\_E\_LINK\_INV\_ENTITY]

The link entity does not exist.

### [DLA\_E\_DOC\_READ\_ONLY]

The document is read only.

### [DLA\_E\_INV\_MODE]

The writing mode of this document is not appropriate for this function.

### [DLA\_E\_CLUMP\_FINISHED]

The operation is not permitted because it would require that the Generator for subordinates of a finished clump be updated.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

`dia_create_specific_entity`

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dia\_delete\_document**

## **dia\_delete\_document**

Deletes a DAP API document.

### **C Synopsis**

```
dia_status      dia_delete_document(  
    dia_document_handle    document_handle);
```

### **Description**

The `dia_delete_document` function deletes a DAP API document. The document handle is then no longer valid.

All entity and structure handles belonging to a deleted document are also invalid and no further use can be made of these handles.

The read-only state of a pre-existing document does not of course prevent it from being deleted when it is of no further use. A generic document is only deleted as a result of deleting its specific document.

### **Arguments**

**document\_handle**

The handle of the DAP API document to be deleted.

### **Results**

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### **Errors**

**[DLA\_E\_INV\_DOC]**

The document handle is not valid.

**[DLA\_E\_GEN\_DOC]**

The document handle is that of a generic document. It is a limitation of the OOA API that generic documents cannot be deleted, except through deletion of the owning document.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dia\_duplicate\_specific\_entity**

## **dia\_duplicate\_specific\_entity**

Creates a copy of the specified entity and links it into the specific logical structure.

### **C Synopsis**

```
dia_status      dia_duplicate_specific_entity(  
    dia_document_handle    document_handle,  
    dia_entity_handle      model_entity,  
    dia_entity_handle      link_entity,  
    dia_link_option        link_option,  
    dia_entity_handle      *entity_handle_return);
```

### **Description**

The `dia_duplicate_specific_entity` function creates a copy of the model entity specified and links it into the specific logical structure. The new entity is fixed automatically and no further setting of properties is allowed unless it is unfixed.

The `link_entity` and `link_option` arguments specify the superior entity to which the new logical entity is to be linked and its position in the sequence of subordinates of that entity. The superior must be one for which the new entity can be added as a subordinate in the given position. That is, it must be within the DAP constraint for subordinates, and the Generator for subordinates of the ODA class of the superior must either reference the ODA class of the new entity or, in mode-1 only, must be capable of being expanded to reference it. The properties of the model entity must be appropriate for the position of the new entity. In particular the level of a model numbered segment must correspond with the level of the new one.

When the specific entity type is `NumberedSegment`, `Footnote` or `Reference`, the cluster is copied. In the case of FOD36 entities within clumps, the set of entities that is copied depends upon the particular clump that is involved. A single specific entity is copied for all other entity types. All model entities that are to be copied must be in the fixed state, and all resulting new entities are in the fixed state.

When a basic entity is duplicated, any content information that belongs to it is not copied.

### **Arguments**

#### **document\_handle**

The handle of the document being operated on.

#### **model\_entity**

The handle of a specific logical entity whose entity type is a member of the list given for the `entity_type` argument of the `dia_create_specific_entity` function.

#### **link\_entity**

The handle of the entity which, in combination with the argument `link_option`, specifies the position for the new entity in the specific logical structure.



**link\_option**

One of the values in the following table which, in combination with the argument `link_entity`, specifies the position for the new entity in the specific logical structure.

Option	Meaning
<code>DLA_C_LINK_BEFORE</code>	The new entity will be linked to the parent of <code>link_entity</code> and will precede it.
<code>DLA_C_LINK_AFTER</code>	The new entity will be linked to the parent of <code>link_entity</code> and will succeed it.
<code>DLA_C_LINK_AS_FIRST</code>	The new entity will be linked to the <code>link_entity</code> , becoming its first subordinate.
<code>DLA_C_LINK_AS_LAST</code>	The new entity will be linked to the <code>link_entity</code> , becoming its last subordinate.

**entity\_handle\_return [write]**

The handle of the new specific logical entity.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_MODEL\_INV\_ENTITY]**

The model entity does not exist.

**[DLA\_E\_LINK\_INV\_ENTITY]**

The link entity does not exist.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_ILLEGAL\_SUBORDINATE]**

The link entity type prohibits this operation.

**[DLA\_E\_ILLEGAL\_POSITION]**

The position requested for the new entity is not allowed. Applies especially to `NumberedSegment` vis a vis `Paragraph`, `BodyText`, `BodyHeader` or `BodyGeometric`.

**[DLA\_E\_NOT\_PERM\_AT\_DAP\_LEVEL]**

The link requested is not valid at the DAP level of the document.

**[DLA\_E\_GFS\_NOT\_PERM]**

The pair of entities of these types cannot be linked in this manner because the necessary Generator for subordinates entry is not permitted, is not present or (in mode-1 only) cannot be established for the current document and DAP level.

## dla\_duplicate\_specific\_entity

### [DLA\_E\_WRONG\_PARENT]

The requested structural modification is not allowed because of one of the following restrictions applying to the generic logical structure:

- The child entity is a generic Number and the child entity is already linked to an entity whose type is not the same as the parent entity.
- The child entity is a generic Reference, the parent entity is a generic FootnoteBody and the child entity is already linked to an entity that is not a generic FootnoteBody.
- The child entity is a generic Reference, the parent entity is not a generic FootnoteBody and the child entity is already linked to a generic FootnoteBody.
- The child entity is a generic EntryElement, the parent entity is a generic Row and the child entity is already linked to an entity that is not a generic Row.
- The child entity is a generic EntryElement, the parent entity is not a generic Row and the child entity is already linked to a generic Row.

### [DLA\_E\_WRONG\_NUM\_SEG\_CLASS]

The new NumberedSegment entity is not allowed to have the same class as the model. Sharing of classes is implied by duplication.

### [DLA\_E\_WRONG\_CLASS]

The new entity is not allowed to have the same class as the model. Sharing of classes is implied by duplication. This is a generalization of the NumberedSegment case for other entity types in FOC36 documents.

### [DLA\_E\_WRONG\_LEVEL]

The given context for a numbered segment would be an inconsistent depth or level of numbering. NumberedSegment entities must be used at only one level.

### [DLA\_E\_MODEL\_NOT\_FIXED]

The model entity specified is not fixed.

### [DLA\_E\_MODEL\_SUBORD\_NOT\_FIXED]

A subordinate of the model entity specified is not fixed.

### [DLA\_E\_CLUMP\_FINISHED]

The operation is not permitted because it would require that the Generator for subordinates of a finished clump be updated.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_error\_class

Returns the class of an error status.

### C Synopsis

```
dia_integer    dia_error_class(  
    dia_status    error_status);
```

### Description

The `dia_error_class` function returns the class of an earlier status return from some other function.

### Arguments

**error\_status**

An earlier status return from some other function.

### Results

An integer class number, one of those in the list given in 3.3 which describes error handling. The result of calling this function is undefined if the status return is returned by a successful function.

### Errors

None.

## **dla\_error\_number**

## **dla\_error\_number**

Returns the error number of an error status.

### **C Synopsis**

```
dla_integer    dla_error_number(  
    dla_status    error_status);
```

### **Description**

The `dla_error_number` function returns the error number of an earlier status return from some other function.

### **Arguments**

**error\_status**  
An earlier status return from some other function.

### **Results**

An integer error number, one of those in the list given in 3.3 which describes error handling. The result of calling this function is undefined if the status return is returned by a successful function.

### **Errors**

None.

## **dia\_failed**

Determines whether or not a status return refers to an error.

### **C Synopsis**

```
dia_boolean    dia_failed(  
    dia_status    status);
```

### **Description**

The `dia_failed` function determines whether or not a status return refers to an error.

### **Arguments**

**status**  
An earlier status return from some other function.

### **Results**

A boolean which is false if the status return was returned by a successful function.

### **Errors**

None.

## **dla\_find\_first\_entity**

## **dla\_find\_first\_entity**

Returns the handle of the first entity of the generic logical or layout structure, taken in an arbitrary order.

### **C Synopsis**

```
dla_status      dla_find_first_entity(  
    dla_document_handle      document_handle,  
    dla_structure_type       structure_type,  
    dla_entity_handle        *first_entity);
```

### **Description**

The `dla_find_first_entity` function returns the handle of the first entity of the specified structure, taken in an arbitrary order. The application must specify whether the handle of the first entity of the generic logical or layout structures is required.

In the case of generic logical structure the entities include all the common entities and in particular the composite ones, which are hidden when creating or navigating the structures.

### **Arguments**

#### **document\_handle**

The handle of the document being operated on.

#### **structure\_type**

The type of structure whose first entity is required. An enumerated value which is one of the following

Structure type	Meaning
DLA_C_GENERIC_LAYOUT	Generic layout first entity
DLA_C_GENERIC_LOGICAL	Generic logical first entity

#### **first\_entity [write]**

The handle of the first entity of the specified structure.



## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned.

## Errors

### [DLA\_E\_STRUCT\_ABSENT]

The requested document structure is absent. (This should not occur in conformant documents but may occur in partially completed or resource documents.)

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dla\_find\_next\_entity**

## **dla\_find\_next\_entity**

Returns the handle of the next generic entity, taken in an arbitrary order.

### **C Synopsis**

```
dla_status      dla_find_next_entity(  
    dla_document_handle    document_handle,  
    dla_entity_handle      current_entity,  
    dla_entity_handle      *next_entity);
```

### **Description**

The `dla_find_next_entity` function returns the handle of the next generic logical or layout entity, taken in an arbitrary order.

In the case of generic logical structure the entities include all the common entities and in particular the composite ones, which are hidden when creating or navigating the structures.

### **Arguments**

#### **document\_handle**

The handle of the document being operated on.

#### **current\_entity**

The handle of the generic logical or layout entity from which the next handle is required.

#### **next\_entity [write]**

The handle of the next to the current generic logical or layout entity, taken in an arbitrary order. If none exists, it receives a value of `DLA_C_NULL_HANDLE`.

### **Results**

#### **Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

### [DLA\_E\_INV\_ENTITY\_OPER]

The requested operation is not valid for an entity of this type.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dla\_find\_profile

## dla\_find\_profile

Finds the document profile entity in a document.

### C Synopsis

```
dla_status      dla_find_profile(  
    dla_document_handle    document_handle,  
    dla_entity_handle      *document_profile_handle);
```

### Description

Returns the handle of the document profile entity in the specified document.

### Arguments

**document\_handle**  
The handle of the ODA document being operated on.

**document\_profile\_handle [write]**  
The handle of the document profile entity.

### Results

**Status**  
Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**  
Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**  
An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_fix\_properties

Declares that the setting of properties for an entity is complete.

### C Synopsis

```
dia_status      dia_fix_properties(  
    dia_document_handle    document_handle,  
    dia_entity_handle      entity_handle);
```

### Description

The `dia_fix_properties` function declares that the setting of properties for a logical entity is complete.

In modes other than mode-1 this function can be applied to generic logical entities.

In the case of the principal entity of a cluster of entities, all members of the cluster are also fixed. The member entities can also be fixed separately. Note that the FootnoteText entities are not members of the Footnote cluster.

The invocation of this function may cause some conformance checking to be done. If this checking fails, additional information can be obtained by using the function `dia_get_fix_error`.

If any entity is already fixed, performing further fix functions on that entity has no effect, and is not an error.

### Arguments

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

### Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## **dla\_fix\_properties**

### **Errors**

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_DAP\_NON\_CONFORM]**

A property or properties are inconsistent or nonconformant to the given DAP (usually a missing mandatory property).

**[DLA\_E\_API\_NON\_CONFORM]**

A property or properties are inconsistent or nonconformant to API rules (for instance missing level on segment number initialization).

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_char\_set\_count**

Determines the number of escape sequences that designate character sets in accordance with ISO 2022, within the octet string read in the Graphic character sets property. All other ISO 2022 sequences are ignored. The function is for graphic character sets that can be designated for any basic entity, and for the profile properties Character content defaults, Graphic character sets, Profile character sets, Comments character sets, and Alternative repr char sets, but not for Non-basic char presentation features.

The function `dia_get_nth_char_set` can be used to extract information from an individual designating sequence.

**C Synopsis**

```
dia_status      dia_get_char_set_count(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_property_code       property_code,
    dia_integer            *count,
    dia_property_status     *property_status,
    dia_derived_location    *derived_location,
    dia_entity_handle       *derived_entity_handle);
```

**Description**

The function `dia_get_char_set_count` determines the number of character set designating escape sequences present in a character set property. Other ISO 2022 sequences are ignored. The function is for graphic character sets that can be designated for any basic entity and the document profile defaults, and for the non-basic values that can be specified in the document profile, other than Non-basic character presentation features.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the document profile being operated on.

**property\_code**

A symbolic constant representing the property being operated on.

**count [write]**

An integer that gives the number of escape sequences in the graphic character set property.

## dia\_get\_char\_set\_count

**property\_status [write]**

Receives the status of the property which is one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than **DLA\_C\_DEF\_STD\_DEF** this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is **DLA\_C\_NULL\_HANDLE**.

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_INV\_CODE]

The entity is not valid for this function.

[DLA\_E\_INV\_ESC\_SEQ]

An escape sequence within the property octet string value is badly formed.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_choice**

Reads the choice identifier value of a property applying to an entity, for a property of choice type.

**C Synopsis**

```

dia_status      dia_get_choice(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_integer            *choice_identifier,
    dia_property_status     *property_status,
    dia_derived_location    *derived_location,
    dia_entity_handle      *derived_entity_handle);

```

**Description**

The `dia_get_choice` function reads the choice identifier value and the status of a property applying to an entity of an ODA document. The property must be of choice type.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**choice\_identifier [write]**

Receives the choice identifier applying to the property. If the property status is unspecified or null, this argument is not written to.

## dia\_get\_choice

### property\_status [write]

Receives the status of a property which is one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

### derived\_location [write]

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

### derived\_entity\_handle [write]

When the value of derived\_location is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

### [DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

### [DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

### [DLA\_E\_INV\_CODE]

The property does not apply to the entity.

### [DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_INV\_CHOICE]

Underlying ODA value is not valid in the DAP, so no choice code is valid.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dla\_get\_construct\_error**

## **dla\_get\_construct\_error**

Returns the details of a failure to construct an ODIF document, as constrained by the DAP level to which the document has been defined to conform.

### **C Synopsis**

```
dla_status      dla_get_construct_error(  
    dla_document_handle    document_handle,  
    dla_entity_handle      *entity_handle_return,  
    dla_property_code       *property_code_return,  
    dla_integer             *err_num_return);
```

### **Description**

The `dla_get_construct_error` function returns the details of the previous failure to construct an ODIF document.

The information relates to attributes that are constructed by generating them from the properties used in the DAP API. During this process some inconsistencies may be discovered, or essential information may be missing. This function assists with making corrections.

If the most recent call of the function `dla_write_document` did not fail with a construction error, this function returns an error status.

### **Arguments**

**document\_handle**

The handle of the DAP API document that was being output.

**entity\_handle\_return [write]**

The handle of the entity that was being constructed when the nonconformance was found.

If no entity is directly involved with the problem then the value `DLA_C_NULL_HANDLE` is returned.

**property\_code\_return [write]**

The code for the property that was being constructed when the nonconformance was found.

If no property is directly involved with the problem then the special code `DLA_C_NO_PROPERTY` is returned.



## dia\_get\_construct\_error

**err\_num\_return [write]**

An error code that provides additional details of the construction problem encountered. This is the error number from the error status that was returned from the failed call of `dia_write_document`.

The value is one of:

**DLA\_E\_NO\_LOG\_SRC**

The given sourced generic layout entity (DLA\_C\_GEN\_SRC\_D\_CONT\_VAR or DLA\_C\_GEN\_SRC\_D\_CONT\_FIXED) does not have any common content entities.

**DLA\_E\_NO\_SUBORD**

The given composite specific logical entity does not have a subordinate.

**DLA\_E\_WRONG\_LAST\_SUBORD**

The last subordinate of a specific logical entity is of the wrong type.

**DLA\_E\_MISSING\_PEL\_SPC\_PROP**

Only one of the properties Pel Spacing, Spacing, Length and Pel Spaces has been specified.

**DLA\_E\_MISSING\_DIMENSION\_PROP**

Only one of the properties Fixed Dimension, Horizontal Dimension and Vertical Dimension has been specified.

**DLA\_E\_MISSING\_POSITION\_PROP**

Only one of the properties Position, Horizontal Position and Vertical Position has been specified.

**DLA\_E\_MISSING\_PROPERTY**

The given mandatory property has not been specified. The property code is returned in the variable `property_code_return`.

**DLA\_E\_MISSING\_LEVEL\_NUMBER**

The property level number has not been specified for a Segment Number Initialization or Resetting structure. The variable `property_code_return` specifies which type of structure.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_INV\_DOC]**

The document handle is not valid.

**[DLA\_E\_NO\_FAIL]**

The most recent call of the function `dia_write_document` did not fail with a construction error.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dla\_get\_content****dla\_get\_content**

Reads the content information associated with an entity.

**C Synopsis**

```

dla_status      dla_get_content(
    dla_document_handle    document_handle,
    dla_entity_handle      entity_handle,
    dla_length             content_offset,
    dla_length             buffer_length,
    dla_octet_string       buffer_ptr,
    dla_length             *content_length);

```

**Description**

The **dla\_get\_content** function reads the content information associated with an entity into the application supplied buffer, whose length and location are specified by the **buffer\_length** and **buffer\_ptr** arguments.

The content can be associated with the generic entity of which the given entity is an instance, or with an associated resource entity. There is no **derived\_location** argument that distinguishes these cases because it is not part of the defined ODA defaulting process. However, such content is returned if none is present on the given entity.

The **content\_offset** argument indicates the starting offset within the content information; getting the entire content requires the **content\_offset** to be set to 0. The function places the content information in the buffer, starting from the specified offset up to the length of the buffer.

The value of **content\_offset** must not exceed the length of the content information.

The length of the content information copied to the buffer is returned in the **content\_length** argument.

Content information can be stored in ODA in more than one content portion. The boundaries between content portions are not visible.

