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
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
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.

Adopted as an ECMA Standard by the General Assembly of June 1993
### Table of contents

3.4 Functions

- `dla_add_content`  
- `dla_append_char_set`  
- `dla_change_writing_mode`  
- `dla_convert_content`  
- `dla_create_document`  
- `dla_create_generic_basic_float`  
- `dla_create_generic_body_frame`  
- `dla_create_generic_comm_entity`  
- `dla_create_generic_fnote_area`  
- `dla_create_generic_frame`  
- `dla_create_generic_frame_seq`  
- `dla_create_generic_head_footer`  
- `dla_create_generic_pageset`  
- `dla_create_generic_snaking`  
- `dla_create_generic_synchronized`  
- `dla_create_gen_log_entity`  
- `dla_create_gen_log_entity_seq`  
- `dla_create_specific_entity`  
- `dla_delete_document`  
- `dla_duplicate_specific_entity`  
- `dla_error_class`  
- `dla_error_number`  
- `dla_failed`  
- `dla_find_first_entity`  
- `dla_find_next_entity`  
- `dla_find_profile`  
- `dla_fix_properties`  
- `dla_get_char_set_count`  
- `dla_get_choice`  
- `dla_get_construct_error`  
- `dla_get_content`  
- `dla_get_content_length`  
- `dla_get_entity_handle`  
- `dla_get_entity_type`  
- `dla_get_fix_error`  
- `dla_get_index_char_set`  
- `dla_get_index_char_set_count`  
- `dla_get_index_choice`  
- `dla_get_index_entity`  
- `dla_get_index_integer`  
- `dla_get_index_length`  
- `dla_get_index_string`  
- `dla_get_index_structure`  
- `dla_get_integer`  

1  
2  
4  
7  
9  
11  
14  
16  
18  
20  
22  
25  
28  
31  
33  
36  
38  
41  
44  
48  
50  
53  
54  
55  
56  
58  
60  
61  
63  
65  
68  
70  
73  
75  
78  
79  
81  
84  
86  
89  
92  
95  
98  
101  
104
dla_get_interp_error
107
dla_get_length
109
dla_get_nth_char_set
112
dla_get_ola_read_error
115
dla_get_ola_write_error
117
dla_get_other_parent
119
dla_get_size
121
dla_get_spec_char_set_count
124
dla_get_spec_choice
126
dla_get_spec_content
128
dla_get_spec_content_length
131
dla_get_spec_entity_handle
133
dla_get_spec_index_choice
135
dla_get_spec_index_entity
137
dla_get_spec_index_integer
139
dla_get_spec_index_length
141
dla_get_spec_index_string
143
dla_get_spec_index_structure
146
dla_get_spec_integer
148
dla_get_spec_length
150
dla_get_spec_nth_char_set
152
dla_get_spec_size
155
dla_get_spec_status
157
dla_get_spec_string
159
dla_get_spec_tab_stop
161
dla_get_spec_tab_stop_strings
163
dla_get_status
165
dla_get_string
168
dla_get_tab_stop
171
dla_get_tab_stop_strings
173
dla_init_toolkit
175
dla_instantiate_spec_log
177
dla_link_generic_entity
180
dla_link_generic_entity_seq
183
dla_locate_gen_entity
186
dla_make_frame_equiv
190
dla_move_to_child_spec_entity
193
dla_move_to_parent_spec_entity
195
dla_move_to_root_entity
197
dla_move_to_sibling_spec_entity
199
dla_read_document
201
dla_read_generic_doc
204
dla_read_writing_mode
207
dla_register_callback_function
209
dla_set_entity_handle
211
dla_set_index_entity
213
dla_set_index_integer
216
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>dla_set_index_null_value</td>
<td>219</td>
</tr>
<tr>
<td>dla_set_index_string</td>
<td>222</td>
</tr>
<tr>
<td>dla_set_index_structure</td>
<td>225</td>
</tr>
<tr>
<td>dla_set_integer</td>
<td>228</td>
</tr>
<tr>
<td>dla_set_null_value</td>
<td>230</td>
</tr>
<tr>
<td>dla_set_status</td>
<td>232</td>
</tr>
<tr>
<td>dla_set_string</td>
<td>235</td>
</tr>
<tr>
<td>dla_set_tab_stop</td>
<td>237</td>
</tr>
<tr>
<td>dla_term_toolkit</td>
<td>239</td>
</tr>
<tr>
<td>dla_unfix_properties</td>
<td>240</td>
</tr>
<tr>
<td>dla_version</td>
<td>242</td>
</tr>
<tr>
<td>dla_write_document</td>
<td>243</td>
</tr>
</tbody>
</table>
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_status value of DLA_C_PROP_UNSPECIFIED.
dla_add_content

Add 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_octet_string 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 one of the following types:

DLA_C_GEN_BODY_GEO (mode-2)
DLA_C_GEN_BODY_RASTER (mode-2)
DLA_C_GEN_BODY_TEXT (mode-2)
DLA_C_GEN_COMMON_GEO (mode-2)
DLA_C_GEN_COMMON_RASTER (mode-2)
DLA_C_GEN_COMMON_TEXT (mode-2)
DLA_C_GEN_ENTRY_GEOMETRIC (mode-2)
DLA_C_GEN_ENTRY_RASTER (mode-2)
DLA_C_GEN_ENTRY_TEXT (mode-2)
DLA_C_GEN_FOOTNOTE_TEXT (mode-2)
DLA_C_SPEC_BODY_GEO (mode-2)
DLA_C_SPEC_BODY_RASTER (mode-2)
DLA_C_SPEC_BODY_TEXT (mode-2)
DLA_C_SPEC_ENTRY_GEOMETRIC (mode-2)
DLA_C_SPEC_ENTRY_RASTER (mode-2)
DLA_C_SPEC_ENTRY_TEXT (mode-2)
DLA_C_SPEC_FOOTNOTE_TEXT (mode-2)

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.
dla_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

dla_status     dla_append_char_set(
        document_handle,     entity_handle,
        property_code,      character_set_id,
        string_length,      code_area,
        character_set_type);  

Description

The dla_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_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.
**dla_change_writing_mode**

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

**C Synopsis**

```c
#include <dladm.h>
dla_status dla_change_writing_mode(  
    dla_document_handle document_handle,  
    dla_document_options document_option);
```

**Description**

The `dla_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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_MODE_2</td>
<td>Application creates generic logical structure</td>
</tr>
</tbody>
</table>

**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_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. (FOD36 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.
**C Synopsis**

```
dla_status    dla_convert_content(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle,
    dla_conversion_type type_of_conversion);
```

**Description**

The `dla_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 ODIF. 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.
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 69372 to ISO 8859-1 character sets
- DLA_C_CHAR_8859_TO_6937: ISO 8859-1 to ISO 69372 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.
dla_create_document

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

C Synopsis

dla_status      dla_create_document(  
dla_toolkit_handle  toolkit_handle,  
dla_dap_level      dap_level,  
dla_document_options  document_options,  
char              *store_file,  
dla_length        file_spec_length,  
dla_document_handle  *document_handle_return);

Description

The dla_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_CPROFILE

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

DLA_CSPECLOG_ROOT
DLA_CGENLOG_ROOT
DLA_CGENLAY_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.
Arguments

toolkit_handle
The handle of the toolkit being operated on.

dap_level
An enumerated value that indicates the required DAP level.

<table>
<thead>
<tr>
<th>Dap_level</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_DAP_LEVEL_1</td>
<td>DAP level 1</td>
</tr>
<tr>
<td>DLA_C_DAP_LEVEL_2</td>
<td>DAP level 2</td>
</tr>
<tr>
<td>DLA_C_DAP_LEVEL_3</td>
<td>DAP level 3</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_NO_OPTIONS</td>
<td>Equivalent to DLA_C_MODE_1</td>
</tr>
<tr>
<td>DLA_C_MODE_1</td>
<td>Toolkit creates generic logical structure</td>
</tr>
<tr>
<td>DLA_C_AUTO_GEN</td>
<td>Toolkit creates generic logical structure</td>
</tr>
<tr>
<td></td>
<td>(Equivalent to DLA_C_MODE_1)</td>
</tr>
<tr>
<td>DLA_C_MODE_2</td>
<td>Application creates generic logical structure</td>
</tr>
<tr>
<td>DLA_C_WITH_EXTERNAL</td>
<td>Application will read generic document from ODIF</td>
</tr>
<tr>
<td>DLA_C_FOR_EXTERNAL</td>
<td>Application will supply ODIF identifiers for all entities</td>
</tr>
<tr>
<td>DLA_C_FOR_RESOURCE</td>
<td>Application will supply ODIF identifiers for all entities and the resources property</td>
</tr>
</tbody>
</table>

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.
dlacentral, generic basic float

dlacentral, generic basic float

Creates a generic basic float entity.

C Synopsis

dla_status dla_create_generic_basic_float(
    dla_document_handle document_handle,
    dla_entity_handle generic_parent_handle,
    dla_entity_handle *generic_basic_float_handle);

Description

The dla_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.
dl_create_generic_body_frame

dl_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**

```c
dia_status    dia_create_generic_comm_entity(  
    dla_document_handle   document_handle,  
    dla_entity_handle    frame_entity_handle,  
    dla_entity_type      common_entity_type,  
    dla_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 dla_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_GEOM
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

document_entity_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_INVENTITY_OP]
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.

Standard ECMA-187 - Volume 2
dlacreate_generic_fnote_area

dlacreate_generic_fnote_area

Creates a generic footnote area entity.

C Synopsis

```
dla_status dla_create_generic_fnote_area(
    dla_document_handle document_handle,
    dla_entity_handle generic_composite_frame_handle,
    dla_entity_handle *generic_footnote_area_handle);
```

Description

The dla_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.
**dlᵃ_create_generic_frame**

**C Synopsis**

```c
void *dlᵃ_create_generic_frame( 
    void *document_handle, 
    int frame_entity_type, 
    void *generic_parent_handle, 
    void *generic_frame_handle);
```

**Description**

The `dlᵃ_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 `dlᵃ_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
dla_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 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_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_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

dla_status    dla_create_generic_frame_seq(
    dla_document_handle    document_handle,
    dla_entity_type       frame_entity_type,
    dla_entity_handle     generic_parent_handle,
    dla_integer           index,
    dla_entity_handle     *generic_frame_handle);

Description

The dla_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 dla_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 dla_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.
dl_a_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_SRC_GD_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 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_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.
Create a cluster of generic layout entities associated with a generic header or footer.

C Synopsis

dla_status  dla_create_generic_head_footer(
    dla_document_handle  document_handle,
    dla_entity_handle    generic_page_handle,
    dla_header_footer_type header_footer_type,
    dla_integer          num_content_entities,
    dla_entity_handle    *generic_head_foot_handle);

Description

The dla_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_SRCD_CONT_VAR or DLA_C_GEN_SRCD_CONT_FIXED.

The num_content_entities argument specifies the required number of DLA_C_GEN_SRCD_CONT_VAR or DLA_C_GEN_SRCD_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 SourcedContentViewable 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.
header_footer_type
The type of header or footer. An enumerated value, that is one of the following:

<table>
<thead>
<tr>
<th>Header_footer_type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_COMP_HEAD_FIXED</td>
<td>A composite header with a sequence of subordinate entities of type</td>
</tr>
<tr>
<td></td>
<td>DLA_C_GEN_SRCD_CONT_FIXED</td>
</tr>
<tr>
<td>DLA_C_COMP_HEAD_VAR</td>
<td>A composite header with a sequence of subordinate entities of type</td>
</tr>
<tr>
<td></td>
<td>DLA_C_GEN_SRCD_CONT_VAR</td>
</tr>
<tr>
<td>DLA_C_COMP FOOT_FIXED</td>
<td>A composite footer with a sequence of subordinate entities of type</td>
</tr>
<tr>
<td></td>
<td>DLA_C_GEN_SRCD_CONT_FIXED</td>
</tr>
<tr>
<td>DLA_C_COMP FOOT_VAR</td>
<td>A composite footer with a sequence of subordinate entities of type</td>
</tr>
<tr>
<td></td>
<td>DLA_C_GEN_SRCD_CONT_VAR</td>
</tr>
<tr>
<td>DLA_C_BASIC_HEAD</td>
<td>A basic header</td>
</tr>
<tr>
<td>DLA_C_BASIC FOOT</td>
<td>A basic footer</td>
</tr>
</tbody>
</table>

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.
[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.
**dla_create_generic_pageset**

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

**C Synopsis**