The specified entity, whose content information is required, must be an entity of one of the following types:

```

DLA_C_SPEC_BODY_TEXT
DLA_C_SPEC_BODY_GEOM
DLA_C_SPEC_BODY_RASTER
DLA_C_SPEC_FOOTNOTE_TEXT
DLA_C_SPEC_ENTRY_TEXT
DLA_C_SPEC_ENTRY_RASTER
DLA_C_SPEC_ENTRY_GEOMETRIC
DLA_C_GEN_ENTRY_TEXT
DLA_C_GEN_ENTRY_RASTER
DLA_C_GEN_ENTRY_GEOMETRIC

```

DLA\_C\_GEN\_BODY\_TEXT  
 DLA\_C\_GEN\_BODY\_GEO  
 DLA\_C\_GEN\_BODY\_RASTER  
 DLA\_C\_GEN\_FOOTNOTE\_TEXT  
 DLA\_C\_GEN\_COMMON\_TEXT  
 DLA\_C\_GEN\_COMMON\_GEO  
 DLA\_C\_GEN\_COMMON\_RASTER  
 DLA\_C\_GEN\_GENERIC\_BLOCK  
 DLA\_C\_SPEC\_GENERIC\_BLOCK  
 DLA\_C\_SPEC\_SPECIFIC\_BLOCK

The application can invoke the `dla_get_content_length` function to determine the length of the content information prior to invoking this function.

## Arguments

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**content\_offset**

The starting octet offset from which the content information is to be copied.

**buffer\_length**

The length of the application supplied buffer, expressed as the number of octets.

**buffer\_ptr [write]**

Receives the content information requested by the application.

**content\_length [write]**

Receives the length of the content information copied to the application supplied buffer.

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_INV\_OFFSET]**

The specified offset is greater than the length of the content information.

**[DLA\_E\_NO\_CONTENT]**

No content information present for this entity.

## `dla_get_content`

### `[DLA_E_REQ_ARGS_NOTSPEC]`

Not all arguments were correctly specified.

### `[DLA_E_MEM_FAIL]`

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dla\_get\_content\_length

Reads the length of the content information associated with an entity.

### C Synopsis

```
dla_status      dla_get_content_length(
    dla_document_handle    document_handle,
    dla_entity_handle      entity_handle,
    dla_length              *content_length);
```

### Description

The `dla_get_content_length` function reads the length of the content information associated with an entity.

The content can be associated with the generic entity of which the given entity is an instance, or with an associated resource entity. There is no `derived_location` argument that distinguishes these cases because it is not part of the defined ODA defaulting process. However, such content is returned if none is present on the given entity.

The following types of entity have associated content information:

```
DLA_C_SPEC_BODY_TEXT
DLA_C_SPEC_BODY_GEOM
DLA_C_SPEC_BODY_RASTER
DLA_C_SPEC_FOOTNOTE_TEXT
DLA_C_SPEC_ENTRY_TEXT
DLA_C_SPEC_ENTRY_RASTER
DLA_C_SPEC_ENTRY_GEOMETRIC
DLA_C_GEN_ENTRY_TEXT
DLA_C_GEN_ENTRY_RASTER
DLA_C_GEN_ENTRY_GEOMETRIC
DLA_C_GEN_BODY_TEXT
DLA_C_GEN_BODY_GEOM
DLA_C_GEN_BODY_RASTER
DLA_C_GEN_FOOTNOTE_TEXT
DLA_C_GEN_COMMON_TEXT
DLA_C_GEN_COMMON_GEOM
DLA_C_GEN_COMMON_RASTER
DLA_C_GEN_GENERIC_BLOCK
DLA_C_SPEC_GENERIC_BLOCK
DLA_C_SPEC_SPECIFIC_BLOCK
```



## dla\_get\_content\_length

### Arguments

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**content\_length [write]**

The length of the content information associated with the specified entity, expressed as the number of octets.

### Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for an entity of this type.

**[DLA\_E\_NO\_CONTENT]**

No content information present for this entity.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_entity\_handle**

Reads the value of a property applying to an entity, for a property of entity handle type.

**C Synopsis**

```

dia_status      dia_get_entity_handle(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_entity_handle      *property_entity_handle,
    dia_property_status     *property_status,
    dia_derived_location    *derived_location,
    dia_entity_handle      *derived_entity_handle);

```

**Description**

The `dia_get_entity_handle` function reads the value and status of a property applying to an entity of an ODA document. The property must be of entity handle type.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**property\_entity\_handle [write]**

Receives the entity handle value applying to the property. If the property status is unspecified or null, this argument is not written to.

**dla\_get\_entity\_handle****property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STO\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_NO\_CAT\_FRAME]**

The underlying attribute uses a layout category that cannot be found in the generic layout structure. Applies to properties DLA\_C\_LYD\_COL\_BRK, DLA\_C\_LYD\_IDV\_ARE and DLA\_C\_LYD\_LAY\_COL.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dla\_get\_entity\_type

## dla\_get\_entity\_type

Gets the type of an entity.

### C Synopsis

```
dla_status      dla_get_entity_type(  
    dla_document_handle    document_handle,  
    dla_entity_handle      entity_handle,  
    dla_entity_type        *entity_type);
```

### Description

The `dla_get_entity_type` function reads the type of a document entity.

### Arguments

**document\_handle**  
The handle of the document being operated on.

**entity\_handle**  
The handle of the entity being operated on.

**entity\_type [write]**  
The type of the entity.

### Results

**Status**  
Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

**[DLA\_E\_INV\_ENTITY]**  
The entity does not exist.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**  
Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**  
An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_fix\_error**

Returns the details of a failure to fix an entity, as constrained by the DAP level to which the document has been defined to conform.

**C Synopsis**

```
dia_status      dia_get_fix_error(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_property_code       *property_code_return,
    dia_integer            *err_num_return);
```

**Description**

The `dia_get_fix_error` function returns the details of the previous failure to fix an entity.

The information relates to properties that are fixed when calling `dia_fix_properties` for an entity. During this process some inconsistencies may be discovered, or essential information may be found to be missing. This function assists with discovering the cause of failure.

If the principal entity of a cluster was being fixed, the reported property code may apply to a subordinate entity.

If the most recent call of the function `dia_fix_properties` for any entity did not fail with a fixing error, this function returns an error status.

**Arguments****document\_handle**

The handle of the DAP API document instance that is in use.

**entity\_handle**

The handle of the entity that was being fixed when the nonconformance was found.

**property\_code\_return [write]**

The code for the property that was being fixed when the nonconformance was found.

**err\_num\_return [write]**

An error code that provides additional detail about the problem of fixing that was encountered. This is the error number from the error status that was returned from the failed call of `dia_fix_properties`.

## `dia_get_fix_error`

### Results

#### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

#### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

#### [DLA\_E\_NO\_FAIL]

The most recent call of the function `dia_fix_properties` did not fail with a fixing error.

#### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

#### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## dia\_get\_index\_char\_set

This function interprets the Graphic character sets property at the indexed position and returns the character set type, code area, and id encoded in the nth character set designation escape sequence. The function is for the non-basic values that can be specified in the profile for Char presentation features.

## C Synopsis

```

dia_status      dia_get_index_char_set(
    dia_document_handle    document_handle,
    dia_entity_handle      profile_handle,
    dia_integer            index,
    dia_integer            n_value,
    dia_length             buffer_length,
    dia_octet_string       character_set_id,
    dia_length             *string_length,
    dia_integer            *code_area,
    dia_integer            *character_set_type,
    dia_property_status     *property_status);

```

## Description

The function `dia_get_index_char_set` reads the non-basic graphic character sets that can be designated and/or invoked at the start of any basic component within the document. The function is for the non-basic values that can be specified in the entity. This property is also accessible in octet string form if interpretation is not required.

The index must be in the range: 1 to the number of entries in the sequence property `DLA_C_NONB_CHAR_FEATS`.

The index must be one for which the structure contains the choice property `DLA_C_NONB_CHAR_FEAT` which has the choice value `DLA_C_FEAT_GR_CHR_SET` indicating the graphic character set property, that is `DLA_C_NONB_GR_CHR_SET`.

The `n_value` must be in the range: 1 to the number of escape sequences within the property. This number can be determined by using the function `dia_get_index_char_set_count`.

The argument `code_area` indicates which one of the sets G0, G1, G2 or G3 is designated.

The graphic character set is indicated by the final byte identifier and the character set type. The final byte identifier becomes the first byte of the argument `character_set_id`. The character set type becomes the value of the argument `character_set_type` which shows whether the character set contains 94 or 96 characters and is single-byte or multi-byte.

## dia\_get\_index\_char\_set

## Arguments

**document\_handle**

The handle of the document being operated on.

**profile\_handle**

The handle of the profile being operated on.

**index**

An integer that identifies the non-basic value within the sequence of non-basic values.

**n\_value**

An integer that selects the *n*th escape sequence from a character set property. A value of 1 selects the first escape sequence.

**buffer\_length**

Length of octet string provided to receive final character identifier.

**character\_set\_id [write]**

A buffer to receive the final character identifier of a graphic character set.

**string\_length [write]**

Length of character\_set\_id returned.

**code\_area [write]**

An integer that identifies the code area for the graphic character set. The following values are applicable:

DLA\_C\_G0  
DLA\_C\_G1  
DLA\_C\_G2  
DLA\_C\_G3

**character\_set\_type [write]**

An integer whose value specifies whether the character set consists of 94 or 96 characters and is single-byte or multi-byte. The following values are defined for use:

DLA\_C\_SIN\_94  
DLA\_C\_SIN\_96  
DLA\_C\_MUL\_94  
DLA\_C\_MUL\_96

**property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_CODE]**

The entity is not valid for this function.

**[DLA\_E\_INDEX\_OUT\_OF\_RANGE]**

The index is out of the range of valid indexes for the entries in the sequence.

**[DLA\_E\_INDEX\_INVALID]**

The index value is not one with a correct choice property.

**[DLA\_E\_INV\_ESC\_SEQ]**

An escape sequence within the property octet string value is badly formed.

**[DLA\_E\_INV\_N\_VALUE]**

There are less than n\_value escape sequences within the property octet string value.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_index\_char\_set\_count****dia\_get\_index\_char\_set\_count**

Determines the number of designating escape sequences in the indexed graphic character set property. Other ISO 2022 sequences are ignored. The function is for the non-basic values that can be specified in the profile for Character presentation features.

**C Synopsis**

```
dia_status      dia_get_index_char_set_count(
    dia_document_handle    document_handle,
    dia_entity_handle      profile_handle,
    dia_integer            index,
    dia_integer            *count,
    dia_property_status    *property_status);
```

**Description**

The function `dia_get_index_char_set_count` determines the number of ISO 2022 escape sequences designating character encodings in the non-basic graphic character sets for a given index. The function is for the non-basic values that can be specified in the profile for Character presentation features.

The index must be in the range: 1 to the number of entries in the sequence property `DLA_C_NONB_CHAR_FEATS`.

The index must be one for which the structure contains the choice property `DLA_C_NONB_CHAR_FEAT` which has the choice value `DLA_C_FEAT_GR_CHAR_SET` indicating the graphic character set property, that is `DLA_C_NONB_GR_CHAR_SET`.

The function `dia_get_index_char_set` can be used to extract information from an individual designating sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**profile\_handle**

The handle of the profile being operated on.

**index**

An integer that identifies the non-basic value within the sequence of non-basic values.

**count [write]**

An integer that gives the number of escape sequences in the non-basic graphic character sets that can be designated for any basic component within the document.

property\_status [write]

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_INV\_ENTITY]

The profile does not exist.

### [DLA\_E\_INV\_CODE]

The entity is not valid for this function.

### [DLA\_E\_INDEX\_OUT\_OF\_RANGE]

The index is out of the range of valid indexes for the entries in the sequence.

### [DLA\_E\_INDEX\_INVALID]

The index value is not one with a correct choice property.

### [DLA\_E\_INV\_ESC\_SEQ]

An escape sequence within the property octet string value is badly formed.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dla\_get\_index\_choice****dla\_get\_index\_choice**

Reads the choice identifier value at an indexed position in the sequence of values of a property applying to an entity, for a property of sequence of choice type.

**C Synopsis**

```

dla_status      dla_get_index_choice(
    dla_document_handle    document_handle,
    dla_entity_handle     entity_handle,
    dla_structure_handle   structure_handle,
    dla_property_code      property_code,
    dla_integer            index,
    dla_integer            *choice_identifier,
    dla_property_status     *property_status,
    dla_derived_location    *derived_location,
    dla_entity_handle       *derived_entity_handle);

```

**Description**

The **dla\_get\_index\_choice** function reads the choice identifier value and status at an indexed position in the sequence of values of a property applying to an entity. The property must be of sequence of choice type.

The index must be in the range: 1 to the number of entries in the sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of **DLA\_C\_NULL\_HANDLE** must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**choice\_identifier [write]**

Receives the choice identifier applying to the property. If the property status is unspecified or null, this argument is not written to.



**property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than **DLA\_C\_DEF\_STD\_DEF** this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is **DLA\_C\_NULL\_HANDLE**.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the **entity\_handle** at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

## `dla_get_index_choice`

### **[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

### **[DLA\_E\_INDEX\_OUT\_OF\_RNG]**

The index is out of the range of valid indexes for the entries in the sequence.

### **[DLA\_E\_INV\_CHOICE]**

Underlying ODA value is not valid in the DAP, so no choice code is valid.

### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_index\_entity**

Reads the entity handle value at an indexed position in the sequence of values of a property applying to an entity, for a property of sequence of entity handle type.

**C Synopsis**

```

dia_status      dia_get_index_entity(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle   structure_handle,
    dia_property_code      property_code,
    dia_integer            index,
    dia_entity_handle      *property_entity_handle,
    dia_property_status     *property_status,
    dia_derived_location   *derived_location,
    dia_entity_handle      *derived_entity_handle);

```

**Description**

The `dia_get_index_entity` function reads the entity handle value and status at an indexed position in the sequence of values of a property applying to an entity. The property must be of sequence of entity handle type.

The index must be in the range: 1 to the number of entries in the sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**property\_entity\_handle [write]**

Receives the entity handle value applying to the property. If the property status is unspecified or null, this argument is not written to.

## dla\_get\_index\_entity

### property\_status [write]

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

### derived\_location [write]

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

### derived\_entity\_handle [write]

When the value of derived\_location is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

### [DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

### [DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

### [DLA\_E\_INV\_CODE]

The property does not apply to the entity.

### [DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_INDEX\_OUT\_OF\_RANGE]

The index is out of the range of valid indexes for the entries in the sequence.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dla\_get\_index\_integer****dla\_get\_index\_integer**

Reads the integer at an indexed position in the sequence of values of a property applying to an entity, for a property of sequence of integer type.

**C Synopsis**

```

dla_status      dla_get_index_integer(
    dla_document_handle    document_handle,
    dla_entity_handle      entity_handle,
    dla_structure_handle   structure_handle,
    dla_property_code      property_code,
    dla_integer            index,
    dla_integer            *integer_value,
    dla_property_status    *property_status,
    dla_derived_location   *derived_location,
    dla_entity_handle      *derived_entity_handle);

```

**Description**

The **dla\_get\_index\_integer** function reads the integer value and status at the specified indexed position in the sequence of values of a property applying to an entity. The property must be sequence of integer type.

The index specified must in the range: 1 to the number of entries in the sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of **DLA\_C\_NULL\_HANDLE** must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in a sequence is at index 1.

**integer\_value [write]**

Receives the integer value applying to the property. If the property status is unspecified or null, this argument is not written to.



**property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the **entity\_handle** at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

## `dla_get_index_integer`

### **[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

### **[DLA\_E\_INDEX\_OUT\_OF\_RANGE]**

The index is out of the range of valid indexes for the entries in the sequence.

### **[DLA\_E\_INV\_INT\_VALUE]**

The integer value found is not valid within the given DAP.

### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dla\_get\_index\_length**

Reads the length of the value of a property applying to an entity, for an indexed property of octet string type.

**C Synopsis**

```
dla_status      dla_get_index_length(
    dla_document_handle document_handle,
    dla_entity_handle  entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code   property_code,
    dla_integer         index,
    dla_length          *string_length,
    dla_property_status  *property_status,
    dla_derived_location *derived_location,
    dla_entity_handle    *derived_entity_handle);
```

**Description**

The `dla_get_index_length` function reads the length of the octet string and the status of an indexed property applying to an entity. The indexed property must be of octet string type.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**string\_length [write]**

Receives the length of the octet string applying to the property. If the property status is unspecified or null, this argument is not written to.

**dla\_get\_index\_length****property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the **entity\_handle** at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_INDEX\_OUT\_OF\_RNG]**

The index is out of the range of valid indexes for the entries in the sequence.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_index\_string****dia\_get\_index\_string**

Reads the value of a property applying to an entity, for an indexed property of octet string type.

**C Synopsis**

```

dia_status      dia_get_index_string(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle   structure_handle,
    dia_property_code      property_code,
    dia_integer            index,
    dia_length             buffer_length,
    dia_octet_string       string_value,
    dia_length             *string_length,
    dia_property_status    *property_status,
    dia_derived_location   *derived_location,
    dia_entity_handle      *derived_entity_handle);

```

**Description**

The `dia_get_index_string` function reads the value and status of an indexed property applying to an entity of an ODA document. The indexed property must be of octet string type.

The application specifies a buffer into which the octet string is returned. Octet strings that have a length greater than the buffer are truncated.

The length of the octet string applying to the property is returned in the `string_length` argument as a number of octets.

The application can invoke the `dia_get_index_length` function prior to invoking this function to determine the length of the string.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.



**index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**buffer\_length**

The length of the buffer in the string\_value argument, expressed as the number of octets.

**string\_value [write]**

The buffer that receives the octet string applying to the property. If the property status is unspecified or null this argument is not written to.

**string\_length [write]**

Receives the length of the octet string applying to the property. If the property status is unspecified or null, this argument is not written to.

**property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of derived\_location is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

dla\_get\_index\_string

## Errors

[DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

[DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

[DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

[DLA\_E\_INV\_CODE]

The property does not apply to the entity.

[DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_INDEX\_OUT\_OF\_RNG]

The index is out of the range of valid indexes for the entries in the sequence.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_index\_structure**

Reads the structure handle value at an indexed position in the sequence of values of a property applying to an entity, for a property of sequence of structure type.

**C Synopsis**

```

dia_status      dia_get_index_structure(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_integer             index,
    dia_structure_handle    *new_structure_handle,
    dia_property_status      *property_status,
    dia_derived_location     *derived_location,
    dia_entity_handle       *derived_entity_handle);

```

**Description**

The `dia_get_index_structure` function reads the structure handle value and status at an indexed position in the sequence of values of a property applying to an entity. The property must be of sequence of structure handle type.

The index must be in the range: 1 to the number of entries in the sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**new\_structure\_handle [write]**

Receives the structure handle value applying to the property. If the property status is unspecified or null, this argument is not written to.

**dla\_get\_index\_structure****property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than **DLA\_C\_DEF\_STD\_DEF** this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is **DLA\_C\_NULL\_HANDLE**.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the **entity\_handle** at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INDEX\_OUT\_OF\_RANGE]**

The index is out of the range of valid indexes for the entries in the sequence.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dla\_get\_integer**

## **dla\_get\_integer**

Reads the value of a property applying to an entity, for a property of integer type.