```c
dla_status  dla_create_generic_pageset(  
dla_document_handle   document_handle,  
dla_pageset_type      pageset_type,  
dla_entity_handle     *generic_pageset_handle);
```

**Description**

The `dla_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.
**DLA_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:

<table>
<thead>
<tr>
<th>Pageset_type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_TITLE_PAGE</td>
<td>Create a DLA_C_GEN_PAGE entity representing a title page.</td>
</tr>
<tr>
<td>DLA_C_SINGLE_PAGE</td>
<td>Create a DLA_C_GEN_PAGE entity representing a repetition of single pages.</td>
</tr>
<tr>
<td>DLA_C_RECTO_VERSO</td>
<td>Create DLA_C_GEN_RECTO_PAGE and DLA_C_GEN_VERSO_PAGE entities representing a repetition of recto and verso pages.</td>
</tr>
<tr>
<td>DLA_C_TITLE_SINGLE_PAGE</td>
<td>Create two DLA_C_GEN_PAGE entities representing a title page and a repetition of single pages.</td>
</tr>
<tr>
<td>DLA_C_TITLE_VERSO_RECTO</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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**

```c
int dla_status (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:

<table>
<thead>
<tr>
<th>Snaking_type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_SNAKING_REP</td>
<td>A repetition of a single (composite) column</td>
</tr>
<tr>
<td></td>
<td>variable entity is required.</td>
</tr>
<tr>
<td>DLA_C_SNAKING_SEQ</td>
<td>A sequence of different (composite) column</td>
</tr>
<tr>
<td></td>
<td>variable entities is required.</td>
</tr>
</tbody>
</table>

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 (9999999) 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.
dla_create_generic_synchronized

dla_create_generic_synchronized

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

C Synopsis

dla_status dla_create_generic_synchronized(dla_document_handle document_handle,
        dla_entity_handle     generic_composite_body_handle,
        dla_integer           num_col_fix,
        dla_entity_handle     *generic_synch_handle);

Description

The dla_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.
dlacreate_gen_log_entity

dlacreate_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

dla_status dla_create_gen_log_entity(
  dla_document_handle document_handle,
  dla_entity_type entity_type,
  dla_entity_handle parent_entity,
  dla_entity_handle *entity_handle_return);

Description

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

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

**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_REQARGS_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.
**C Synopsis**

```c
#include <dlacore.h>
dla_status dla_create_gen_log_entity_seq(
    dla_document_handle document_handle,
    dla_entity_type entity_type,
    dla_entity_handle parent_entity,
    dla_integer index,
    *entity_handle_return);
```

**Description**

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

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_GEN_PASSAGE</td>
<td>Passage</td>
</tr>
<tr>
<td>DLA_C_GEN_NUM_SEG</td>
<td>NumberedSegment</td>
</tr>
<tr>
<td>DLA_C_GEN_NUMBER</td>
<td>Number</td>
</tr>
<tr>
<td>DLA_C_GEN_TITLE</td>
<td>Title</td>
</tr>
<tr>
<td>DLA_C_GEN_CAPTION</td>
<td>Caption</td>
</tr>
<tr>
<td>DLA_C_GEN_PARAGRAPH</td>
<td>Paragraph</td>
</tr>
<tr>
<td>DLA_C_GEN_PHRASE</td>
<td>Phrase</td>
</tr>
<tr>
<td>DLA_C_GEN_FOOTNOTE</td>
<td>Footnote</td>
</tr>
<tr>
<td>DLA_C_GEN_FOOTNOTE_TEXT</td>
<td>FootnoteText</td>
</tr>
<tr>
<td>DLA_C_GEN FIGURE</td>
<td>Figure</td>
</tr>
<tr>
<td>DLA_C_GEN_BODY_TEXT</td>
<td>BodyText</td>
</tr>
<tr>
<td>DLA_C_GEN_REFERENCE</td>
<td>Reference</td>
</tr>
<tr>
<td>DLA_C_GEN_BODY_RASTER</td>
<td>BodyRaster</td>
</tr>
<tr>
<td>DLA_C_GEN_BODY_GEOM</td>
<td>BodyGeometric</td>
</tr>
<tr>
<td>DLA_C_GEN DESCRIPTION</td>
<td>Description</td>
</tr>
<tr>
<td>DLA_C_GEN_ARTWORK</td>
<td>Artwork</td>
</tr>
<tr>
<td>DLA_C_GEN_NUMBERED_LIST</td>
<td>NumberedList</td>
</tr>
<tr>
<td>DLA_C_GEN_UNNUMBERED_LIST</td>
<td>UnNumberedList</td>
</tr>
<tr>
<td>DLA_C_GEN_DEFINITION_LIST</td>
<td>DefinitionList</td>
</tr>
<tr>
<td>DLA_C_GEN_LIST_ITEM</td>
<td>ListItem</td>
</tr>
<tr>
<td>DLA_C_GEN_LIST_TERM</td>
<td>ListTerm</td>
</tr>
<tr>
<td>DLA_C_GEN_TABLE</td>
<td>Table</td>
</tr>
<tr>
<td>DLA_C_GEN_ROW</td>
<td>Row</td>
</tr>
<tr>
<td>DLA_C_GEN_TABLE_COMPONENT</td>
<td>TableComponent</td>
</tr>
<tr>
<td>DLA_C_GEN_ROW_COMPONENT</td>
<td>RowComponent</td>
</tr>
<tr>
<td>DLA_C_GEN_FORM</td>
<td>Form</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_ELEMENT</td>
<td>EntryElement</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_GROUP</td>
<td>EntryGroup</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_TEXT</td>
<td>EntryText</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_RASTER</td>
<td>EntryRaster</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_GEOMETRIC</td>
<td>EntryGeometric</td>
</tr>
</tbody>
</table>

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

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_RANGE]
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.
dl�示 create_specific_entity

dl示 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:

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

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.
SIA_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.

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

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

dlad_delete_document

Deletes a DAP API document.

C Synopsis

dla_status      dla_delete_document(
                 dla_document_handle  document_handle);

Description

The dla_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 ODA API that generic documents cannot be deleted, except through deletion of the owning document.

[DLA_E_REQ_ARGS_NOTSPECC]
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.
C Synopsis

dla_status  dla_duplicate_specific_entity(
    dla_document_handle  document_handle,
    dla_entity_handle    model_entity,
    dla_entity_handle    link_entity,
    dla_link_option      link_option,
    dla_entity_handle    *entity_handle_return);

Description

The dla_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 dla_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.

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C.Link_Before</td>
<td>The new entity will be linked to the parent of link_entity and will precede it.</td>
</tr>
<tr>
<td>DLA_C.Link_After</td>
<td>The new entity will be linked to the parent of link_entity and will succeed it.</td>
</tr>
<tr>
<td>DLA_C.Link_As_First</td>
<td>The new entity will be linked to the link_entity, becoming its first subordinate.</td>
</tr>
<tr>
<td>DLA_C.Link_As_Last</td>
<td>The new entity will be linked to the link_entity, becoming its last subordinate.</td>
</tr>
</tbody>
</table>

duplicate_entity

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, 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.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_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 FOD36 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

dla_integer    dla_error_class(
    dla_status    error_status);

Description

The dla_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

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.
dla_failed

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

C Synopsis

dla_boolean   dla_failed(
    dla_status   status);

Description

The dla_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

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

C Synopsis

```c
    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:

<table>
<thead>
<tr>
<th>Structure type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_GENERIC_LAYOUT</td>
<td>Generic layout first entity</td>
</tr>
<tr>
<td>DLA_C_GENERIC_LOGICAL</td>
<td>Generic logical first entity</td>
</tr>
</tbody>
</table>

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**

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

Finds the document profile entity in a document.

C Synopsis

```c
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.
dla_fix_properties

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

C Synopsis

dla_status dla_fix_properties(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle);

Description

The dla_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 dla_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:
Errors

[DLA_E_INV_ENTITY]
The entity does not exist.

[DLA_E_DOC_READ_ONLY]
The document is read only.

[DLA_E_INVENTITY_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.
dla_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 dla_get_nth_char_set can be used to extract information from an individual designating sequence.

C Synopsis

dla_status   dla_get_char_set_count(
    dla_document_handle   document_handle,
    dla_entity_handle     entity_handle,
    dla_property_code     property_code,
    dla_integer           *count,
    dla_property_status   *property_status,
    dlaDerived_location   *derived_location,
    dla_entity_handle     *derived_entity_handle);

Description

The function dla_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.
dla_get_char_set_count

property_status [write]
Receives the status of the property which is one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
</tbody>
</table>

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.
**dla_get_choice**

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

**C Synopsis**

```c
#include <dlagent.h>

dla_status    dla_get_choice(  
dla_document_handle   document_handle,  
dla_entity_handle     entity_handle,  
dla_structure_handle  structure_handle,  
dla_property_code     property_code,  
dla_integer           *choice_identifier,  
dla_property_status   *property_status,  
dla Derived Location  *derived_location,  
dla_entity_handle     *derived_entity_handle);
```

**Description**

The `dla_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.
**dla_get_choice**

**property_status [write]**
Receives the status of a property which is one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

**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_OTER]
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**

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**

```c
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_NOPROPERTY` is returned.
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 \texttt{dia_write_document}.

The value is one of:

\texttt{DLA\_E\_NO\_LOG\_SRC}
The given sourced generic layout entity (\texttt{DLA\_C\_GEN\_SRCD\_CONT\_VAR} or \texttt{DLA\_C\_GEN\_SRCD\_CONT\_FIXED}) does not have any common content entities.

\texttt{DLA\_E\_NO\_SUBORD}
The given composite specific logical entity does not have a subordinate.

\texttt{DLA\_E\_WRONG\_LAST\_SUBORD}
The last subordinate of a specific logical entity is of the wrong type.

\texttt{DLA\_E\_MISSING\_PEL\_SPC\_PROP}
Only one of the properties Pel Spacing, Spacing, Length and Pel Spaces has been specified.

\texttt{DLA\_E\_MISSING\_DIMENSION\_PROP}
Only one of the properties Fixed Dimension, Horizontal Dimension and Vertical Dimension has been specified.

\texttt{DLA\_E\_MISSING\_POSITION\_PROP}
Only one of the properties Position, Horizontal Position and Vertical Position has been specified.

\texttt{DLA\_E\_MISSING\_PROPERTY}
The given mandatory property has not been specified. The property code is returned in the variable \texttt{property\_code\_return}.

\texttt{DLA\_E\_MISSING\_LEVEL\_NUMBER}
The property level number has not been specified for a Segment Number Initialization or Resetting structure. The variable \texttt{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

\texttt{[DLA\_E\_INV\_DOC]}
The document handle is not valid.

\texttt{[DLA\_E\_NO\_FAIL]}
The most recent call of the function \texttt{dia_write_document} did not fail with a construction error.

\texttt{[DLA\_E\_REQ\_ARGS\_NOTSPECF]}
Not all arguments were correctly specified.

\texttt{[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