### **C Synopsis**

```
dla_status      dla_get_integer(  
    dla_document_handle    document_handle,  
    dla_entity_handle      entity_handle,  
    dla_structure_handle    structure_handle,  
    dla_property_code       property_code,  
    dla_integer             *integer_value,  
    dla_property_status     *property_status,  
    dla_derived_location    *derived_location,  
    dla_entity_handle       *derived_entity_handle);
```

### **Description**

The `dla_get_integer` function reads the value and status of a property applying to an entity of an ODA document. The property must be of integer type.

### **Arguments**

#### **document\_handle**

The handle of the document being operated on.

#### **entity\_handle**

The handle of the entity being operated on.

#### **structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

#### **property\_code**

A symbolic constant representing the property being operated on.

#### **integer\_value [write]**

Receives the integer value applying to the property. If the property status is unspecified or null, this argument is not written to.



**property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

## `dla_get_integer`

### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

### [DLA\_E\_INV\_INT\_VALUE]

The integer value found is not valid within the given DAP.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_get\_interp\_error

Returns details of a failure to interpret an ODIF document, as constrained by the DAP level to which the document is intended to conform.

## C Synopsis

```

dia_status      dia_get_interp_error(
    dia_toolkit_handle toolkit_handle,
    dia_length      ident_buffer_length,
    dia_octet_string ident_return_buffer,
    dia_length      *ident_length_return,
    dia_entity_type *entity_type_return,
    dia_property_code *property_code_return,
    dia_integer      *err_msg_return);

```

## Description

The `dia_get_interp_error` function returns the details of the previous failure to interpret an ODIF document.

The information relates to attributes that are interpreted in the process of rendering them into the form of properties used in the DAP API.

If the most recent call of the function `dia_read_document` or `dia_read_generic_doc` did not fail with an interpretation error, this function returns an error status.

## Arguments

**toolkit\_handle**

The handle of the DAP API toolkit instance that is in use.

**ident\_buffer\_length**

The length of the buffer provided by the application to receive the identifier of the erroneous entity.

**ident\_return\_buffer [write]**

The address of the buffer provided by the application to receive the identifier of the erroneous entity.

**ident\_length\_return [write]**

The length of the identifier of the erroneous entity. If the document profile is faulty then this indicates zero octets because the profile does not have an identifier.

**entity\_type\_return [write]**

The type of the entity that was being interpreted when the nonconformance was found. In addition to the valid types, this may be `DLA_C_TYP_UNDEFINED`.

## dla\_get\_interp\_error

property\_code\_return [write]

The code for the property that was being interpreted when the nonconformance was found.

err\_num\_return [write]

A numeric code that provides additional detail of the interpretation problem that was encountered. This is the error number from the error status that was returned from the failed call of dla\_read\_document.

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_INV\_TOOLKIT]

The toolkit handle is not valid.

### [DLA\_E\_NO\_FAIL]

The most recent call of the function dla\_read\_document did not fail with an interpretation error.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_get\_length

Reads the length of the value of a property applying to an entity, for a property of octet string type.

## C Synopsis

```
dia_status      dia_get_length(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_length             *string_length,
    dia_property_status     *property_status,
    dia_derived_location    *derived_location,
    dia_entity_handle       *derived_entity_handle);
```

## Description

The `dia_get_length` function reads the length of the octet string and the status of a property applying to an entity. The property must be of octet string type.

## Arguments

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**string\_length [write]**

Receives the length of the octet string applying to the property. If the property status is unspecified or null, this argument is not written to.

**dia\_get\_length****property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the **entity\_handle** at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.



[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dla\_get\_nth\_char\_set****dla\_get\_nth\_char\_set**

This function interprets the *n*th ISO 2022 character set designation escape sequence from a character set property. The function is for the Graphic character set property of basic objects and the profile defaults graphic character sets, and for any non-basic values that are specified in graphic character sets designated for the document profile. That is, Profile character sets, Comments character sets and Alternative repr char sets, but not Non-basic char presentation features, for which there is a separate function.

**C Synopsis**

```

dla_status      dla_get_nth_char_set (
    dla_document_handle    document_handle,
    dla_entity_handle      entity_handle,
    dla_property_code      property_code,
    dla_integer            n_value,
    dla_length             buffer_length,
    dla_octet_string       character_set_id,
    dla_length             *string_length,
    dla_integer            *code_area,
    dla_integer            *character_set_type,
    dla_property_status    *property_status,
    dla_derived_location    *derived_location,
    dla_entity_handle      *derived_entity_handle);

```

**Description**

The function `dla_get_nth_char_set` reads and interprets the *n*th ISO 2022 character set designation escape sequence from a character set property. Other ISO 2022 sequences are ignored. The function is for the Graphic character set property of basic objects, for document profile default character sets, and for any non-basic values that are specified for graphic character sets designated in the document profile, other than Char presentation features. These properties are also accessible in octet string form if interpretation is not required.

The *n\_value* must be in the range: 1 to the number of escape sequences within the property. This number can be determined by using the function `dla_get_char_set_count`.

The argument *code\_area* indicates which one of the sets G0, G1, G2 or G3 is designated.

The graphic character set is indicated by the final byte identifier and the character set type. The final byte identifier becomes the first byte of the argument *character\_set\_id*. The character set type becomes the value of the argument *character\_set\_type*, which shows whether the character set contains 94 or 96 characters and is single-byte or multi-byte.

## Arguments

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the document profile being operated on.

**property\_code**

A symbolic constant representing the property being operated on.

**n\_value**

An integer that selects the *n*th escape sequence from a character set property. A value of 1 selects the first escape sequence.

**buffer\_length**

Length of the octet string provided to receive the final character identifier.

**character\_set\_id [write]**

A buffer to receive the final character identifier of a graphic character set.

**string\_length [write]**

Length of character\_set\_id returned.

**code\_area [write]**

An integer that identifies the code area for the graphic character set. The following values are applicable:

DLA\_C\_G0  
DLA\_C\_G1  
DLA\_C\_G2  
DLA\_C\_G3

**character\_set\_type [write]**

An integer whose value specifies whether the character set consists of 94 or 96 characters and is single-byte or multi-byte. The following values are defined for use:

DLA\_C\_SIN\_94  
DLA\_C\_SIN\_96  
DLA\_C\_MUL\_94  
DLA\_C\_MUL\_96

**property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.

## **dia\_get\_nth\_char\_set**

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than **DLA\_C\_DEF\_STD\_DEF** this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is **DLA\_C\_NULL\_HANDLE**.

## **Results**

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## **Errors**

[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_INV\_CODE]

The entity is not valid for this function.

[DLA\_E\_INV\_ESC\_SEQ]

An escape sequence within the property octet string value is badly formed.

[DLA\_E\_INV\_N\_VALUE]

There are less than *n\_value* escape sequences within the property octet string value.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_ola\_read\_error**

Returns the details of an error in reading an ODIF document, as determined by the ODA API.

**C Synopsis**

```

dia_status      dia_get_ola_read_error(
    dia_toolkit_handle toolkit_handle,
    ola_constituent_type *constituent_type_return,
    ola_integer          *byte_position_return,
    ola_attribute_code   *attribute_code_return,
    ola_integer          *ola_err_num_return);

```

**Description**

The `dia_get_ola_read_error` function returns the details of the previous failure to read an ODIF document, either with `dia_read_document` or with `dia_read_generic_doc`.

The information that is returned is the same as that returned by the corresponding ODA API function, with one exception. The ODA API function returns a status object that classifies the fault in the ODIF. In the DAP API function, the returned error status has been rendered as an error number by the ODA API function `ola_error_number`.

**Arguments****toolkit\_handle**

The handle of the DAP API toolkit instance that is in use. This is converted to the handle for the underlying ODA API toolkit instance for use in the corresponding ODA API call.

**constituent\_type\_return [write]**

Refer to the ODA API documentation for a description of the corresponding argument in the underlying ODA API call.

**byte\_position\_return [write]**

Refer to the ODA API documentation for a description of the corresponding argument in the underlying ODA API call.

**attribute\_code\_return [write]**

Refer to the ODA API documentation for a description of the corresponding argument in the underlying ODA API call.

**ola\_err\_num\_return [write]**

The ODA API function returns a status object that classifies the fault in the ODIF. In the DAP API function the returned error status has been rendered as the corresponding error number by the ODA API function `ola_error_number`.

## `dla_get_ola_read_error`

### Results

#### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

#### [DLA\_E\_INV\_TOOLKIT]

The toolkit handle is not valid.

#### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

#### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_ola\_write\_error**

Returns the details of an error in writing an ODIF document, as determined by the ODA API.

**C Synopsis**

```

dia_status      dia_get_ola_write_error(
    dia_document_handle    document_handle,
    dia_constituent_handle *constituent_handle_return,
    dia_attribute_code      *attribute_code_return,
    dia_integer             *ola_err_num_return);

```

**Description**

The `dia_get_ola_write_error` function returns the details of the previous failure to write an ODIF document.

The information that is returned is the same as that returned by the corresponding ODA API function, with one exception. The ODA API function returns a status object that classifies the fault in the ODIF. In the DAP API function the returned error status has been rendered as an error number by the ODA API function `ola_error_number`.

**Arguments****document\_handle**

The handle of the DAP API document that is in use. This is converted to the handle for the underlying ODA API document for use in the corresponding ODA API call.

**constituent\_handle\_return [write]**

Refer to the ODA API documentation for a description of the corresponding argument in the underlying ODA API call.

**attribute\_code\_return [write]**

Refer to the ODA API documentation for a description of the corresponding argument in the underlying ODA API call.

**ola\_err\_num\_return [write]**

The ODA API function returns a status object that classifies the fault in the ODIF. In the DAP API function the returned error status has been rendered as the corresponding error number by the ODA API function `ola_error_number`.

## `dia_get_ola_write_error`

### Results

#### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

#### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

#### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_get\_other\_parent

This function makes it possible, in the case of formatted processable documents, to identify the basic logical and layout entities that both pertain to a given part of the content.

### C Synopsis

```

dia_status      dia_get_other_parent(
    dia_document_handle    document_handle,
    dia_entity_handle       entity_handle,
    dia_length              content_offset,
    dia_entity_handle       *other_parent_handle,
    dia_length              *other_content_offset);

```

### Description

This function makes it possible, in the specific structures of formatted processable documents, to identify the basic logical and layout entities that both pertain to a given octet of the content. Starting from a basic logical entity, a basic layout entity (specific block) is found. Starting from a basic layout entity (specific block), a basic logical entity is found.

Not all specific blocks have content that originates in the specific logical structure. If the content originates as generic content of a generic block, from a common content entity (via a logical source attribute), or from a content generator, then this function fails with an error response.

### Arguments

#### document\_handle

The handle of the document being operated on.

#### entity\_handle

The handle of the specific basic logical or block entity.

#### content\_offset

The offset in octets of the particular octet of interest, from the start of the content that belongs to the entity\_handle.

#### other\_parent\_handle [write]

Returns the handle of the corresponding entity from the other specific structure. The hidden content portion that contains the identified octet belongs both to other\_parent\_handle and to entity\_handle.

#### other\_content\_offset [write]

Returns the offset in octets from the start of the content that belongs to other\_parent\_handle of the particular octet of interest.

## dla\_get\_other\_parent

### Results

#### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

#### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

#### [DLA\_E\_DOC\_NOT\_FPDA]

The document is not a formatted processable document.

#### [DLA\_E\_NO\_CONTENT]

No content information present for this entity.

#### [DLA\_E\_NOT\_BASIC\_SPEC\_ENTITY]

The given entity is not a basic specific logical or layout entity.

#### [DLA\_E\_INV\_OFFSET]

Insufficient content information present for the value of offset.

#### [DLA\_E\_CONTENT\_NOT\_LAID\_OUT]

This specific logical content does not have a correspondence to the specific layout structure. In a conformant formatted processable document all specific logical content should have been laid out.

#### [DLA\_E\_CONTENT\_NOT\_SPEC\_LOG]

This specific layout content does not have a correspondence to the specific logical structure. The content probably originates as generic content of a generic block, from a common content entity via a logical source attribute, or from a content generator.

#### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

#### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_size**

Reads the number of entries in a sequence of values of a property applying to an entity, for a property of sequence type.

**C Synopsis**

```

dia_status      dia_get_size(
    dia_document_handle document_handle,
    dia_entity_handle entity_handle,
    dia_structure_handle structure_handle,
    dia_property_code property_code,
    dia_integer *sequence_size,
    dia_property_status *property_status,
    dia_derived_location *derived_location,
    dia_entity_handle *derived_entity_handle);

```

**Description**

The function `dia_get_size` reads the number of entries in a sequence of values of a property applying to an entity. The property must be of sequence type.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DIA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**sequence\_size [write]**

Receives the number of entries in the sequence of values applying to the property. If the property status is unspecified or null, this argument is not written to.

**dia\_get\_size****property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than **DLA\_C\_DEF\_STD\_DEF** this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is **DLA\_C\_NULL\_HANDLE**.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the **entity\_handle** at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.



**[DLA\_E\_INV\_CODE]**

The type of the entity is not valid for this function.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_char\_set\_count****dia\_get\_spec\_char\_set\_count**

Determines the number of escape sequences that designate character sets in accordance with ISO2022, within the octet string read in the Graphic character sets property. All other ISO 2022 sequences are ignored. Using the function with `_spec_` on a specific entity, rather than using the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile. The function is for graphic character sets designated for any basic entity.

**C Synopsis**

```
dia_status      dia_get_spec_char_set_count(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_property_code      property_code,
    dia_integer            *count,
    dia_property_status     *property_status);
```

**Description**

The function `dia_get_spec_char_set_count` determines the number of character set designating escape sequences present in a character set property. Other ISO 2022 sequences are ignored. The function is for graphic character sets designated for any basic entity and the document profile defaults, and for the non-basic values specified in the document profile other, than Non-basic character presentation features.

The function `dia_get_spec_nth_char_set` can be used to extract information from an individual designating sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the document profile being operated on.

**property\_code**

A symbolic constant representing the property being operated on.

**count [write]**

An integer that gives the number of escape sequences in the Graphic character set property.

property\_status [write]

Receives the status of the property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

### [DLA\_E\_INV\_CODE]

The entity is not valid for this function.

### [DLA\_E\_INV\_ESC\_SEQ]

An escape sequence within the property octet string value is badly formed.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_choice****dia\_get\_spec\_choice**

Reads the choice identifier value of a property applying to an entity, for a property of choice type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to generic sources of property data from the generic entity or profile.

**C Synopsis**

```
dia_status      dia_get_spec_choice(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle   structure_handle,
    dia_property_code      property_code,
    dia_integer            *choice_identifier,
    dia_property_status    *property_status);
```

**Description**

The `dia_get_spec_choice` function reads the choice identifier value and the status of a property applying to an entity of an ODA document. The property must be of choice type.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for this argument.

**property\_code**

A symbolic constant representing the property being operated on.

**choice\_identifier [write]**

Receives the choice identifier applying to the property. If the property status is unspecified or null, this argument is not written to.

**property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_CHOICE]**

Underlying ODA value is not valid in DAP, so no choice code is valid.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dla\_get\_spec\_content****dla\_get\_spec\_content**

Reads the content information associated with an entity. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to data from the class.

**C Synopsis**

```

dla_status      dla_get_spec_content(
    dla_document_handle    document_handle,
    dla_entity_handle     entity_handle,
    dla_length            content_offset,
    dla_length            buffer_length,
    dla_octet_string      buffer_ptr,
    dla_length            *content_length);

```

**Description**

The `dla_get_spec_content` function reads the content information associated with an entity into the application supplied buffer, whose length and location are specified by the `buffer_length` and `buffer_ptr` arguments.

The ODA attribute Content information is not a defaultable attribute and is not directly represented by a DAP API property. The distinction between `dla_get_spec_content` and `dla_get_content` is provided in order to model the behaviour of content information in a document.

The `content_offset` argument indicates the starting offset within the content information. To get the entire content, the `content_offset` must be set to 0. The function places the content information in the buffer, starting from the specified offset, up to the length of the buffer.

The value of `content_offset` must not exceed the length of the content information.

The length of the content information copied to the buffer is returned in the `content_length` argument.

Content information can be stored in ODA in more than one content portion. The boundaries between content portions are not visible.

The entity specified, whose content information is required, must be an entity of one of the following types:

```

DLA_C_SPEC_BODY_TEXT
DLA_C_SPEC_BODY_GEOM
DLA_C_SPEC_BODY_RASTER
DLA_C_SPEC_FOOTNOTE_TEXT
DLA_C_SPEC_ENTRY_TEXT
DLA_C_SPEC_ENTRY_RASTER
DLA_C_SPEC_ENTRY_GEOMETRIC
DLA_C_GEN_ENTRY_TEXT
DLA_C_GEN_ENTRY_RASTER

```



DLA\_C\_GEN\_ENTRY\_GEOMETRIC  
 DLA\_C\_GEN\_BODY\_TEXT  
 DLA\_C\_GEN\_BODY\_GEO  
 DLA\_C\_GEN\_BODY\_RASTER  
 DLA\_C\_GEN\_FOOTNOTE\_TEXT  
 DLA\_C\_GEN\_COMMON\_TEXT  
 DLA\_C\_GEN\_COMMON\_GEO  
 DLA\_C\_GEN\_COMMON\_RASTER  
 DLA\_C\_GEN\_GENERIC\_BLOCK  
 DLA\_C\_SPEC\_GENERIC\_BLOCK  
 DLA\_C\_SPEC\_SPECIFIC\_BLOCK

The application can invoke the `dia_get_spec_content_length` function to determine the length of the content information, prior to invoking this function.

## Arguments

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**content\_offset**

The starting octet offset from which the content information is to be copied.

**buffer\_length**

The length of the application supplied buffer, expressed as the number of octets.

**buffer\_ptr [write]**

Receives the content information requested by the application.

**content\_length[write]**

Receives the length of the content information copied to the application supplied buffer.

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is invalid for an entity of this type.

**[DLA\_E\_INV\_OFFSET]**

The specified offset is greater than the length of the content information.

## `dia_get_spec_content`

### `[DLA_E_NO_CONTENT]`

No content information present for this entity.

### `[DLA_E_REQ_ARGS_NOTSPEC]`

Not all arguments were correctly specified.

### `[DLA_E_MEM_FAIL]`

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_get\_spec\_content\_length

Reads the length of the content information associated with an entity. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the class.

## C Synopsis

```
dia_status      dia_get_spec_content_length(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_length             *content_length);
```

## Description

The `dia_get_spec_content_length` function reads the length of the content information associated with an entity.

The ODA attribute Content information is not a defaultable attribute and is not directly represented by a DAP API property. The distinction between `dia_get_spec_content` and `dia_get_content` is provided in order to model the behaviour of content information in a document.