```c
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

Standard ECMA-187 - Volume 2
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_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**

```c
    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.
**dla_get_entity_handle**

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

**C Synopsis**

```c
#include "dla.h"

dla_status  dla_get_entity_handle(
    dla_document_handle   document_handle,
    dla_entity_handle     entity_handle,
    dla_structure_handle  structure_handle,
    dla_property_code     property_code,
    dla_entity_handle     *property_entity_handle,  
    dla_property_status   *property_status,
    dla_derived_location  *derived_location,
    dla_entity_handle     *derived_entity_handle);
```

**Description**

The `dla_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

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_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_ARC 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.
**dla_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**

```c
dla_status dla_get_fix_error(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle,
    dla_property_code *property_code_return,
    dla_integer *err_num_return);
```

**Description**

The `dla_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 `dla_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 `dla_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 `dla_fix_properties`. 


daGetFixError

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 dla_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.
**dla_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**

```
dla_status    dla_get_index_char_set(   
    dla_document_handle document_handle,   
    dla_entity_handle profile_handle,   
    dla_integer index,   
    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_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 using the function `dla_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.
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 nth 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:

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_G0</td>
</tr>
<tr>
<td>DLA_C_G1</td>
</tr>
<tr>
<td>DLA_C_G2</td>
</tr>
<tr>
<td>DLA_C_G3</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_SIN_94</td>
</tr>
<tr>
<td>DLA_C_SIN_96</td>
</tr>
<tr>
<td>DLA_C_MUL_94</td>
</tr>
<tr>
<td>DLA_C_MUL_96</td>
</tr>
</tbody>
</table>

property_status [write]
Receives the status of a property; one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
</tbody>
</table>

Results

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

[DILA_E_INV_ENTITY]
The entity does not exist.

[DILA_E_INV_CODE]
The entity is not valid for this function.

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

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

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

[DILA_E_INV_N_VALUE]
There are less than n_value escape sequences within the property octet string value.

[DILA_E_REQ_ARGS_NOTSPEC]
Not all arguments were correctly specified.

[DILA_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**

**C Synopsis**

```c
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_CHR_SET` indicating the graphic character set property, that is `DLA_C_NONB_GR_CHR_SET`.

The function `dia_get_index_char_set_count` 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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
</tbody>
</table>

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_RNG]
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**

**C Synopsis**

```c
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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

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_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.
**dla_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**

```c
dla_status    dla_get_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_entity_handle     *property_entity_handle,
    dla_property_status   *property_status,
    dla Derived location  *derived_location,
    dla_entity_handle     *derived_entity_handle);
```

**Description**

The `dla_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.
property_status [write]
Receives the status of a property; one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

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_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.
**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**

```c
sla_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 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 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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

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_RNG]
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**

```c
#include <dla.h>

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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

**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_PROPOPER]**
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.
**dla_get_index_string**

Reads the value of a property applying to an entity, for an indexed property of octet string type.

**C Synopsis**

```c
int dla_get_index_string(  
dla_status    dla_get_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_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_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 `dla_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

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:
dlaget_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.
**dlia_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**

```c
dlia_status dlia_get_index_structure(
    dlia_document_handle    document_handle,
    dlia_entity_handle     entity_handle,
    dlia_structure_handle  structure_handle,
    dlia_property_code     property_code,
    dlia_integer           index,
    dlia_structure_handle  *new_structure_handle,
    dlia_property_status   *property_status,
    dliaDerived_location   *derived_location,
    dlia_entity_handle     *derived_entity_handle);
```

**Description**

The `dlia_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

**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_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.
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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

**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**

```c
dia_status  dla_get_interp_error(
    dla_toolkit_handle  toolkit_handle,
    dla_length          ident_buffer_length,
    dla_octet_string    ident_return_buffer,
    dla_length          *ident_length_return,
    dla_entity_type     *entity_type_return,
    dla_property_code   *property_code_return,
    dla_integer         *err_num_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.

er_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.
**dla_get_length**

Reads the length of the value of a property applying to an entity, for a property of octet string type.

**C Synopsis**

```c
void dla_get_length(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code property_code,
    *string_length,
    *property_status,
    *derived_location,
    *derived_entity_handle);
```

**Description**

The `dla_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.
dla_get_length

property_status [write]
Receives the status of a property; one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

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.
This function interprets the nth 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

```c
int 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,
    *string_length,
    *code_area,
    *character_set_type,
    *property_status,
    *derived_location,
    *derived_entity_handle);
```

Description

The function dla_get_nth_char_set reads and interprets the nth 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 nth 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.

color_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
</tbody>
</table>
**dla_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.
**dla_get_ola_read_error**

Returns the details of an error in reading an ODIF document, as determined by the ODA API.