The following types of entity have associated content information:

```
DLA_C_SPEC_BODY_TEXT
DLA_C_SPEC_BODY_GEOM
DLA_C_SPEC_BODY_RASTER
DLA_C_SPEC_FOOTNOTE_TEXT
DLA_C_SPEC_ENTRY_TEXT
DLA_C_SPEC_ENTRY_RASTER
DLA_C_SPEC_ENTRY_GEOMETRIC
DLA_C_GEN_ENTRY_TEXT
DLA_C_GEN_ENTRY_RASTER
DLA_C_GEN_ENTRY_GEOMETRIC
DLA_C_GEN_BODY_TEXT
DLA_C_GEN_BODY_GEOM
DLA_C_GEN_BODY_RASTER
DLA_C_GEN_FOOTNOTE_TEXT
DLA_C_GEN_COMMON_TEXT
DLA_C_GEN_COMMON_GEOM
DLA_C_GEN_COMMON_RASTER
DLA_C_GEN_GENERIC_BLOCK
DLA_C_SPEC_GENERIC_BLOCK
DLA_C_SPEC_SPECIFIC_BLOCK
```

## **dia\_get\_spec\_content\_length**

### **Arguments**

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**content\_length [write]**

The length of the content information associated with the specified entity, expressed as the number of octets.

### **Results**

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### **Errors**

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is invalid for an entity of this type.

**[DLA\_E\_NO\_CONTENT]**

No content information present for this entity.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_entity\_handle**

Reads the value of a property applying to an entity, for a property of entity handle type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity.

**C Synopsis**

```

dia_status      dia_get_spec_entity_handle(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_entity_handle      *property_entity_handle,
    dia_property_status     *property_status);

```

**Description**

The `dia_get_spec_entity_handle` function reads the value and status of a property applying to an entity of an ODA document. The property must be of entity handle type.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for this argument.

**property\_code**

A symbolic constant representing the property being operated on.

**property\_entity\_handle [write]**

Receives the entity handle value applying to the property. If the property status is unspecified or null, this argument is not written to.

**dla\_get\_spec\_entity\_handle****property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_NO\_CAT\_FRAME]**

The underlying attribute uses a layout category that cannot be found in the generic layout structure. Applies to properties DLA\_C\_LYD\_COL\_BRK, DLA\_C\_LYD\_IDV\_ARE and DLA\_C\_LYD\_LAY\_COL.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_spec\_index\_choice**

Reads the choice identifier value at an indexed position in the sequence of values of a property applying to an entity, for a property of sequence of choice type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```
dia_status      dia_get_spec_index_choice(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_integer            index,
    dia_integer            *choice_identifier,
    dia_property_status     *property_status);
```

**Description**

The `dia_get_spec_index_choice` reads the choice identifier value and status at an indexed position in the sequence of values of a property applying to an entity. The property must be of sequence of choice type.

The index must be in the range: 1 to the number of entries in the sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for this argument.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**choice\_identifier [write]**

Receives the choice identifier applying to the property. If the property status is unspecified or null, this argument is not written to.

## dia\_get\_spec\_index\_choice

**property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned.

## Errors

**[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INDEX\_OUT\_OF\_RNG]**

The index is out of the range of valid indexes for the entries in the sequence.

**[DLA\_E\_INV\_CHOICE]**

Underlying ODA value is not valid in DAP, so no choice code is valid.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_index\_entity**

Reads the entity handle value at an indexed position in the sequence of values of a property applying to an entity, for a property of sequence of entity handle type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity.

**C Synopsis**

```

dia_status      dia_get_spec_index_entity(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code      property_code,
    dia_integer            index,
    dia_entity_handle      *property_entity_handle,
    dia_property_status     *property_status);

```

**Description**

The `dia_get_spec_index_entity` reads the entity handle value and status at an indexed position in the sequence of values of a property applying to an entity. The property must be of sequence of entity handle type.

The index must be in the range: 1 to the number of entries in the sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for this argument.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**property\_entity\_handle [write]**

Receives the entity handle value applying to the property. If the property status is unspecified or null, this argument is not written to.

## dla\_get\_spec\_index\_entity

### property\_status [write]

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

### [DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

### [DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

### [DLA\_E\_INV\_CODE]

The property does not apply to the entity.

### [DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

### [DLA\_E\_INDEX\_OUT\_OF\_RNG]

The index is out of the range of valid indexes for the entries in the sequence.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_spec\_index\_integer**

Reads the integer at an indexed position in the sequence of values of a property applying to an entity, for a property of sequence of integer type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```

dia_status      dia_get_spec_index_integer(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_integer            index,
    dia_integer            *integer_value,
    dia_property_status     *property_status);

```

**Description**

The `dia_get_spec_index_integer` function reads the integer value and status at the specified indexed position in the sequence of values of a property applying to an entity. The property must be sequence of integer type.

The index specified must in the range: 1 to the number of entries in the sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for this argument.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in a sequence is at index 1.

**integer\_value [write]**

Receives the integer value applying to the property. If the property status is unspecified or null, this argument is not written to.

## dla\_get\_spec\_index\_integer

property\_status [write]

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

## Results

Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

[DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

[DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

[DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

[DLA\_E\_INV\_CODE]

The property does not apply to the entity.

[DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_INDEX\_OUT\_OF\_RANGE]

The index is out of the range of valid indexes for the entries in the sequence.

[DLA\_E\_INV\_INT\_VALUE]

The integer value found is not valid within the given DAP.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_spec\_index\_length**

Reads the length of the value of a property applying to an entity, for an indexed property of octet string type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```
dia_status      dia_get_spec_index_length(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_integer            index,
    dia_length             *string_length,
    dia_property_status     *property_status);
```

**Description**

The `dia_get_spec_index_length` function reads the length of the octet string and the status of an indexed property applying to an entity. The indexed property must be of octet string type.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for the argument.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**string\_length [write]**

Receives the length of the octet string applying to the property. If the property status is unspecified or null, this argument is not written to.

## dla\_get\_spec\_index\_length

## property\_status [write]

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

## Results

## Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

## [DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

## [DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

## [DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

## [DLA\_E\_INV\_CODE]

The property does not apply to the entity.

## [DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

## [DLA\_E\_INV\_ENTITY]

The entity does not exist.

## [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

## [DLA\_E\_INDEX\_OUT\_OF\_RNG]

The index is out of the range of valid indexes for the entries in the sequence.

## [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_index\_string**

Reads the value of a property applying to an entity, for an indexed property of octet string type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```

dia_status      dia_get_spec_index_string(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle   structure_handle,
    dia_property_code      property_code,
    dia_integer            index,
    dia_length             buffer_length,
    dia_octet_string       string_value,
    dia_length             *string_length,
    dia_property_status    *property_status);

```

**Description**

The `dia_get_spec_index_string` function reads the value and status of an indexed property applying to an entity of an ODA document. The indexed property must be of octet string type.

The application specifies a buffer into which the octet string is returned. Octet strings with a length greater than the buffer are truncated.

The length of the octet string applying to the property is returned in the `string_length` argument as a number of octets.

The application can invoke the `dia_get_spec_index_length` function prior to invoking this function to determine the length of the string.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for this argument.

**property\_code**

A symbolic constant representing the property being operated on.

**dla\_get\_spec\_index\_string****Index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**buffer\_length**

The length of the buffer in the string\_value argument, expressed as the number of octets.

**string\_value [write]**

The buffer which receives the octet string applying to the property. If the property status is unspecified or null this argument is not written to.

**string\_length [write]**

Receives the length of the octet string applying to the property. If the property status is unspecified or null, this argument is not written to.

**property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INDEX\_OUT\_OF\_RNG]**

The index is out of the range of valid indexes for the entries in the sequence.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_spec\_index\_structure****dia\_get\_spec\_index\_structure**

Reads the structure handle value at an indexed position in the sequence of values of a property applying to an entity, for a property of sequence of structure type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```

dia_status      dia_get_spec_index_structure(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_integer            index,
    dia_structure_handle    *new_structure_handle,
    dia_property_status     *property_status);

```

**Description**

The `dia_get_spec_index_structure` reads the structure handle value and status at an indexed position in the sequence of values of a property applying to an entity. The property must be of sequence of structure handle type.

The index must be in the range: 1 to the number of entries in the sequence.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for this argument.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index identifying the entry within the sequence, whose value is required. The first entry in the sequence is at index 1.

**new\_structure\_handle [write]**

Receives the structure handle value applying to the property. If the property status is unspecified or null, this argument is not written to.



property\_status [write]

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

## Results

Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned.

## Errors

[DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

[DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

[DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

[DLA\_E\_INV\_CODE]

The property does not apply to the entity.

[DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_INDEX\_OUT\_OF\_RANGE]

The index is out of the range of valid indexes for the entries in the sequence.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dla\_get\_spec\_integer**

## **dla\_get\_spec\_integer**

Reads the value of a property applying to an entity, for a property of integer type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

### **C Synopsis**

```
dla_status      dla_get_spec_integer(  
    dla_document_handle    document_handle,  
    dla_entity_handle      entity_handle,  
    dla_structure_handle    structure_handle,  
    dla_property_code       property_code,  
    dla_integer             *integer_value,  
    dla_property_status     *property_status);
```

### **Description**

The `dla_get_spec_integer` function reads the value and status of a property applying to an entity of an ODA document. The property must be of integer type.

### **Arguments**

#### **document\_handle**

The handle of the document being operated on.

#### **entity\_handle**

The handle of the entity being operated on.

#### **structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for this argument.

#### **property\_code**

A symbolic constant representing the property being operated on.

#### **integer\_value [write]**

Receives the integer value applying to the property. If the property status is unspecified or null, this argument is not written to.

**property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_INT\_VALUE]**

The integer value found is not valid within the given DAP.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_length****dia\_get\_spec\_length**

Reads the length of the value of a property applying to an entity, for a property of octet string type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```

dia_status      dia_get_spec_length(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_length             *string_length,
    dia_property_status     *property_status);

```

**Description**

The `dia_get_spec_length` function reads the length of the octet string and the status of a property applying to an entity. The property must be of octet string type.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for the argument.

**property\_code**

A symbolic constant representing the property being operated on.

**string\_length [write]**

Receives the length of the octet string applying to the property. If the property status is unspecified or null, this argument is not written to.

**property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dla\_get\_spec\_nth\_char\_set****dla\_get\_spec\_nth\_char\_set**

This function interprets the *n*th ISO 2022 character set designation escape sequence from a character set property. Using the function with *\_spec\_* on a specific entity, rather than the corresponding function without *\_spec\_*, inhibits reference to defaulting sources of property data from the generic entity or profile. The function is for the Graphic character set property of basic objects.

**C Synopsis**

```
dla_status      dla_get_spec_nth_char_set(
    dla_document_handle    document_handle,
    dla_entity_handle      entity_handle,
    dla_property_code      property_code,
    dla_integer            n_value,
    dla_length             buffer_length,
    dla_octet_string       character_set_id,
    dla_length             *string_length,
    dla_integer            *code_area,
    dla_integer            *character_set_type,
    dla_property_status     *property_status);
```

**Description**

The function *dla\_get\_spec\_nth\_char\_set* reads and interprets the *n*th ISO 2022 character set designation escape sequence from a character set property. Other ISO 2022 sequences are ignored. The function is for the Graphic character set property of basic objects, document profile default character sets, and the non-basic values that can be specified for graphic character sets designated in the document profile (other than Char presentation features). These properties are also accessible in octet string form if interpretation is not required.

The *n\_value* must be in the range: 1 to the number of escape sequences within the property. This number can be determined by using the function *dla\_get\_spec\_char\_set\_count*.

The argument *code\_area* indicates which one of the sets G0, G1, G2 or G3 is designated.

The graphic character set is indicated by the final byte identifier and the character set type. The final byte identifier becomes the first byte of the argument *character\_set\_id*. The character set type becomes the value of the argument *character\_set\_type* which shows whether the character set contains 94 or 96 characters and is single-byte or multi-byte.



## Arguments

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the document profile being operated on.

**property\_code**

A symbolic constant representing the property being operated on.

**n\_value**

An integer that selects the *n*th escape sequence from a character set property. A value of 1 selects the first escape sequence.

**buffer\_length**

Length of octet string provided to receive final character identifier.

**character\_set\_id [write]**

A buffer to receive the final character identifier of a graphic character set.

**string\_length [write]**

Length of character\_set\_id returned.

**code area [write]**

An integer that identifies the code area for the graphic character set. The following values are applicable:

DLA\_C\_G0  
DLA\_C\_G1  
DLA\_C\_G2  
DLA\_C\_G3

**character\_set\_type [write]**

An integer whose value specifies whether the character set consists of 94 or 96 characters and is single-byte or multi-byte. The following values are defined for use:

DLA\_C\_SIN\_94  
DLA\_C\_SIN\_96  
DLA\_C\_MUL\_94  
DLA\_C\_MUL\_96

**property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

`dla_get_spec_nth_char_set`

## Errors

[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_INV\_CODE]

The entity is not valid for this function.

[DLA\_E\_INV\_ESC\_SEQ]

An escape sequence within the property octet string value is badly formed.

[DLA\_E\_INV\_N\_VALUE]

There are less than `n_value` escape sequences within the property octet string value.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_size**

Reads the number of entries in a sequence of values of a property applying to an entity for a property of sequence type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```

dia_status      dia_get_spec_size(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle   structure_handle,
    dia_property_code      property_code,
    dia_integer            *sequence_size,
    dia_property_status    *property_status);

```

**Description**

The function `dia_get_spec_size` reads the number of entries in a sequence of values of a property applying to an entity. The property must be of sequence type.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DIA_C_NULL_HANDLE` should be specified for this argument.

**property\_code**

A symbolic constant representing the property being operated on.

**sequence\_size [write]**

Receives the number of entries in the sequence of values applying to the property. If the property status is unspecified or null, this argument is not written to.

## dla\_get\_spec\_size

### property\_status [write]

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

### [DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

### [DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

### [DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

### [DLA\_E\_INV\_CODE]

The type of the entity is not valid for this function.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_status**

Gets the status of a property applying to an entity. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```
dia_status      dia_get_spec_status(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle   structure_handle,
    dia_property_code      property_code,
    dia_property_status    *property_status);
```

**Description**

The `dia_get_spec_status` function reads the status of a property applying to an entity of an COA document.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for the argument.

**property\_code**

A symbolic constant representing the property being operated on.

**dla\_get\_spec\_status****property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

The property status DLA\_C\_PROP\_NULL can only occur for a property that is not specified as a CHOICE property but can take the value null. The value of null causes the status DLA\_C\_PROP\_NULL to be returned.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_spec\_string**

Reads the value of a property applying to an entity, for a property of octet string type. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```

dia_status      dia_get_spec_string(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle   structure_handle,
    dia_property_code      property_code,
    dia_length            buffer_length,
    dia_octet_string      string_value,
    dia_length            *string_length,
    dia_property_status    *property_status);

```

**Description**

The `dia_get_spec_string` function reads the value and status of a property applying to an entity of an ODA document. The property must be of octet string type.

The application specifies a buffer into which the octet string is returned. Octet strings with a length greater than the buffer are truncated.

The length of the octet string applying to the property is returned in the `string_length` argument as a number of octets.

The application can invoke the `dia_get_spec_length` function prior to invoking this function to determine the length of the string.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If not used, a value of `DLA_C_NULL_HANDLE` should be specified for this argument.

**property\_code**

A symbolic constant representing the property being operated on.

**dla\_get\_spec\_string****buffer\_length**

The length of the buffer in the string\_value argument, expressed as the number of octets.

**string\_value [write]**

The buffer which receives the octet string applying to the property. If the property status is unspecified or null this argument is not written to.

**string\_length [write]**

Receives the length of the octet string applying to the property. If the property status is unspecified or null, this argument is not written to.

**property\_status [write]**

Receives the status of a property. One of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified
DLA_C_PROP_UNSPECIFIED	The property status is unspecified
DLA_C_PROP_NULL	The property status is null

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_tab\_stop**

Reads the details of a tabulation stop. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```

dia_status      dia_get_spec_tab_stop(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_integer            index,
    dia_length             *tab_reference_length,
    dia_integer            *tab_position,
    dia_tab_alignment      *alignment,
    dia_length             *alignment_string_length);

```

**Description**

The `dia_get_spec_tab_stop` function reads the details of a tabulation stop. The tabulation stop is specified by its index in the line layout table that applies to the specified entity.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**index**

An integer value representing the index of the tabulation stop in the line layout table.

**tab\_reference\_length [write]**

Receives the length of the tabulation reference, in octets.

**tab\_position [write]**

The integer value representing the tabulation position.

**dia\_get\_spec\_tab\_stop****alignment [write]**

The alignment type of the tabulation stop at the specified position.

An enumerated value that is one of:

Alignment type	Meaning
DLA_C_TAB_ALIGN_START	Start aligned
DLA_C_TAB_ALIGN_END	End aligned
DLA_C_TAB_ALIGN_CENTRED	Center aligned
DLA_C_TAB_ALIGN_AROUND	Align around

**alignment\_string\_length [write]**

Receives the length of the alignment string, in octets.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_INDEX\_OUT\_OF\_RANGE]**

The index is out of range for valid indexes for the entries in the line layout table.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_spec\_tab\_stop\_strings**

Reads the tab reference and alignment strings from the tab stop at the specified indexed position in the line layout table. Using the function with `_spec_` on a specific entity, rather than the corresponding function without `_spec_`, inhibits reference to defaulting sources of property data from the generic entity or profile.

**C Synopsis**

```

dia_status      dia_get_spec_tab_stop_strings(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_index              index,
    dia_length             tab_reference_buffer_length,
    dia_length             alignment_string_buffer_length,
    dia_octet_string        tab_reference_string,
    dia_length             *tab_reference_length,
    dia_octet_string        alignment_string,
    dia_length             *alignment_string_length);

```

**Description**

The `dia_get_spec_tab_stop_strings` reads the tab reference and alignment strings of a tabulation stop. The tabulation is specified by its index in the line layout table that applies to the specified entity.

The application specifies buffers into which the tab reference and alignment octet strings are returned. Octet strings with a length greater than the buffer are truncated.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**index**

An integer value representing the index of the tabulation stop in the line layout table.

**tab\_reference\_buffer\_length**

The length in octets of the buffer in the `tab_reference` argument.

**alignment\_string\_buffer\_length**

The length in octets of the buffer in the `alignment_string` argument.

**tab\_reference\_string [write]**

Receives the octet string representing the required tab reference. Specify `DIA_C_NULL_STRING` if the tab reference is not required.