**C Synopsis**

```c
dla_status    dla_get_ola_read_error(    
    dla_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 `dla_get_ola_read_error` function returns the details of the previous failure to read an ODIF document, either with `dla_read_document` or with `dla_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_err_num`.

**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_err_num`. 
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_REQARGS_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_ola_write_error**

Returns the details of an error in writing an ODIF document, as determined by the ODA API.

**C Synopsis**

```c
dla_status    dla_get_ola_write_error(
    dla_document_handle    document_handle,
    ola.constituent.handle *constituent.handle.return,
    ola.attribute.code    *attribute.code.return,
    ola.integer            *ola.err.num.return);
```

**Description**

The `dla_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`. 
dla_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**

```c
void dia_get_other_parent(
    const void *document_handle,
    const void *entity_handle,
    int content_offset,
    const void *other_parent_handle,
    int 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.
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_LAI D_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.
dla_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

dla_status  dla_get_size(
    dla_document_handle  document_handle,
    dla_entity_handle    entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code    property_code,
    dla_integer          *sequence_size,
    dla_property_status  *property_status,
    dla_derived_location *derived_location,
    dla_entity_handle    *derived_entity_handle);

Description

The function dla_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 DLA_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.
**property_status [write]**
Receives the status of a property; one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

**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.
dla_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

```c

dla_status      dla_get_spec_char_set_count(
    dla_document_handle document_handle,
    dla_entity_handle    entity_handle,
    dla_property_code   property_code,
    dla_integer         *count,
    dla_property_status *property_status);
```

### Description

The function `dla_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 `dla_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
</tbody>
</table>

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.
**dla_get_spec_choice**

**dla_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**

```c
dla_status  dla_get_spec_choice(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code property_code,
    dla_integer choice_identifier,
    dla_property_status *property_status);
```

**Description**

The *dla_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

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.
**C Synopsis**

```c
int 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,  
    *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`
The application can invoke the dla_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.
[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**

```c
dia_status    dia_get_spec_content_length(
    dia_document_handle    document_handle,
    dia_entity_handle     entity_handle,
    dla_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
dlai_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.
**dla_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**

```c
    dla_status    dla_get_spec_entity_handle(
                        dla_document_handle    document_handle,
                        dla_entity_handle       entity_handle,
                        dla_structure_handle   structure_handle,
                        dla_property_code      property_code,
                        dla_entity_handle       *property_entity_handle,
                        dla_property_status     *property_status);
```

**Description**

The **dla_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.
property_status [write]
Receives the status of a property, one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

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_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**

```c
dla_status  dla_get_spec_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);
```

**Description**

The **dla_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.
### dla_get_spec_index_choice

**property_status [write]**
Receives the status of a property, one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

**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.
**dla_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**

```c
int dla_get_spec_index_entity(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code property_code,
    int index,
    dla_entity_handle *property_entity_handle,
    dla_property_status *property_status);
```

**Description**

The `dla_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

**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.
dla_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

```c
sla_status    dla_get_spec_index_integer(
    dla_document_handle    document_handle,
    dla_entity_handle     entity_handle,
    dla_structure_handle  structure_handle,
    dla_property_code     property_code,
    dla_quantity          index,
    integer_value         *integer_value,
    property_status       *property_status);
```

Description

The `dla_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 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 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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

**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_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_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**

```c
    dla_status  dla_get_spec_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); 
```

**Description**

The **dla_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

**Results**

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

**Errors**

**[DLA_E_NOT_APROPERTY_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.
**dla_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**

```c
void *dla_get_spec_index_string(
    void *document_handle,
    void *entity_handle,
    void *structure_handle,
    void *property_code,
    void *index,
    void *buffer_length,
    void *string_length,
    void *property_status);
```

**Description**

The `dla_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 `dla_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

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

```c
void dla_get_spec_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_structure_handle *new_structure_handle,
    dla_property_status *property_status);
```

**Description**

The `dla_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

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.
**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**

```c
int32_t dla_get_spec_integer(  
    dla_document_handle document_handle,  
    dla_entity_handle entity_handle,  
    dla_structure_handle structure_handle,  
    dla_property_code property_code,  
    int32_t* integer_value,  
    int32_t* 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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

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.
dla_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

```c
dla_status   dla_get_spec_length(
    dla_document_handle  document_handle,
    dla_entity_handle    entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code    property_code,
    dla_length           *string_length,
    dla_property_status  *property_status);
```

Description

The `dla_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

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**

This function interprets the nth 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**

```c
void 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 nth 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 nth 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.

code area [write]
A buffer to receive the final character identifier of a graphic character set.

string_length [write]
Length of character_set_id returned.

value 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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
</tbody>
</table>

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_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**

```c
dia_status    dia_get_spec_size(
    dla_document_handle  document_handle,
    dla_entity_handle    entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code    property_code,
    dla_integer          *sequence_size,
    dla_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 DLA_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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

**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.
**dlaget_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**

```c
dl_status dlaget_spec_status(
    dlag_document_handle document_handle,
    dlag_entity_handle entity_handle,
    dlag_structure_handle structure_handle,
    dlag_property_code property_code,
    dlag_property_status * property_status);
```

**Description**

The `dlaget_spec_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 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.
**DIA_GET_SPEC_STATUS**

**property_status [write]**
Receives the status of a property, one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

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**

[DIA_E_NOT_A_PROPERTY_CODE]
The property code argument does not represent any property.

[DIA_E_INV_CODE]
The property does not apply to the entity.

[DIA_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.

[DIA_E_WRONG_STRUCTURE]
The given structure does not exist for this entity.

[DIA_E_INV_ENTITY]
The entity does not exist.

[DIA_E_REQ_ARGS_NOTSPEC]
Not all arguments were correctly specified.

[DIA_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_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**

```c
#include "dla.h"

dla_status  dla_get_spec_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);
```

**Description**

The `dla_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 `dla_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.
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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null</td>
</tr>
</tbody>
</table>

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_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**

```c
dla_status  dla_get_spec_tab_stop(
    dla_document_handle  document_handle,  
    dla_entity_handle    entity_handle,    
    dla_integer          index,            
    dla_length           *tab_reference_length, 
    dla_integer          *tab_position,      
    dla_tab_alignment    *alignment,        
    dla_length           *alignment_string_length);
```

**Description**

The `dla_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.
alignment [write]
The alignment type of the tabulation stop at the specified position.
An enumerated value that is one of:

<table>
<thead>
<tr>
<th>Alignment type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_TAB_ALIGN_START</td>
<td>Start aligned</td>
</tr>
<tr>
<td>DLA_C_TAB_ALIGN_END</td>
<td>End aligned</td>
</tr>
<tr>
<td>DLA_C_TAB_ALIGN_CENTRED</td>
<td>Center aligned</td>
</tr>
<tr>
<td>DLA_C_TAB_ALIGN_AROUND</td>
<td>Align around</td>
</tr>
</tbody>
</table>

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

[DIA_E_INDEX_OUT_OF_RNG]
The index is out of range for valid indexes for the entries in the line layout table.

[DIA_E_INV_ENTITY_OPER]
The requested operation is not valid for this entity.

[DIA_E_INV_ENTITY]
The entity does not exist.

[DIA_E_REQ_ARGS_NOTSPEC]
Not all arguments were correctly specified.

[DIA_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_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

```c
dla_status dla_get_spec_tab_stop_strings(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle,
    dla_index index,
    dla_length tab_reference_buffer_length,
    dla_length alignment_string_buffer_length,
    dla_octet_string tab_reference_string,
    dla_length *tab_reference_length,
    dla_length *alignment_string_length,
    )
```

Description

The `dla_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 DLA_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_RNG]**
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.
**dla_get_status**

Gets the status of a property applying to an entity.

**C Synopsis**

```c
#include <dle.h>

dla_status  dla_get_status(
    dla_document_handle document_handle,
    dla_entity_handle  entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code  property_code,
    *property_status,
    *derived_location,
    *derived_entity_handle);
```

**Description**

The `dla_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.
**dla_get_status**

**property_status [write]**
Receives the status of a property; one of the following:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

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_INVENTITY]
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**

```c
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:

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_SPECIFIED</td>
<td>The property status is specified.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified.</td>
</tr>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
</tbody>
</table>

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_get_string

[DLA_EPROP_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_INVENTITY]
The entity does not exist.

[DLA_E_REQARGS_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_tab_stop**

Reads the details of a tabulation stop.

**C Synopsis**

```c
#include "dla.h"

dla_status dla_get_tab_stop(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle,
    dla_integer index,
    dla_length *tab_reference_length,
    dla_integer *tab_position,
    dla_tab_alignment *alignment,
    dla_length *alignment_string_length);
```

**Description**

The `dla_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.
alignment [write]
The alignment type of the tabulation stop at the specified position. An enumerated value that is one of:

<table>
<thead>
<tr>
<th>Alignment type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_TAB_ALIGN_START</td>
<td>Start aligned</td>
</tr>
<tr>
<td>DLA_C_TAB_ALIGN_END</td>
<td>End aligned</td>
</tr>
<tr>
<td>DLA_C_TAB_ALIGN_CENTRED</td>
<td>Center aligned</td>
</tr>
<tr>
<td>DLA_C_TAB_ALIGN_AROUND</td>
<td>Align around</td>
</tr>
</tbody>
</table>

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_RNG]
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**

```c
dia_status dia_get_tab_stop_strings(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle,
    dla_integer index,
    dla_length tab_reference_buffer_length,
    dla_length alignment_string_buffer_length,
    dla_octet_string tab_reference_string,
    dla_length *tab_reference_length,
    dla_octet_string alignment_string,
    dla_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 `DLA_C_NULL_STRING` if the tab reference is not required.
**dlar_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_RNG]**  
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_REQARGS_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_init_toolkit

Initializes an instance of the DAP Level API toolkit.

C Synopsis

```c
dla_status      dla_init_toolkit(
    dla_toolkit_options toolkit_options,
    dla_integer       toolkit_memory_allocation,
    dla_integer       max_open_files,
    dla_toolkit_handle *toolkit_handle);
```

Description

The `dla_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.

<table>
<thead>
<tr>
<th>Option type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_NO_OPTIONS</td>
<td>No options are specified.</td>
</tr>
</tbody>
</table>

- **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.
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.
**C Synopsis**

```c
void dla_status (dla_document_handle document_handle,
                dla_entity_handle generic_entity,
                dla_entity_handle link_entity,
                dla_link_option link_option,
                dla_entity_handle *entity_handle_return);
```

**Description**

The `dlaInstantiateSpecLog` 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 `dlaCreateGenLogEntry`, 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.
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.

<table>
<thead>
<tr>
<th>Option value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_LINK_BEFORE</td>
<td>The new entity will be linked to the parent of link_entity and will precede it</td>
</tr>
<tr>
<td>DLA_C_LINK_AFTER</td>
<td>The new entity will be linked to the parent of link_entity and will succeed it</td>
</tr>
<tr>
<td>DLA_C_LINK_AS_FIRST</td>
<td>The new entity will be linked to the link_entity, becoming its first subordinate</td>
</tr>
<tr>
<td>DLA_C_LINK_AS_LAST</td>
<td>The new entity will be linked to the link_entity, becoming its last subordinate</td>
</tr>
</tbody>
</table>

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**

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**

```c
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 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_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**

```c
dla_status dla_link_generic_entity_seq(
    dla_document_handle document_handle,
    dla_entity_handle parent_entity_handle,
    dla_integer index,
    dla_entity_handle child_entity_handle);
```

**Description**

The `dla_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 `dla_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.
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_RNG]
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 only be one header area on a page and it is already present.

[DLA_E_FOOTER_EXISTS]
There can only be one footer area on a page and it is already present.