## **dla\_get\_spec\_tab\_stop\_strings**

### **tab\_reference\_length [write]**

Receives the length of the tab reference octet string, expressed as the number of octets. This is not written if the tab\_reference argument is DLA\_C\_NULL\_STRING.

### **alignment\_string [write]**

Receives the octet string representing the alignment string. Specify DLA\_C\_NULL\_STRING if the alignment string is not required.

### **alignment\_string\_length [write]**

Receives the integer value representing the alignment string length, expressed as the number of octets. This is not written if the alignment\_string argument is DLA\_C\_NULL\_STRING.

## **Results**

### **Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## **Errors**

### **[DLA\_E\_INDEX\_OUT\_OF\_RANGE]**

The index is out of range for valid indexes for the entries in the line layout table.

### **[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for this entity.

### **[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_status**

Gets the status of a property applying to an entity.

**C Synopsis**

```
dia_status      dia_get_status(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_property_status     *property_status,
    dia_derived_location    *derived_location,
    dia_entity_handle       *derived_entity_handle);
```

**Description**

The `dia_get_status` function reads the status of a property applying to an entity of an ODA document.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**dia\_get\_status****property\_status [write]**

Receives the status of a property; one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

The property status DLA\_C\_PROP\_NULL can only occur for a property that is not specified as a CHOICE property but can take the value null. The value of null causes the status DLA\_C\_PROP\_NULL to be returned.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of derived\_location is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

[DLA\_E\_INV\_ENTITY]

The entity does not exist.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dla\_get\_string****dla\_get\_string**

Reads the value of a property applying to an entity, for a property of octet string type.

**C Synopsis**

```

dla_status      dla_get_string(
    dla_document_handle    document_handle,
    dla_entity_handle     entity_handle,
    dla_structure_handle   structure_handle,
    dla_property_code      property_code,
    dla_length             buffer_length,
    dla_octet_string        string_value,
    dla_length             *string_length,
    dla_property_status     *property_status,
    dla_derived_location    *derived_location,
    dla_entity_handle       *derived_entity_handle);

```

**Description**

The **dla\_get\_string** function reads the value and status of a property applying to an entity of an ODA document. The property must be of octet string type.

The application specifies a buffer into which the octet string is returned. Octet strings that have a length greater than the buffer are truncated.

The length of the octet string applying to the property is returned in the **string\_length** argument as a number of octets.

The application can invoke the **dla\_get\_length** function prior to invoking this function, in order to determine the length of the string.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of **DLA\_C\_NULL\_HANDLE** must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**buffer\_length**

The length of the buffer in the **string\_value** argument, expressed as the number of octets.

**string\_value [write]**

The buffer that receives the octet string applying to the property. If the property status is unspecified or null this argument is not written to.

**string\_length [write]**

Receives the length of the octet string applying to the property. If the property status is unspecified or null, this argument is not written to.

**property\_status [write]**

Receives the status of a property, one of the following:

Status	Meaning
DLA_C_PROP_SPECIFIED	The property status is specified.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.
DLA_C_PROP_NULL	The property status is null.

**derived\_location [write]**

One of the following values:

DLA\_C\_DEF\_ENTITY  
DLA\_C\_DEF\_STYLE  
DLA\_C\_DEF\_CLASS  
DLA\_C\_DEF\_CLASS\_STYLE  
DLA\_C\_DEF\_REF\_CLASS  
DLA\_C\_DEF\_REF\_STYLE  
DLA\_C\_DEF\_VAL\_LIST  
DLA\_C\_DEF\_DAP\_DEF  
DLA\_C\_DEF\_STD\_DEF

Meanings for these values are given at the beginning of 3.4.

**derived\_entity\_handle [write]**

When the value of **derived\_location** is other than DLA\_C\_DEF\_STD\_DEF this parameter returns the handle of the entity in which the property value was found to be specified. In this exception case the value returned is DLA\_C\_NULL\_HANDLE.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

## `dia_get_string`

### **[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the `entity_handle` at the DAP level of the document, but would be valid if the DAP level of the document were different.

### **[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

### **[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

### **[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_get\_tab\_stop**

Reads the details of a tabulation stop.

**C Synopsis**

```

dia_status      dia_get_tab_stop(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_integer            index,
    dia_length             *tab_reference_length,
    dia_integer            *tab_position,
    dia_tab_alignment      *alignment,
    dia_length             *alignment_string_length);

```

**Description**

The `dia_get_tab_stop` function reads the details of a tabulation stop. The tabulation stop is specified by its index in the line layout table that applies to the specified entity.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**index**

An integer value representing the index of the tabulation stop in the line layout table.

**tab\_reference\_length [write]**

Receives the length of the tabulation reference, in octets.

**tab\_position [write]**

The integer value representing the tabulation position.

**dla\_get\_tab\_stop****alignment [write]**

The alignment type of the tabulation stop at the specified position. An enumerated value that is one of:

Alignment type	Meaning
DLA_C_TAB_ALIGN_START	Start aligned
DLA_C_TAB_ALIGN_END	End aligned
DLA_C_TAB_ALIGN_CENTRED	Center aligned
DLA_C_TAB_ALIGN_AROUND	Align around

**alignment\_string\_length [write]**

Receives the length of the alignment string, in octets.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_INDEX\_OUT\_OF\_RANGE]**

The index is out of range for valid indexes for the entries in the line layout table.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_get\_tab\_stop\_strings**

Reads the tab reference and alignment strings from the tab stop at the specified indexed position in the line layout table.

**C Synopsis**

```

dia_status      dia_get_tab_stop_strings(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_integer            index,
    dia_length             tab_reference_buffer_length,
    dia_length             alignment_string_buffer_length,
    dia_octet_string       tab_reference_string,
    dia_length             *tab_reference_length,
    dia_octet_string       alignment_string,
    dia_length             *alignment_string_length);

```

**Description**

The `dia_get_tab_stop_strings` reads the tab reference and alignment strings of a tabulation stop. The tabulation stop is specified by its index in the line layout table that applies to the specified entity.

The application specifies buffers into which the tab reference and alignment octet strings are returned. Octet strings that have a length greater than the buffer are truncated.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**index**

An integer value representing the index of the tabulation stop in the line layout table.

**tab\_reference\_buffer\_length**

The length in octets of the buffer in the `tab_reference` argument.

**alignment\_string\_buffer\_length**

The length in octets of the buffer in the `alignment_string` argument.

**tab\_reference\_string [write]**

Receives the octet string representing the required tab reference. Specify `DIA_C_NULL_STRING` if the tab reference is not required.

## **dla\_get\_tab\_stop\_strings**

### **tab\_reference\_length [write]**

Receives the length of the tab reference octet string, expressed as the number of octets. This is not written if the tab\_reference argument is DLA\_C\_NULL\_STRING.

### **alignment\_string [write]**

Receives the octet string representing the alignment string. Specify DLA\_C\_NULL\_STRING if the alignment string is not required.

### **alignment\_string\_length [write]**

Receives the integer value representing the alignment string length, expressed as the number of octets. This is not written if the alignment\_string argument is DLA\_C\_NULL\_STRING.

## **Results**

### **Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## **Errors**

### **[DLA\_E\_INDEX\_OUT\_OF\_RANGE]**

The index is out of range for valid indexes for the entries in the line layout table.

### **[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for this entity.

### **[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_init\_toolkit**

Initializes an instance of the DAP Level API toolkit.

**C Synopsis**

```

dia_status      dia_init_toolkit(
    dia_toolkit_options toolkit_options,
    dia_integer      toolkit_memory_allocation,
    dia_integer      max_open_files,
    dia_toolkit_handle *toolkit_handle);

```

**Description**

The `dia_init_toolkit` function initializes an instance of the DAP Level API toolkit and an instance of the ODA level API toolkit. This function must be called before any document processing is performed.

The application can restrict the amount of system memory available to the toolkit by specifying a non zero value for the `toolkit_memory_allocation` parameter.

**Arguments****toolkit\_options**

Application toolkit options for this toolkit instance; reserved for future use.

Option type	Meaning
DLA_C_NO_OPTIONS	No options are specified.

**toolkit\_memory\_allocation**

This specifies how much system memory can be allocated by the toolkit. If the implementation does not require this argument, specify zero. This information is for the use of the ODA level API.

**max\_open\_files**

The maximum number of files that are permitted to be open simultaneously. This information is for the use of the ODA level API, which treats the value zero as special and substitutes a default value.

**toolkit\_handle [write]**

The handle of the toolkit instance.

## `dla_init_toolkit`

### Results

#### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

#### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

#### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## dia\_instantiate\_spec\_log

Creates a specific logical entity as an instance of a given generic logical entity and links it into the specific logical structure.

## C Synopsis

```

dia_status      dia_instantiate_spec_log(
    dia_document_handle    document_handle,
    dia_entity_handle      generic_entity,
    dia_entity_handle      link_entity,
    dia_link_option        link_option,
    dia_entity_handle      *entity_handle_return);

```

## Description

The `dia_instantiate_spec_log` function creates a specific entity of the type established by the type of the cited generic entity. The only types allowed are those that can be created by `dia_create_gen_log_entity`, whether the generic entity was actually so created or is contained in a generic document.

The `link_entity` and `link_option` arguments specify the superior entity to which the new logical entity is to be linked and its position in the sequence of subordinates of that entity. The superior must be one for which the new entity can be added as a subordinate. That is, the Generator for subordinates of the superior class must reference the class of the new entity.

In some cases the DAP constraints are such that the position of subordinates is restricted. This may cause the requested position of the new entity to be invalid and the function to fail. Failure does not occur merely because the Generator for subordinates of the class of the superior is more restrictive than is required by the DAP constraints. In mode-1 and mode-2 the Generator for subordinates is never more restrictive than is required by the DAP constraints. Such additional degrees of restriction arise only with external document classes and are not taken into account.

When the entity type of the `generic_entity` is `DLA_C_GEN_LOG_ROOT` the value of the `link_entity` parameter must be `DLA_C_NULL_HANDLE`.

When the entity type of the `generic_entity` is `NumberedSegment`, `Footnote` or `Reference`, a cluster is created.

## Arguments

**document\_handle**

The handle of the document being operated on.

**generic\_entity**

The handle of the generic logical entity of which the new entity is to be an instance.

**dla\_instantiate\_spec\_log****link\_entity**

The handle of the entity which, in combination with the link\_option, specifies the position for the new entity in the specific logical structure.

**link\_option**

One of the values below which, in combination with the argument link\_entity, specifies the position for the new entity in the specific logical structure.

Option value	Meaning
DLA_C_LINK_BEFORE	The new entity will be linked to the parent of link_entity and will precede it
DLA_C_LINK_AFTER	The new entity will be linked to the parent of link_entity and will succeed it
DLA_C_LINK_AS_FIRST	The new entity will be linked to the link_entity, becoming its first subordinate
DLA_C_LINK_AS_LAST	The new entity will be linked to the link_entity, becoming its last subordinate

**entity\_handle\_return [write]**

The handle of the new specific logical entity.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_GEN\_INV\_ENTITY]**

The generic\_entity does not exist or is not a generic logical entity.

**[DLA\_E\_GEN\_WRONG\_ENTITY]**

The generic\_entity is not one that can be instantiated.

**[DLA\_E\_LINK\_INV\_ENTITY]**

The link entity does not exist.

**[DLA\_E\_DUP\_ROOT]**

A specific logical root already exists.

**[DLA\_E\_ILLEGAL\_SUBORDINATE]**

The link entity cannot have a subordinate of the type of the given generic logical entity.

**[DLA\_E\_NOT\_PERM\_AT\_DAP\_LEVEL]**

The link entity cannot have a subordinate of the type of the given generic logical entity at the DAP level of the document.

**[DLA\_E\_ILLEGAL\_POSITION]**

The link entity cannot have such a subordinate entity at the given position at the DAP level of the document.

[DLA\_E\_DOC\_READ\_ONLY]

The document is read only.

[DLA\_E\_INV\_MODE]

The writing mode of this document is not appropriate for this function.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

[DLA\_E\_CLUMP\_NOT\_FINISHED]

The generic entity is a clump whose shape is not yet determined and thus no instances may be created at this time (See section 2.2.2.4 "Creating and linking clumps").

## **dla\_link\_generic\_entity**

## **dla\_link\_generic\_entity**

Specifies an entity to which another entity should be linked in one of the generic structures.

This function permits the use of an entity at positions in the generic structure in addition to the one at which the entity was created.

### **C Synopsis**

```
dla_status      dla_link_generic_entity(  
    dla_document_handle    document_handle,  
    dla_entity_handle      parent_entity_handle,  
    dla_entity_handle      child_entity_handle);
```

### **Description**

The `dla_link_generic_entity` function specifies that the entity specified by the argument `child_entity_handle` should be appended to the Generator for subordinates of the entity specified by `parent_entity_handle`.

The function ensures that the resultant Generator for subordinates satisfies DAP constraints.

This function permits the reuse of entities in the generic structures.

Pages must have consistent Page layout types, from the point of view of any frames that are to be shared. Page layout types are consistent if they are the same. In addition, if `child_entity_handle` is a Body frame or a frame within a Body frame, Page layout types C and D are consistent; if `child_entity_handle` is a Header or Footer frame, Page layout types A, B and C are consistent. The purpose of this consistency check is to ensure that the Layout path attribute is consistent with FOD26.

### **Arguments**

**document\_handle**

The handle of the document being operated on.

**parent\_entity\_handle**

The handle of the entity whose Generator for subordinates is being updated.

**child\_entity\_handle**

The handle of the entity being added to a Generator for subordinates.



## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_INV\_PARENT\_ENTITY]

The parent entity is not in the document.

### [DLA\_E\_INV\_CHILD\_ENTITY]

The child entity is not in the document.

### [DLA\_E\_INV\_MODE]

The writing mode of this document is not appropriate for this function.

### [DLA\_E\_BODY\_AREA\_EXISTS]

There can be only one body area on a page and it is already present.

### [DLA\_E\_HEADER\_EXISTS]

There can be only one header area on a page and it is already present.

### [DLA\_E\_FOOTER\_EXISTS]

There can be only one footer area on a page and it is already present.

### [DLA\_E\_FNOTE\_AREA\_EXISTS]

There can be only one footnote area within a frame and it is already present. (Restriction additional to level 3 DAP constraint.)

### [DLA\_E\_COLUMN\_AREA\_LINKED]

The column area is already linked subordinate to the given VariableCompositeBody. This is the only case to which this restriction applies.

### [DLA\_E\_SNAKING\_REP]

A CompositeColumn/Variable frame cannot be added to a Snaking Column frame where the earlier call to `dla_create_generic_snaking` specified `DLA_C_SNAKING_REP` as the value of `snaking_type` and a subordinate generic frame is already established.

### [DLA\_E\_WRONG\_PARENT]

The requested structural modification is not allowed because of one of the following restrictions applying to the generic logical structure:

- The child entity is a generic Number and the child entity is already linked to an entity whose type is not the same as the parent entity.
- The child entity is a generic Reference, the parent entity is a generic FootnoteBody and the child entity is already linked to an entity that is not a generic FootnoteBody.
- The child entity is a generic Reference, the parent entity is not a generic FootnoteBody and the child entity is already linked to a generic FootnoteBody.
- The child entity is a generic EntryElement, the parent entity is a generic Row and the child entity is already linked to an entity that is not a generic Row.
- The child entity is a generic EntryElement, the parent entity is not a generic Row and the child entity is already linked to a generic Row.

## **dla\_link\_generic\_entity**

### **[DLA\_E\_WRONG\_LEVEL]**

The given context for a NumberedSegment or FOD36 Passage, (see 2.2.2.3, which describes creating and linking Passages and NumberedSegments) would be an inconsistent depth or level of numbering. NumberedSegment entities must be used at only one level.

### **[DLA\_E\_RECURSIVE\_LINK\_INV]**

This entity cannot be linked to itself directly or indirectly, (practical restriction for cases such as EntryGroup and EntryGroupArea). Direct and indirect recursion are detected for NumberedSegments (and for some FOD36 Passages) in the check for wrong depth or level of numbering.

### **[DLA\_E\_GFS\_NOT\_PERM]**

The pair of entities of these types cannot be linked in this manner because the resulting Generator for subordinates is not permitted for the current DAP level.

### **[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

### **[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for a (parent) entity of this type.

### **[DLA\_E\_PAG\_LAY\_NOTSPEC]**

The parent entity handle is a generic page handle, but its Page Layout Type has not been set.

### **[DLA\_E\_PAG\_LAY\_DIFF]**

The parent entity (or a higher level parent) is a generic page whose Page Layout Type is inconsistent with the Page Layout Type of the generic Page for which the child\_entity\_handle was created.

### **[DLA\_E\_CLUMP\_FINISHED]**

The operation is not permitted because it would require that the Generator for subordinates of a finished clump be updated.

### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_link\_generic\_entity\_seq**

Specifies an entity to which another entity should be linked in a sequential part of one of the generic structures.

This function permits the use of an entity at a particular position in the generic structure in addition to the one in which the entity was created.

**C Synopsis**

```
dia_status      dia_link_generic_entity_seq(
    dia_document_handle    document_handle,
    dia_entity_handle      parent_entity_handle,
    dia_integer            index,
    dia_entity_handle      child_entity_handle);
```

**Description**

The `dia_link_generic_entity_seq` function specifies that the entity specified by the argument `child_entity_handle` should be included in the Generator for subordinates of the entity specified by `parent_entity_handle`.

The function ensures that the resultant Generator for subordinates satisfies DAP constraints.

This function should be used rather than `dia_link_generic_entity` when the entity should not be appended to the sequence of entities in the Generator for subordinates. This may apply because that position is not conformant with DAP constraints or because it would not produce the desired effect.

The linking is done by index and not by link entity because a link entity is generally not unique in a Generator for subordinates (unlike the case for a specific structure).

The index argument must have a valid value for the required position (starting at one) of the entity. Thus the new entry precedes an existing one at that position.

**Arguments****document\_handle**

The handle of the document being operated on.

**parent\_entity\_handle**

The handle of the entity whose Generator for subordinates is being updated.

**index**

The index of the entry at which the value is to be stored. The first entry in a sequence is at index 1.

**child\_entity\_handle**

The handle of the entity being added to a Generator for subordinates.

## `dla_link_generic_entity_seq`

### Results

#### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

#### [DLA\_E\_INV\_PARENT\_ENTITY]

The parent entity is not in the document.

#### [DLA\_E\_INV\_CHILD\_ENTITY]

The child entity is not in the document.

#### [DLA\_E\_INV\_MODE]

The writing mode of this document is not appropriate for this function.

#### [DLA\_E\_INDEX\_OUT\_OF\_RANGE]

The index is out of range for valid indexes for the entries in the Generator for subordinates.

#### [DLA\_E\_BODY\_AREA\_EXISTS]

There can only be one body area on a page and it is already present.

#### [DLA\_E\_HEADER\_EXISTS]

There can be only one header area on a page and it is already present.

#### [DLA\_E\_FOOTER\_EXISTS]

There can be only one footer area on a page and it is already present.

#### [DLA\_E\_FNOTE\_AREA\_EXISTS]

There can be only one footnote area within a frame and it is already present. (Restriction additional to level 3 DAP constraint.)

#### [DLA\_E\_COLUMN\_AREA\_LINKED]

The column area is already linked subordinate to the given VariableCompositeBody. This is the only case to which this restriction applies.

#### [DLA\_E\_SNAKING\_REP]

A CompositeColumnVariable frame cannot be added to a Snaking Column frame where the earlier call to `dla_create_generic_snaking` specified `DLA_C_SNAKING_REP` as the value of `snaking_type` and a subordinate generic frame is already established.

**[DLA\_E\_WRONG\_PARENT]**

The requested structural modification is not allowed because of one of the following restrictions applying to the generic logical structure:

- The child entity is a generic Number and the child entity is already linked to an entity whose type is not the same as the parent entity.
- The child entity is a generic Reference, the parent entity is a generic FootnoteBody and the child entity is already linked to an entity that is not a generic FootnoteBody.
- The child entity is a generic Reference, the parent entity is not a generic FootnoteBody and the child entity is already linked to a generic FootnoteBody.
- The child entity is a generic EntryElement, the parent entity is a generic Row and the child entity is already linked to an entity that is not a generic Row.
- The child entity is a generic EntryElement, the parent entity is not a generic Row and the child entity is already linked to a generic Row.

**[DLA\_E\_WRONG\_LEVEL]**

The given context for a NumberedSegment or FOD36 Passage, (see 2.2.2.3, which describes creating and linking Passages and NumberedSegments) would be an inconsistent depth or level of numbering. NumberedSegment entities must be used at only one level.

**[DLA\_E\_RECURSIVE\_LINK\_INV]**

This entity cannot be linked to itself directly or indirectly, (practical restriction for cases such as EntryGroup and EntryGroupArea). Direct and indirect recursion are detected for NumberedSegments (and for some FOD36 Passages) in the check for wrong depth or level of numbering.

**[DLA\_E\_GFS\_NOT\_PERM]**

The pair of entities of these types cannot be linked in this manner because the resulting Generator for subordinates is not permitted for the current DAP level.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for a (parent) entity of this type.

**[DLA\_E\_PAG\_LAY\_NOTSPEC]**

The parent entity handle is a generic page handle, but its Page Layout Type has not been set.

**[DLA\_E\_PAG\_LAY\_DIFF]**

The parent entity (or a higher level parent) is a generic page whose Page Layout Type is inconsistent with the Page Layout Type of the generic Page for which the child\_entity\_handle was created.

**[DLA\_E\_CLUMP\_FINISHED]**

The operation is not permitted because it would require that the Generator for subordinates of a finished clump be updated.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dla\_locate\_gen\_entity**

## **dla\_locate\_gen\_entity**

Locates a generic entity using its ODIIF identifier.

### **C Synopsis**

```
dla_status      dla_locate_gen_entity(  
    dla_document_handle    document_handle,  
    dla_octet_string       entity_id,  
    dla_length             entity_id_length,  
    dla_entity_handle      *entity_handle_return,  
    dla_entity_type        *entity_type_return);
```

### **Description**

The `dla_locate_gen_entity` function locates a generic entity having the identifier specified. It also gives the DAP API code for the DAP type of the entity.

This function is intended for use when creating a document using a generic document.

If this function is used to locate an entity in a resource document then the handle of the resource (generic) document must be used to ensure that the required entity is located. Otherwise either the handle of the generic document or of the specific document may be given.

### **Arguments**

**document\_handle**

The handle of the document being operated on.

**entity\_id**

The ODIIF identifier of the required entity.

**entity\_id\_length**

The length of the ODIIF identifier of the required entity in octets.

**entity\_handle\_return [write]**

The handle of the located generic entity.



**entity\_type\_return [write]**

Indicates the type of DAP entity found. The value supplied is one of the following:

Entity type	Meaning
DLA_C_GEN_LOG_ROOT	Generic DocumentLogicalRoot
DLA_C_GEN_PASSAGE	Generic Passage
DLA_C_GEN_NUM_SEG	Generic NumberedSegment
DLA_C_GEN_NUMBER	Generic Number
DLA_C_GEN_TITLE	Generic Title
DLA_C_GEN_CAPTION	Generic Caption
DLA_C_GEN_PARAGRAPH	Generic Paragraph
DLA_C_GEN_PHRASE	Generic Phrase
DLA_C_GEN_FOOTNOTE	Generic Footnote
DLA_C_GEN_FOOTNOTE_NUM	Generic FootnoteNumber
DLA_C_GEN_FOOTNOTE_REF	Generic FootnoteReference
DLA_C_GEN_FOOTNOTE_BODY	Generic FootnoteBody
DLA_C_GEN_FOOTNOTE_TEXT	Generic FootnoteText
DLA_C_GEN_FIGURE	Generic Figure
DLA_C_GEN_BODY_TEXT	Generic BodyText
DLA_C_GEN_REFERENCE	Generic Reference
DLA_C_GEN_REF_CONTENT	Generic ReferencedContent
DLA_C_GEN_BODY_RASTER	Generic BodyRaster
DLA_C_GEN_BODY_GEOM	Generic BodyGeometric
DLA_C_GEN_COMMON_CONTENT	Generic CommonContent
DLA_C_GEN_COMMON_TEXT	Generic CommonText
DLA_C_GEN_COMMON_REFERENCE	Generic CommonReference
DLA_C_GEN_COMMON_NUMBER	Generic CommonNumber
DLA_C_GEN_CURRENT_INSTANCE	Generic CurrentInstance
DLA_C_GEN_COMMON_RASTER	Generic CommonRaster
DLA_C_GEN_COMMON_GEOM	Generic CommonGeometric
DLA_C_GEN_PAGE_NUMBER	Generic PageNumber
DLA_C_GEN_TABLE_NUMBER	Generic TableNumber
DLA_C_GEN_DESCRIPTION	Generic Description
DLA_C_GEN_ARTWORK	Generic Artwork
DLA_C_GEN_NUMBERED_LIST	Generic NumberedList
DLA_C_GEN_UNNUMBERED_LIST	Generic UnNumberedList
DLA_C_GEN_DEFINITION_LIST	Generic DefinitionList
DLA_C_GEN_LIST_ITEM	Generic ListItem
DLA_C_GEN_LIST_TERM	Generic ListTerm
DLA_C_GEN_TABLE	Generic Table

## dia\_locate\_gen\_entity

Entity type	Meaning
DLA_C_GEN_ROW	Generic Row
DLA_C_GEN_TABLE_COMPONENT	Generic TableComponent
DLA_C_GEN_ROW_COMPONENT	Generic RowComponent
DLA_C_GEN_FORM	Generic Form
DLA_C_GEN_ENTRY_ELEMENT	Generic EntryElement
DLA_C_GEN_ENTRY_GROUP	Generic EntryGroup
DLA_C_GEN_ENTRY_TEXT	Generic EntryText
DLA_C_GEN_ENTRY_RASTER	Generic EntryRaster
DLA_C_GEN_ENTRY_GEOMETRIC	Generic EntryGeometric
DLA_C_GEN_LAY_ROOT	Generic DocumentLayoutRoot
DLA_C_GEN_PAGESET	Generic PageSet
DLA_C_GEN_PAGE	Generic Page
DLA_C_GEN_RECTO_PAGE	Generic RectoPage
DLA_C_GEN_VERSO_PAGE	Generic VersoPage
DLA_C_GEN_COMP_HEADER	Generic CompositeHeader
DLA_C_GEN_COMP_FOOTER	Generic CompositeFooter
DLA_C_GEN_COMP_BODY_FIX	Generic FixedCompositeBody
DLA_C_GEN_COMP_BODY_VAR	Generic VariableCompositeBody
DLA_C_GEN_COL_FIXED	Generic ColumnFixed
DLA_C_GEN_COL_VARIABLE	Generic ColumnVariable
DLA_C_GEN_SNAKING_COLUMN	Generic SnakingColumns
DLA_C_GEN_SYNC_COLUMN	Generic SynchronizedColumns
DLA_C_GEN_BASIC_FLOAT	Generic BasicFloat
DLA_C_GEN_COMP_FLOAT	Generic CompositeFloat
DLA_C_GEN_BASIC_COLUMN	Generic BasicColumn
DLA_C_GEN_FOOTNOTE_AREA	Generic FootnoteArea
DLA_C_GEN_ARRNGO_CONT_FIXED	Generic ArrangedContentFixed
DLA_C_GEN_ARRNGO_CONT_VAR	Generic ArrangedContentVariable
DLA_C_GEN_SRCD_CONT_FIXED	Generic SourcedContentFixed
DLA_C_GEN_SRCD_CONT_VAR	Generic SourcedContentVariable
DLA_C_GEN_COMP_FIXTURE_VAR	Generic CompositeFixtureVariable
DLA_C_GEN_COMP_FIXTURE_FIXED	Generic CompositeFixtureFixed
DLA_C_GEN_BASIC_FIXTURE	Generic BasicFixture
DLA_C_GEN_COMP_COL_FIXED	Generic CompositeColumnFixed
DLA_C_GEN_COMP_COL_VARIABLE	Generic CompositeColumnVariable
DLA_C_GEN_COMP_COMMON	Generic CompositeCommon
DLA_C_GEN_COMP_ARTWORK	Generic CompositeArtwork



Entity type	Meaning
DLA_C_GEN_BASIC_HEADER	Generic BasicHeader
DLA_C_GEN_BASIC_FOOTER	Generic BasicFooter
DLA_C_GEN_BASIC_BODY	Generic BasicBody
DLA_C_GEN_GENERIC_BLOCK	Generic Generic GenericBlock
DLA_C_GEN_FORM_AREA	Generic FormArea
DLA_C_GEN_ENTRY_GROUP_AREA	Generic EntryGroupArea
DLA_C_GEN_TABLE_AREA	Generic TableArea
DLA_C_GEN_TABLE_HEADER	Generic TableHeader
DLA_C_GEN_TABLE_LABEL	Generic TableLabel
DLA_C_GEN_COMP_TABLE_LABEL	Generic CompositeTableLabel
DLA_C_GEN_LABEL_COMPONENT	Generic LabelComponent
DLA_C_GEN_ROW_AREA	Generic RowArea
DLA_C_GEN_CELL	Generic Cell
DLA_C_GEN_SUBROW_GROUP	Generic SubRowGroup
DLA_C_GEN_SUBROW	Generic SubRow
DLA_C_GEN_TABLE_LABEL_CONTENT	Generic TableLabelContent
DLA_C_GEN_FORM_ENTRY_AREA	Generic FormEntryArea

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_INV\_ENTITY]

The named entity does not exist.

### [DLA\_E\_GEN\_INV\_ENTITY]

The entity is not a generic entity.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dla\_make\_frame\_equiv****dla\_make\_frame\_equiv**

This function makes two generic frame entities of the same DAP type equivalent for layout so that they can both receive a single stream of flowing text.

**C Synopsis**

```
dla_status      dla_make_frame_equiv(
    dla_document_handle    document_handle,
    dla_entity_handle      first_frame_handle,
    dla_entity_handle      second_frame_handle);
```

**Description**

The `dla_make_frame_equiv` function makes two frames of the same DAP type equivalent for layout so that they can both receive a single stream of flowing text.

The frames must both be either `SnakingColumns`, `SynchronizedColumns`, or lowest level frames that do not have common content via `GenericBlock` or `CommonContent`, that is they must both be of the same type from the following list:

- `SnakingColumns`
- `SynchronizedColumns`
- `BasicColumn`
- `BasicFigure`
- `BasicFloat`
- `Cell`
- `FootnoteArea`
- `FormEntryArea`

One frame is used as a model and is not altered. It may have been used in a layout directive prior to this function call. The other frame, or target frame, is altered to be equivalent. It (or its immediately subordinate lowest-level column frames) must not have been used in a layout directive prior to this function call.

All of the `ColumnVariable` frames in a snaking columns frame (where there is more than one, using `SEQ` rather than `REP`) are already equivalent to each other because the text is intended to flow from one column to the next. Thus individual `ColumnVariable` frames cannot be made equivalent to any other frame. However, a snaking columns frame can be made equivalent to another snaking columns frame. All the `ColumnVariable` frames will then be equivalent.

None of the `ColumnFixed` frames in a synchronized columns frame are allowed to be equivalent to each other because each stream of text is intended to be laid out exclusively in only one column. Thus individual `ColumnFixed` frames cannot be made equivalent to any other frame. However, a synchronized columns frame can be made equivalent to another synchronized columns frame. Both synchronized columns frames must have the same number of subordinate `ColumnFixed` frames. One of each of the pairs of corresponding `ColumnFixed` frames, taken in order, is then altered to be equivalent.

In the case of FOD36 documents, calling this function for snaking or synchronized columns frames modifies only the ColumnFixed or ColumnVariable frames respectively. If there are zero such frames subordinate to the target frame, then the function fails. Any composite column and arranged or sourced content subordinate frames are ignored during the process. For correct layout, the lowest level frames within such composite columns must also (unless they are already linked to ensure equivalence) be rendered equivalent separately by additional calls of `dia_make_frame_equiv`.

Calling this function for footnote frames (only allowed in the case of FOD36 documents) is sensible only if the Permitted categories attribute of the `first_frame_handle` has been modified from the pre-set FOD26 string of footnote by using the property Footnote category stream index (`DLA_C_FTN_CAT_IDX`).

## Arguments

**document\_handle**

The handle of the document being operated on.

**first\_frame\_handle**

The handle of the frame to be used as a model.

**second\_frame\_handle**

The handle of the frame that is to be altered. It (and its subordinate frames if any) must not have been used in a layout directive prior to this function call.

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_OP\_AT\_DAP\_LEVEL]**

This operation is not relevant for DAP level 1 documents.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The type or types of the frames are not valid for this function.

**[DLA\_E\_NUM\_COLS\_MISMATCH]**

In the case of synchronized columns frames both must have the same number of subordinate ColumnFixed frames.

**[DLA\_E\_FRAMES\_NOT\_SAME\_TYPE]**

The types of the frames given must be identical for this function.

## `dia_make_frame_equiv`

### `[DLA_E_ALL_COLS_COMPOSITE]`

In the FOCUS synchronized or snaking columns, none of the column frames are lowest level frames that can be altered.

### `[DLA_E_TARGET_IN_LAY_DIR]`

The frame that is to be altered (or one or more of its immediately subordinate lowest level frames if any) has been used in a layout directive prior to this function call.

### `[DLA_E_REQ_ARGS_NOTSPEC]`

Not all arguments were correctly specified.

### `[DLA_E_MEM_FAIL]`

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_move\_to\_child\_spec\_entity**

Returns the handle of a child of a specific logical or layout entity.

**C Synopsis**

```
dia_status      dia_move_to_child_spec_entity(
    dia_document_handle    document_handle,
    dia_entity_handle      current_entity,
    dia_child_type         child_type,
    dia_entity_handle      *child_entity);
```

**Description**

The `dia_move_to_child_spec_entity` function returns the handle of a child of the specified entity. The application must specify whether the first or last child is required.

**Arguments****document\_handle**

The handle of the document being operated on.

**current\_entity**

The handle of specific logical or layout entity for which the child handle is required.

**child\_type**

The child entity to move to. An enumerated value that is one of:

Child type	Meaning
DLA_C_CHILD_FIRST	Move to the first child.
DLA_C_CHILD_LAST	Move to the last child.

**child\_entity [write]**

The handle of the child of the current specific logical or layout entity.



## **dla\_move\_to\_child\_spec\_entity**

### **Results**

#### **Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### **Errors**

#### **[DLA\_E\_ENTITY\_HAS\_NO\_CHILD]**

The entity has no child.

#### **[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

#### **[DLA\_E\_ENTITY\_TYPE\_NOT\_SPEC\_LOG]**

The entity is not a specific logical entity (PDA document).

#### **[DLA\_E\_ENTITY\_TYPE\_NOT\_SPEC\_LAY]**

The entity is not a specific layout entity (FDA document).

#### **[DLA\_E\_ENTITY\_TYPE\_NOT\_SPEC]**

The entity is not a specific logical or layout entity (FPDA document).

#### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

#### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_move\_to\_parent\_spec\_entity**

Returns the handle of the parent of a specific logical or layout entity.

**C Synopsis**

```
dia_status      dia_move_to_parent_spec_entity(
    dia_document_handle    document_handle,
    dia_entity_handle      current_entity,
    dia_entity_handle      *parent_entity);
```

**Description**

The `dia_move_to_parent_entity` function returns the handle of the parent of the specified entity.

**Argument**

**document\_handle**

The handle of the document being operated on.

**current\_entity**

The handle of specific logical or layout entity for which the parent handle is required.

**parent\_entity [write]**

The handle of the parent of the current specific logical or layout entity.

**Results**

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned.

**Errors**

**[DLA\_E\_AT\_ROOT]**

The specific logical or layout root has no parent.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_ENTITY\_TYPE\_NOT\_SPEC\_LOG]**

The entity is not a specific logical entity (PDA document).

**dla\_move\_to\_parent\_spec\_entity**

**[DLA\_E\_ENTITY\_TYPE\_NOT\_SPEC\_LAY]**

The entity is not a specific layout entity (FDA document).

**[DLA\_E\_ENTITY\_TYPE\_NOT\_SPEC]**

The entity is not a specific logical or layout entity (FPDA document).

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_move\_to\_root\_entity**

Returns the handle of the root entity of the specific logical, specific layout, generic logical or generic layout structure.

**C Synopsis**

```
dia_status      dia_move_to_root_entity(
    dia_document_handle    document_handle,
    dia_structure_type     structure_type,
    dia_entity_handle      *root_entity);
```

**Description**

The `dia_move_to_root_entity` function returns the handle of the root entity of the specified structure. The application must specify whether the handle of the root of the specific logical, specific layout, generic logical or generic layout structure is required.

If an external generic document is in use, this function can be used to return the handle of either of the generic roots, and the handle of either the specific or generic document can be given as a `document_handle`.

If an external generic document is to be used but has not yet been read, this function cannot be used to access any root entity until the external generic document has been read.

**Arguments****document\_handle**

The handle of the document being operated on.

**structure\_type**

The type of structure whose root entity is required. An enumerated value that is one of the following:

Structure type	Meaning
DLA_C_GENERIC_LAYOUT	Generic layout root entity
DLA_C_GENERIC_LOGICAL	Generic logical root entity
DLA_C_SPECIFIC_LAYOUT	Specific layout root entity
DLA_C_SPECIFIC_LOGICAL	Specific logical root entity

**root\_entity [write]**

The handle of the root of the specified structure.

## **dia\_move\_to\_root\_entity**

### **Results**

#### **Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### **Errors**

#### **[DLA\_E\_MISSING\_ROOT]**

The document does not have a root of the given type.

#### **[DLA\_E\_MISSING\_GEN\_STRUCT]**

The document does not have necessary generic structures and/or indicates a requirement for an external generic document (property DLA\_C\_EXT\_DOC) that has not been read. This prevents access to specific roots as well.

#### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

#### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_move\_to\_sibling\_spec\_entity

Returns the handle of a sibling entity.

## C Synopsis

```
dia_status      dia_move_to_sibling_spec_entity(
    dia_document_handle    document_handle,
    dia_entity_handle      current_entity,
    dia_sibling_type       sibling_type,
    dia_entity_handle      *sibling_entity);
```

## Description

The `dia_move_to_sibling_spec_entity` function returns the handle of the sibling specific logical or layout entity of the specified type. The application can request the first, last, previous or next sibling.

## Arguments

**document\_handle**

The handle of the document being operated on.

**current\_entity**

The handle of specific logical or layout entity for which the sibling handle is required.

**sibling\_type**

The type of sibling entity whose handle is required. An enumerated value which is one of the following:

Sibling type	Meaning
DLA_C_SIB_FIRST	Handle of the first sibling
DLA_C_SIB_LAST	Handle of the last sibling
DLA_C_SIB_PREV	Handle of the previous sibling
DLA_C_SIB_NEXT	Handle of the next sibling

**sibling\_entity [write]**

The handle of the sibling to the current specific logical or layout entity. If none exists, it receives a value of `DLA_C_NULL_HANDLE`.

## **dla\_move\_to\_sibling\_spec\_entity**

### **Results**

#### **Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### **Errors**

#### **[DLA\_E\_ENTITY\_ONLY\_CHILD]**

The entity has no sibling.

#### **[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

#### **[DLA\_E\_ENTITY\_TYPE\_NOT\_SPEC\_LOG]**

The entity is not a specific logical entity (PDA document).

#### **[DLA\_E\_ENTITY\_TYPE\_NOT\_SPEC\_LAY]**

The entity is not a specific layout entity (FDA document).

#### **[DLA\_E\_ENTITY\_TYPE\_NOT\_SPEC]**

The entity is not a specific logical or layout entity (FPDA document).

#### **[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

#### **[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## dia\_read\_document

Reads an ODIF data stream and creates the corresponding ODA document.

## C Synopsis

```

dia_status      dia_read_document (
    dia_toolkit_handle    toolkit_handle,
    char                  *file_specification,
    dia_length            file_specification_length,
    dia_document_options  document_options,
    char                  *store_file,
    dia_length            store_file_length,
    dia_dap_level         *dap_level,
    dia_document_handle   *document_handle);

```

## Description

The `dia_read_document` function reads an ODIF data stream from the specified file and creates the corresponding ODA document. The DAP level of the document is returned.

If required by the implementation, the `store_file` argument must be specified. This is a file specification used by the toolkit to create temporary files. These files are used by the memory management scheme of the toolkit.

The `document_options` argument is not used.

## Arguments

**toolkit\_handle**

The handle of the toolkit instance.

**file\_specification**

A string containing the file specification of the file containing the ODIF data stream.

**file\_specification\_length**

The length in octets of the string containing the file specification of the file containing the ODIF data stream.

**document\_options**

Application-specified options used to control the processing of the document. This argument is not used; specify `DLA_C_NO_OPTIONS`.

**store\_file**

A string containing the file specification for the temporary files used by the memory management scheme. If this argument is not required by the toolkit implementation, `DLA_C_NULL_STRING` must be specified.

**dla\_read\_document****store\_file\_length**

The length in octets of the string containing the file specification for the temporary files used by the memory management scheme.

**dap\_level [write]**

The DAP level of the document.

DAP level	Meaning
DLA_C_DAP_LEVEL_1	Document at DAP level 1
DLA_C_DAP_LEVEL_2	Document at DAP level 2
DLA_C_DAP_LEVEL_3	Document at DAP level 3

**document\_handle [write]**

The handle of the newly created document.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_INV\_TOOLKIT]**

The toolkit handle is not valid.

**[DLA\_E\_INV\_ODIF]**

An error was found in the ODIF data stream.

**[DLA\_E\_DAP\_LEVEL]**

The DAP level specified is not supported.

**[DLA\_E\_NO\_DAP\_LEVEL]**

An error occurred determining DAP level of document.

**[DLA\_E\_ODA\_VERSION]**

The ODA Version of the document is not supported by the DAP API.

**[DLA\_E\_MISSING\_STRUCT]**

An FPDA or PDA document may not have only one generic structure.

**[DLA\_E\_GEN\_LAY\_STRUCT]**

The generic layout structure is not declared to be a complete generator set.

**[DLA\_E\_WRONG\_VALUE]**

The value of an essential interpreted property could not be derived. This applies in particular to the constraint name that is interpreted as the property DLA\_C\_ENT\_TYP and is necessary to the correct interpretation of many other properties.

**[DLA\_E\_PERM\_CAT\_TABLE]**

An error occurred while tabulating the permitted categories.

**[DLA\_E\_FILE\_ERROR]**

An error occurred during a file operation.

**[DLA\_E\_CALLBACK\_ERROR]**

Indicates that the callback function returned DLA\_C\_STOP. The read operation was terminated immediately the error was detected and so the document in store may not be complete. The application should not attempt to continue processing of this document.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dla\_read\_generic\_doc****dla\_read\_generic\_doc**

Reads an ODIF data stream that represents an external document class or resource document needed by a pre-existing (created or read) ODA document.

**C Synopsis**

```
dla_status      dla_read_generic_doc(
    dla_document_handle document_handle,
    char           *file_specification,
    dla_length     file_specification_length,
    dla_generic_doc_type generic_doc_type,
    dla_document_handle *generic_doc_handle);
```

**Description**

The `dla_read_generic_doc` function reads an ODIF data stream from the specified file and creates the corresponding generic document. The DAP level of the generic document must match that of the pre-existing document.

If the pre-existing document has been created, then the mode in which it has been created constrains the use of the current function. If the mode is `mode-1`, `DLA_C_FOR_EXTERNAL` or `DLA_C_FOR_RESOURCE`, then this function cannot be used. If the mode is `mode-2`, then this function can only be used to read a resource document. This function can only be used to read an external document class if the writing mode is `DLA_C_WITH_EXTERNAL`.

If the pre-existing document has been read from ODIF, then the use of the current function is constrained only by the checks that are performed by the underlying ODA API.

**Arguments****document\_handle**

The handle of the pre-existing document instance.

**file\_specification**

A string containing the file specification of the file containing the ODIF data stream.

**file\_specification\_length**

The length in octets of the string containing the file specification of the file containing the ODIF data stream.

**generic\_doc\_type**

Indicates which role the generic document is to fulfil, one of the following:

Type	Meaning
DLA_C_EXTERNAL_DOC	External document
DLA_C_RESOURCE_DOC	Resource document

**generic\_doc\_handle [write]**

The handle of the newly read document.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_EXIST\_GEN\_DOC]**

The pre-existing document already has a generic document.

**[DLA\_E\_INV\_MODE]**

The writing mode of the pre-existing document is not appropriate for this function.

**[DLA\_E\_INV\_ODIF]**

An error was found in the ODIF data stream.

**[DLA\_E\_DAP\_LEVEL]**

The DAP level of the pre-existing document is not appropriate for this function, or the DAP levels of the two documents do not match.

**[DLA\_E\_NO\_DAP\_LEVEL]**

An error occurred determining DAP level of the generic document.

**[DLA\_E\_ODA\_VERSION]**

The ODA Version of the document is not supported by the DAP API.

**[DLA\_E\_MISSING\_STRUCT]**

An FPDA or PDA document may not have only one generic structure.

**[DLA\_E\_GEN\_LAY\_STRUCT]**

The generic layout structure is not declared to be a complete generator set (applies only to an external document).

**[DLA\_E\_GEN\_LOG\_STRUCT]**

The generic logical structure is not declared to be a complete generator set (applies only to an external document).

**[DLA\_E\_GEN\_STRUCTS]**

Neither of the generic structures is declared to be a complete generator set (applies only to an external document).

## dla\_read\_generic\_doc

### [DLA\_E\_WRONG\_VALUE]

The value of an essential interpreted property could not be derived. This applies in particular to the constraint name that is interpreted as the property DLA\_C\_ENT\_TYP and is necessary to the correct interpretation of many other properties.

### [DLA\_E\_PERM\_CAT\_TABLE]

An error occurred while tabulating the permitted categories (applies only to an external document).

### [DLA\_E\_FILE\_ERROR]

An error occurred during a file operation.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## dla\_read\_writing\_mode

Reads the mode in which a specific document is being written.

## C Synopsis

```
dla_status      dla_read_writing_mode(
    dla_document_handle      document_handle,
    dla_document_options     *document_options);
```

## Description

The dla\_read\_writing\_mode function reads the mode in which a specific document is being written.

## Arguments

**document\_handle**

The handle of the document being operated on.

**document\_options [write]**

An enumerated value by which the application has specified an option to control the processing of the document. The first three codes listed have the same numeric value as well as the same semantic and so cannot be distinguished.

Option	Meaning
DLA_C_NO_OPTIONS	Equivalent to DLA_C_MODE_1
DLA_C_MODE_1	Toolkit creates generic logical structure
DLA_C_AUTO_GEN	Toolkit creates generic logical structure (Equivalent to DLA_C_MODE_1)
DLA_C_MODE_2	Application creates generic logical structure
DLA_C_WITH_EXTERNAL	Application will read generic document from ODIF
DLA_C_FOR_EXTERNAL	Application will supply ODIF identifiers for all entities
DLA_C_FOR_RESOURCE	Application will supply ODIF identifiers for all entities and the resources property

`dia_read_writing_mode`

## Results

### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

### [DLA\_E\_DOC\_READ\_ONLY]

The document is read only.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dia\_register\_callback\_function

Specifies a function in the application's code space that is invoked periodically during `dia_read_document` and `dia_write_document`.

### C Synopsis

```
dia_status      dia_register_callback_function(  
    dia_toolkit_handle    toolkit_handle,  
    dia_callback_function callback_function);
```

### Description

The `dia_register_callback_function` function informs the DAP API about an application-defined function that the DAP API can call periodically during operations that may take an appreciable length of time to execute. DAP API functions that invoke this application callback function are `dia_read_document` and `dia_write_document`.

The value specified by the `callback_function` argument must be either a valid function pointer or NULL. NULL specifies that any previously registered callback function within the specified instance of the DAP API toolkit is to be un-registered.

See 2.1.5.11 for the specification of the application callback function.

Calling DAP API functions from within the application-supplied callback function is neither supported nor recommended. Because this function is only called within the context of `dia_write_document` or `dia_read_document`, invoking other DAP API functions within the callback function may have unpredictable and undesirable effects.

### Arguments

#### **toolkit\_handle**

The handle of the toolkit being operated on.

#### **callback\_function**

The address of the application-supplied function that is to be used as a callback function, or NULL if no callback function is required or if a previously-registered callback function is to be un-registered. The prototype of the callback function supplied by this parameter must match the specification given in 2.1.5.11.

## dla\_register\_callback\_function

### Results

#### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

#### [DLA\_E\_INV\_TOOLKIT]

The toolkit handle is not valid.

**dia\_set\_entity\_handle**

Specifies a value for a property on an entity, for a property of entity handle type.

**C Synopsis**

```

dia_status      dia_set_entity_handle(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_entity_handle      property_entity_handle);

```

**Description**

The `dia_set_entity_handle` function specifies the value for a property of type entity. The value of the property of the specified entity is replaced and, on completion, the status of the property is set to specified.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**property\_entity\_handle**

The handle of the entity being specified as a property.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## `dla_set_entity_handle`

### Errors

#### [DLA\_E\_PROP\_FIXED]

The properties of the specified entity have been fixed.

#### [DLA\_E\_SHARED\_PROP\_FIXED]

This property is shared with another entity that has been fixed.

#### [DLA\_E\_DOC\_READ\_ONLY]

The document is read only.

#### [DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

#### [DLA\_E\_INV\_ENTITY\_TYPE]

The specified entity handle is not valid for the specified property.

#### [DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

#### [DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

#### [DLA\_E\_INV\_PROP\_HANDLE]

The entity indicated by property\_entity\_handle is invalid for this property.

#### [DLA\_E\_INV\_CODE]

The property code does not apply to the entity.

#### [DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

#### [DLA\_E\_INV\_STYLE]

The entity cannot refer to the given style because its existing usage is incompatible.

#### [DLA\_E\_PRE\_EXIST\_STYLE]

The entity already refers to a style; overwriting is not allowed.

#### [DLA\_E\_GEN\_SPEC\_STYLE]

The same style cannot be referred to by both generic logical and specific logical entities.

#### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

#### [DLA\_E\_INV\_ENTITY\_MODE]

The writing mode of this document is not appropriate for this function on the given entity.

#### [DLA\_E\_MISS\_PERM\_CAT]

The given frame entity does not have a Permitted categories attribute and so cannot be used with the given layout directive property.

#### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

#### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dla\_set\_index\_entity**

Inserts or replaces an entity handle at an indexed position in the sequence of entity type values of a property specified for an entity.

**C Synopsis**

```
dla_status      dla_set_index_entity(
    dla_document_handle    document_handle,
    dla_entity_handle      entity_handle,
    dla_structure_handle    structure_handle,
    dla_property_code       property_code,
    dla_integer             index,
    dla_index_operation     index_operation,
    dla_entity_handle       property_entity_handle);
```

**Description**

The `dla_set_index_entity` function inserts or replaces an entity handle at an indexed position in the sequence of values of a property specified for an entity. The property must be of sequence of entity type.

For a replace operation the index must be in the range: 1 to the number of entries in the sequence.

For an insert operation the index must be in the range: 1 to the number of entries in the sequence plus 1. A new value is inserted at the specified index position.

On completion, the property status is set to specified.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index of the entry at which the value is to be stored. The first entry in a sequence is at index 1.

**dla\_set\_index\_entity****index\_operation**

Indicates whether the entry must be inserted before, or replace the entry at the indexed position. One of the enumerated values:

Index operation	Meaning
DLA_C_INDEX_REPLACE	Replace the indexed entry.
DLA_C_INDEX_INSERT	Insert before the indexed entry.

**property\_entity\_handle**

The entity handle being specified for the property.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_PROP\_FIXED]**

The properties of the specified entity have been fixed.

**[DLA\_E\_SHARED\_PROP\_FIXED]**

This property is shared with another entity that has been fixed.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_ENTITY\_TYPE]**

The specified entity handle is not valid for the specified property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_INV\_PROP\_HANDLE]**

The entity indicated by property\_entity\_handle is invalid for this property.

**[DLA\_E\_INV\_CODE]**

The property code does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

[DLA\_E\_INV\_ENTITY\_MODE]

The writing mode of this document is not appropriate for this function on the given entity.

[DLA\_E\_INDEX\_OUT\_OF\_RANGE]

The index is out of the range of valid indexes for the entries in the sequence.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_set\_index\_integer****dia\_set\_index\_integer**

Inserts or replaces an integer at an indexed position in the sequence of integer type values, of a property specified for an entity.

**C Synopsis**

```

dia_status      dia_set_index_integer(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code      property_code,
    dia_integer            index,
    dia_index_operation     index_operation,
    dia_integer            integer_value);

```

**Description**

The `dia_set_index_integer` function inserts or replaces an integer at an indexed position in the sequence of values of a property specified for an entity. The property must be of sequence of integer type.

For a replace operation the index must be in the range: 1 to the number of entries in the sequence.

For an insert operation the index must be in the range: 1 to the number of entries in the sequence plus 1. A new value is inserted at the specified index position.

On completion, the property status is set to specified.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index of the entry at which the value is to be stored. The first entry in a sequence is at index 1.

## dia\_set\_index\_integer

**index\_operation**

Indicates whether the entry must be inserted before, or replace, the entry existing at the indexed position. One of the enumerated values:

Index operation	Meaning
DLA_C_INDEX_REPLACE	Replace the indexed entry.
DLA_C_INDEX_INSERT	Insert before the indexed entry.

**integer\_value**

The integer value being specified for the property.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_PROP\_FIXED]**

The properties of the specified entity have been fixed.

**[DLA\_E\_SHARED\_PROP\_FIXED]**

This property is shared with another entity that has been fixed.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INDEX\_OUT\_OF\_RNG]**

The index is out of the range of valid indexes for entries in the sequence.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_VAL\_OUT\_OF\_RNG]**

The integer value specified is not within the permitted range of values for this property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_INV\_CODE]**

The property code does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

## `dla_set_index_integer`

### `[DLA_E_INV_ENTITY_MODE]`

The writing mode of this document is not appropriate for this function on the given entity.

### `[DLA_E_REQ_ARGS_NOTSPEC]`

Not all arguments were correctly specified.

### `[DLA_E_MEM_FAIL]`

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_set\_index\_null\_value**

Specifies the value for an indexed property of type null-valued.

**C Synopsis**

```
dia_status      dia_set_index_null_value(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code       property_code,
    dia_integer            index,
    dia_index_operation     index_operation);
```

**Description**

The `dia_set_index_null_value` function specifies the value for an indexed property of type null-valued.

For a replace operation, the index must be in the range: 1 to the number of entries in the sequence. For an insert operation, the index must be in the range: 1 to the number of entries in the sequence plus 1. A new value is inserted at the specified index position. On completion, the property status is set to specified.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index of the entry at which the value is to be altered. The first entry in a sequence is at index 1.

**dla\_set\_index\_null\_value****index\_operation**

Indicates whether the entry must be inserted before, or replace, the entry existing at the indexed position. One of the enumerated values:

Index operation	Meaning
DLA_C_INDEX_REPLACE	Replace the indexed entry.
DLA_C_INDEX_INSERT	Insert before the indexed entry.

**Results****Status**

Normally success, indicating that the operation was successful, otherwise one of the following errors is returned:

**Errors****[DLA\_E\_PROP\_FIXED]**

The properties of the specified entity have been fixed.

**[DLA\_E\_SHARED\_PROP\_FIXED]**

This property is shared with another entity that has been fixed.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property code does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_INDEX\_OUT\_OF\_RANGE]**

The index is out of the range of valid indexes for entries in the sequence.

dla\_set\_index\_null\_value

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dla\_set\_index\_string****dla\_set\_index\_string**

Specifies the value for an indexed property of type string.

**C Synopsis**

```
dla_status      dla_set_index_string(
    dla_document_handle    document_handle,
    dla_entity_handle      entity_handle,
    dla_structure_handle    structure_handle,
    dla_property_code       property_code,
    dla_integer             index,
    dla_index_operation      index_operation,
    dla_octet_string        string_value,
    dla_length              string_value_length);
```

**Description**

The `dla_set_index_string` function specifies the value for an indexed property of type string.

For a replace operation, the index must be in the range: 1 to the number of entries in the sequence. For an insert operation, the index must be in the range: 1 to the number of entries in the sequence plus 1. A new value is inserted at the specified index position. On completion, the property status is set to specified.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**index**

The index of the entry at which the value is to be stored. The first entry in a sequence is at index 1.

**index\_operation**

Indicates whether the entry should be inserted before, or replace, the entry existing at the indexed position. One of the enumerated values:

index_operation	Meaning
DLA_C_INDEX_REPLACE	Replace the indexed entry.
DLA_C_INDEX_INSERT	Insert before the indexed entry.

**string\_value**

The octet string being specified for the property.

**string\_value\_length**

The length of the octet string expressed as the number of octets.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_PROP\_FIXED]**

The properties of the specified entity have been fixed.

**[DLA\_E\_SHARED\_PROP\_FIXED]**

This property is shared with another entity that has been fixed.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property code does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

## `dla_set_index_string`

### `[DLA_E_INDEX_OUT_OF_RNG]`

The index is out of the range of valid indexes for entries in the sequence.

### `[DLA_E_REQ_ARGS_NOTSPEC]`

Not all arguments were correctly specified.

### `[DLA_E_MEM_FAIL]`

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dla\_set\_index\_structure**

Creates a structure and inserts or updates the structure handle at an indexed position in the sequence of values of a property specified for an entity, for a property of sequence of structure type.

**C Synopsis**

```
dla_status      dla_set_index_structure(
    dla_document_handle    document_handle,
    dla_entity_handle      entity_handle,
    dla_structure_handle    structure_handle,
    dla_property_code       property_code,
    dla_integer             index,
    dla_index_operation     index_operation,
    dla_structure_handle    *new_structure_handle);
```

**Description**

The `dla_set_index_structure` function creates a structure and inserts or replaces the structure handle in the sequence of values of a property specified for an entity. The property must be of sequence of structure type.

For a replace operation the index must be in the range: 1 to the number of entries in the sequence.

For an insert operation the index must be in the range: 1 to the number of entries in the sequence plus 1.

On completion, the property status is set to specified.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**dla\_set\_index\_structure****index**

The index of the entry at which the value is to be stored. The first entry in a sequence is at index 1.

**index\_operation**

Indicates whether the entry should be inserted before, or update the entry at the indexed position. One of the enumerated values:

Index operation	Meaning
DLA_C_INDEX_REPLACE	Replace the indexed entry.
DLA_C_INDEX_INSERT	Insert before the indexed entry.

**new\_structure\_handle [write]**

Receives the structure handle inserted or replaced in the sequence of values specified for the property.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_PROP\_FIXED]**

The properties of the specified entity have been fixed.

**[DLA\_E\_SHARED\_PROP\_FIXED]**

This property is shared with another entity that has been fixed.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_CODE]**

The property code does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**dla\_set\_index\_structure**

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

**[DLA\_E\_INDEX\_OUT\_OF\_RNG]**

The index is out of range for valid indexes for the entries in the sequence.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_set\_integer****dia\_set\_integer**

Specifies the value for a property of type integer.

**C Synopsis**

```
dia_status      dia_set_integer(
    dia_document_handle document_handle,
    dia_entity_handle  entity_handle,
    dia_structure_handle structure_handle,
    dia_property_code   property_code,
    dia_integer         integer_value);
```

**Description**

The `dia_set_integer` function specifies the value for a property of type integer. The value of the property of the specified entity is replaced and the property status is set to specified.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**integer\_value**

The integer value being specified for the property.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_PROP\_FIXED]**

The properties of the specified entity have been fixed.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_SHARED\_PROP\_FIXED]**

This property is shared with another entity that has been fixed.

**[DLA\_E\_PAGE\_HAS\_CHILD]**

Page layout type cannot be set on a generic page that has header, footer or body frames.

**[DLA\_E\_RECTO\_VERSO\_HAS\_CHILD]**

Page layout type cannot be set on either generic page of recto verso pair when the other has any of header, footer or body frames.

**[DLA\_E\_SEQ\_LEVEL\_NUM\_EXISTS]**

The specified level of segment numbering property or index of general numbering property (which are equivalent for this comparison) already exists on this entity; duplicates are not permitted.

**[DLA\_E\_VAL\_OUT\_OF\_RNG]**

The integer value specified is not within the permitted range of values for this property.

**[DLA\_E\_INV\_PROP\_OPER]**

The requested operation is not valid for this property.

**[DLA\_E\_INV\_CODE]**

The property code does not apply to the entity.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

**[DLA\_E\_TARGET\_IN\_LAY\_DIR]**

The footnote frame that is to be altered (property DLA\_C\_FTN\_CAT\_IDX) has been used in a layout directive prior to this function call.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## **dia\_set\_null\_value**

## **dia\_set\_null\_value**

Specifies the value for a property of type null-valued.

### **C Synopsis**

```
dia_status      dia_set_null_value(  
    dia_document_handle    document_handle,  
    dia_entity_handle      entity_handle,  
    dia_structure_handle   structure_handle,  
    dia_property_code      property_code);
```

### **Description**

The `dia_set_null_value` function specifies the value for a property of type null-valued. The value of the property of the specified entity is replaced and the property status is set to specified.

### **Arguments**

#### **document\_handle**

The handle of the document being operated on.

#### **entity\_handle**

The handle of the entity being operated on.

#### **structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

#### **property\_code**

A symbolic constant representing the property being operated on.

### **Results**

#### **Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:



## Errors

### [DLA\_E\_PROP\_FIXED]

The properties of the specified entity have been fixed.

### [DLA\_E\_SHARED\_PROP\_FIXED]

This property is shared with another entity that has been fixed.

### [DLA\_E\_DOC\_READ\_ONLY]

The document is read only.

### [DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

### [DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

### [DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

### [DLA\_E\_INV\_CODE]

The property code does not apply to the entity.

### [DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

### [DLA\_E\_INV\_ENTITY\_MODE]

The writing mode of this document is not appropriate for this function on the given entity.

### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dia\_set\_status**

## **dia\_set\_status**

Sets the status of a property of an entity.

### **C Synopsis**

```
dia_status      dia_set_status(  
    dia_document_handle    document_handle,  
    dia_entity_handle      entity_handle,  
    dia_structure_handle   structure_handle,  
    dia_property_code      property_code,  
    dia_property_status    property_status);
```

### **Description**

The `dia_set_status` function sets the status of a property of an entity of a document. Only the following property codes can be set to a value of `DLA_C_PROP_NULL`:

`DLA_C_BDR_LEFT`  
`DLA_C_BDR_RGT`  
`DLA_C_BDR_LDG`  
`DLA_C_BDR_TRL`  
`DLA_C_BAL`  
`DLA_C_PTH_DMS_RUL_A`  
`DLA_C_PTH_DMS_RUL_B`  
`DLA_C_OGL_DMS_RUL_B`  
`DLA_C_FTL_REF_PROPERTIES`

Any property that can be set using one of the other `dia_set` functions can be set to a value of `DLA_C_PROP_UNSPECIFIED`, except for: elements of a `SEQUENCE` property, properties corresponding to attributes that cannot be so set using the ODA API, and the following property codes:

`DLA_C_CON_ARC_CLS` (for raster and geometric entities)  
`DLA_C_CP_TY_COD` (for character and geometric entities)  
`DLA_C_PG_LAY_TYPE`  
`DLA_C_LAY_STYLE`  
`DLA_C_PRES_STYLE`

### **Arguments**

**document\_handle**  
The handle of the document being operated on.

**entity\_handle**  
The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of DLA\_C\_NULL\_HANDLE must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**property\_status**

The status specified for the property.

Status	Meaning
DLA_C_PROP_NULL	The property status is null.
DLA_C_PROP_UNSPECIFIED	The property status is unspecified.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_PROP\_FIXED]**

The properties of the specified entity have been fixed.

**[DLA\_E\_NOT\_A\_PROPERTY\_CODE]**

The property code argument does not represent any property.

**[DLA\_E\_INV\_CODE]**

The property does not apply to the entity.

**[DLA\_E\_PROP\_NOT\_IN\_DAP]**

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

**[DLA\_E\_INV\_PROP\_OPER]**

Status cannot be set for this property.

**[DLA\_E\_WRONG\_STRUCTURE]**

The given structure does not exist for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

## `dia_set_status`

### `[DLA_E_REQ_ARGS_NOTSPEC]`

Not all arguments were correctly specified.

### `[DLA_E_MEM_FAIL]`

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_set\_string**

Specifies the value for a property of type string.

**C Synopsis**

```

dia_status      dia_set_string(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_structure_handle    structure_handle,
    dia_property_code      property_code,
    dia_octet_string        string_value,
    dia_length              string_value_length);

```

**Description**

The `dia_set_string` function specifies the value for a property of type string. The value of the property of the specified entity is replaced and the property status set to specified.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**structure\_handle**

The handle of the structure being operated on. This argument is used when operating on a property in a structure. If the argument is not used, a value of `DLA_C_NULL_HANDLE` must be specified.

**property\_code**

A symbolic constant representing the property being operated on.

**string\_value**

The octet string being specified for the property.

**string\_value\_length**

The length of the octet string expressed as the number of octets.

## `dla_set_string`

### Results

#### Status

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### Errors

#### [DLA\_E\_PROP\_FIXED]

The properties of the specified entity have been fixed.

#### [DLA\_E\_SHARED\_PROP\_FIXED]

This property is shared with another entity that has been fixed.

#### [DLA\_E\_DOC\_READ\_ONLY]

The document is read only.

#### [DLA\_E\_NOT\_A\_PROPERTY\_CODE]

The property code argument does not represent any property.

#### [DLA\_E\_INV\_PROP\_OPER]

The requested operation is not valid for this property.

#### [DLA\_E\_PROP\_NOT\_IN\_DAP]

The property code is not valid for the entity type of the entity\_handle at the DAP level of the document, but would be valid if the DAP level of the document were different.

#### [DLA\_E\_INV\_CODE]

The property code does not apply to the entity.

#### [DLA\_E\_WRONG\_STRUCTURE]

The given structure does not exist for this entity.

#### [DLA\_E\_INV\_ENTITY]

The entity does not exist.

#### [DLA\_E\_INV\_ENTITY\_MODE]

The writing mode of this document is not appropriate for this function on the given entity.

#### [DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

#### [DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



**dia\_set\_tab\_stop**

Sets a tabulation stop at the specified position in the line layout table associated with the specified entity.

**C Synopsis**

```

dia_status      dia_set_tab_stop(
    dia_document_handle    document_handle,
    dia_entity_handle      entity_handle,
    dia_octet_string       tab_reference,
    dia_length              tab_reference_length,
    dia_integer             tab_position,
    dia_tab_alignment       alignment,
    dia_octet_string        alignment_string,
    dia_length              alignment_string_length);

```

**Description**

The `dia_set_tab_stop` function forms a tabulation stop with the specified `tab_reference`, `tab_position`, `alignment` and `alignment_string`. The tabulation stop is added to the line layout table associated with the specified entity.

The entity must be basic and with character content architecture, or it must be the profile.

**Arguments****document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

**tab\_reference**

An octet string representing the required tab reference.

**tab\_reference\_length**

The length of the tab reference octet string, expressed as the number of octets.

**tab\_position**

The integer value representing the tabulation position in EMUs.

**dla\_set\_tab\_stop****alignment**

The required alignment type. An enumerated value that is one of:

Alignment type	Meaning
DLA_C_TAB_ALIGN_START	Start aligned
DLA_C_TAB_ALIGN_END	End aligned
DLA_C_TAB_ALIGN_CENTRED	Center aligned
DLA_C_TAB_ALIGN_AROUND	Align around

**alignment\_string**

The octet string representing the alignment string. This must be DLA\_C\_NULL\_STRING if no alignment string is specified.

**alignment\_string\_length**

The integer value representing the alignment string length, expressed as the number of octets.

**Results****Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors****[DLA\_E\_PROP\_FIXED]**

The properties of the specified entity have been fixed.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

**[DLA\_E\_INV\_ENTITY\_OPER]**

The requested operation is not valid for this entity.

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_INV\_ENTITY\_MODE]**

The writing mode of this document is not appropriate for this function on the given entity.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

**dia\_term\_toolkit**

Terminates an instance of the DAP Level API toolkit.

**C Synopsis**

```
dia_status      dia_term_toolkit(
    dia_toolkit_handle toolkit_handle);
```

**Description**

The `dia_term_toolkit` function terminates an application's use of an instance of the DAP API toolkit. System resources allocated to the toolkit instance are released. Document instances occurring within the toolkit instance are terminated. This function must be called following the completion of document processing with the specified instance of the toolkit.

**Arguments**

**toolkit\_handle**  
The handle of the toolkit instance.

**Results**

**Status**  
Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

**Errors**

**[DLA\_E\_INV\_TOOLKIT]**  
The toolkit handle is not valid.

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**  
Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**  
An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## **dla\_unfix\_properties**

## **dla\_unfix\_properties**

Declares that the setting of properties for an entity can resume.

### **C Synopsis**

```
dla_status      dla_unfix_properties(  
    dla_document_handle    document_handle,  
    dla_entity_handle      entity_handle);
```

### **Description**

The **dla\_unfix\_properties** function declares that the setting of properties for an entity is not complete and can resume.

In the case of the principal entity of a cluster of entities, all members of the cluster are also unfixed. The member entities can also be unfixed separately. Note that the FootnoteText entities are not members of the Footnote cluster.

If any entity is already unfixed, or not yet fixed, performing further unfix functions on that entity has no effect and is not an error.

### **Arguments**

**document\_handle**

The handle of the document being operated on.

**entity\_handle**

The handle of the entity being operated on.

### **Results**

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

### **Errors**

**[DLA\_E\_INV\_ENTITY]**

The entity does not exist.

**[DLA\_E\_DOC\_READ\_ONLY]**

The document is read only.

[DLA\_E\_REQ\_ARGS\_NOTSPEC]

Not all arguments were correctly specified.

[DLA\_E\_MEM\_FAIL]

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

## dla\_version

## dla\_version

Returns the version of the DAP Level API toolkit.

## C Synopsis

```
dla_status      dla_version(  
    char          **version);
```

## Description

The `dla_version` function returns the pointer to a character string that contains the version of the DAP Level API Toolkit.

The format is a zero-terminated string in the form `version;ODAC;release`. For example, `1.0;ODAC;2` represents Release 2.0 of Version 1.0.

## Arguments

**version [write]**

Receives a pointer to a null-terminated character string containing the version.

## Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## Errors

**[DLA\_E\_REQ\_ARGS\_NOTSPEC]**

Not all arguments were correctly specified.

**[DLA\_E\_MEM\_FAIL]**

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.



## dia\_write\_document

Writes a document as an ODIF data stream.

### C Synopsis

```
dia_status      dia_write_document (
    dia_document_handle    document_handle,
    char                   *file_specification,
    dia_length             file_specification_length,
    dia_document_options    document_options);
```

### Description

The `dia_write_document` function writes the specified document as an ODIF data stream to the specified file.

The document options argument is reserved for future use.

### Arguments

**document\_handle**

The handle of the document to be written.

**file\_specification**

A string containing the file specification of the file to contain the ODIF data stream.

**file\_specification\_length**

The length in octets of the string containing the file specification of the file to contain the ODIF data stream.

**document\_options**

Application-specified options used to control the generation of the ODIF data stream. Specify `DIA_C_NO_OPTIONS`.

### Results

**Status**

Normally success, indicating that the operation was successful; otherwise one of the following errors is returned:

## `dia_write_document`

### Errors

#### `[DLA_E_INV_DOC]`

The document handle is not valid.

#### `[DLA_E_DOC_READ_ONLY]`

The document is read only.

#### `[DLA_E_NO_LOG_SRC]`

A sourced generic layout entity (`DLA_C_GEN_SRCD_CONT_VAR` or `DLA_C_GEN_SRCD_CONT_FIXED`) does not have any common content entities.

#### `[DLA_E_PROP_INCOMPLETE]`

A property does not have the required structure. This applies in particular where the sequence of entity handles that represents a Generator for subordinates does not have a required entry.

#### `[DLA_E_NO_SUBORD]`

A composite specific logical entity does not have a subordinate.

#### `[DLA_E_WRONG_LAST_SUBORD]`

The type of the last subordinate of a specific logical entity is of the wrong type.

#### `[DLA_E_MISSING_PEL_SPC_PROP]`

Only one of the properties Pel Spacing, Spacing, Length, and Pel Spaces has been specified.

#### `[DLA_E_MISSING_DIMENSION_PROP]`

Only one of the properties Fixed Dimension, Horizontal Dimension, and Vertical Dimension has been specified.

#### `[DLA_E_MISSING_POSITION_PROP]`

Only one of the properties Position, Horizontal Position, and Vertical Position has been specified.

#### `[DLA_E_MISSING_PROPERTY]`

A mandatory property has not been specified.

#### `[DLA_E_MISSING_LEVEL_NUMBER]`

The property level or index number has not been specified for a General numbering format, Segment number initialization or Resetting structure.

#### `[DLA_E_MISSING_STRING_FUNCTION]`

The DAP API has detected the absence of a string function whose presence is deemed mandatory.

#### `[DLA_E_OLA_WRITE_ERROR]`

An error has been detected by the ODA Level API toolkit. The application can obtain further information by using the function `dia_get_ola_write_error`.

#### `[DLA_E_CALLBACK_ERROR]`

Indicates that the callback function returned `DLA_C_STOP`. The write operation was terminated immediately the error was detected and so the ODIF file may not exist or the document it contains may not be complete.

#### `[DLA_E_REQ_ARGS_NOTSPEC]`

Not all arguments were correctly specified.

#### `[DLA_E_MEM_FAIL]`

An error occurred when allocating or deallocating memory. This is an unrecoverable error. The application must not attempt to continue.

#### `[DLA_E_NO_GEN_LAY_STRUCT]`

The document (including any referenced external document class) does not contain a generic layout structure. This is deemed mandatory.

**[DLA\_E\_NO\_GEN\_LOG\_STRUCT]**

The document (including any referenced external document class) does not contain a generic logical structure. This is deemed mandatory.

**[DLA\_E\_NO\_SPEC\_LOG\_STRUCT]**

The document (including any referenced external document class) does not contain a specific logical structure. This is deemed mandatory.