[DLA_E_FNOTE_AREA_EXISTS]
There can only be 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**

Locates a generic entity using its ODIF identifier.

**C Synopsis**

```c
void 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 ODIF identifier of the required entity.

- **entity_id_length**
  The length of the ODIF 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:

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_GEN_LOG_ROOT</td>
<td>Generic DocumentLogicalRoot</td>
</tr>
<tr>
<td>DLA_C_GEN_PASSAGE</td>
<td>Generic Passage</td>
</tr>
<tr>
<td>DLA_C_GEN_NUM_SEG</td>
<td>Generic NumberedSegment</td>
</tr>
<tr>
<td>DLA_C_GEN_NUMBER</td>
<td>Generic Number</td>
</tr>
<tr>
<td>DLA_C_GEN_TITLE</td>
<td>Generic Title</td>
</tr>
<tr>
<td>DLA_C_GEN_CAPTION</td>
<td>Generic Caption</td>
</tr>
<tr>
<td>DLA_C_GEN_PARAGRAPH</td>
<td>Generic Paragraph</td>
</tr>
<tr>
<td>DLA_C_GEN_PHRASE</td>
<td>Generic Phrase</td>
</tr>
<tr>
<td>DLA_C_GEN_FOOTNOTE</td>
<td>Generic Footnote</td>
</tr>
<tr>
<td>DLA_C_GEN_FOOTNOTE_NUM</td>
<td>Generic FootnoteNumber</td>
</tr>
<tr>
<td>DLA_C_GEN_FOOTNOTE_REF</td>
<td>Generic FootnoteReference</td>
</tr>
<tr>
<td>DLA_C_GEN_FOOTNOTE_BODY</td>
<td>Generic FootnoteBody</td>
</tr>
<tr>
<td>DLA_C_GEN_FOOTNOTE_TEXT</td>
<td>Generic FootnoteText</td>
</tr>
<tr>
<td>DLA_C_GEN_FIGURE</td>
<td>Generic Figure</td>
</tr>
<tr>
<td>DLA_C_GEN_BODY_TEXT</td>
<td>Generic BodyText</td>
</tr>
<tr>
<td>DLA_C_GEN_REFERENCE</td>
<td>Generic Reference</td>
</tr>
<tr>
<td>DLA_C_GEN_REF_CONTENT</td>
<td>Generic ReferencedContent</td>
</tr>
<tr>
<td>DLA_C_GEN_BODY_RASTER</td>
<td>Generic BodyRaster</td>
</tr>
<tr>
<td>DLA_C_GEN_BODY_GEOM</td>
<td>Generic BodyGeometric</td>
</tr>
<tr>
<td>DLA_C_GEN_COMMON_CONTENT</td>
<td>Generic CommonContent</td>
</tr>
<tr>
<td>DLA_C_GEN_COMMON_TEXT</td>
<td>Generic CommonText</td>
</tr>
<tr>
<td>DLA_C_GEN_COMMON_REFERENCE</td>
<td>Generic CommonReference</td>
</tr>
<tr>
<td>DLA_C_GEN_COMMON_NUMBER</td>
<td>Generic CommonNumber</td>
</tr>
<tr>
<td>DLA_C_GEN_CURRENT_INSTANCE</td>
<td>Generic CurrentInstance</td>
</tr>
<tr>
<td>DLA_C_GEN_COMMON_RASTER</td>
<td>Generic CommonRaster</td>
</tr>
<tr>
<td>DLA_C_GEN_COMMON_GEOM</td>
<td>Generic CommonGeometric</td>
</tr>
<tr>
<td>DLA_C_GEN_PAGE_NUMBER</td>
<td>Generic PageNumber</td>
</tr>
<tr>
<td>DLA_C_GEN_TABLE_NUMBER</td>
<td>Generic TableNumber</td>
</tr>
<tr>
<td>DLA_C_GEN_DESCRIPTION</td>
<td>Generic Description</td>
</tr>
<tr>
<td>DLA_C_GEN_ARTWORK</td>
<td>Generic Artwork</td>
</tr>
<tr>
<td>DLA_C_GEN_NUMBERED_LIST</td>
<td>Generic NumberedList</td>
</tr>
<tr>
<td>DLA_C_GEN_UNNUMBERED_LIST</td>
<td>Generic UnNumberedList</td>
</tr>
<tr>
<td>DLA_C_GEN_DEFINITION_LIST</td>
<td>Generic DefinitionList</td>
</tr>
<tr>
<td>DLA_C_GEN_LIST_ITEM</td>
<td>Generic ListItem</td>
</tr>
<tr>
<td>DLA_C_GEN_LIST_TERM</td>
<td>Generic ListTerm</td>
</tr>
<tr>
<td>DLA_C_GEN_TABLE</td>
<td>Generic Table</td>
</tr>
<tr>
<td>Entity type</td>
<td>Meaning</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>DLA_C_GEN_ROW</td>
<td>Generic Row</td>
</tr>
<tr>
<td>DLA_C_GEN_TABLE_COMPONENT</td>
<td>Generic TableComponent</td>
</tr>
<tr>
<td>DLA_C_GEN_ROW_COMPONENT</td>
<td>Generic RowComponent</td>
</tr>
<tr>
<td>DLA_C_GEN_FORM</td>
<td>Generic Form</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_ELEMENT</td>
<td>Generic EntryElement</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_GROUP</td>
<td>Generic EntryGroup</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_TEXT</td>
<td>Generic EntryText</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_RASTER</td>
<td>Generic EntryRaster</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_GEOMETRIC</td>
<td>Generic EntryGeometric</td>
</tr>
<tr>
<td>DLA_C_GEN_LAY_ROOT</td>
<td>Generic DocumentLayoutRoot</td>
</tr>
<tr>
<td>DLA_C_GEN_PAGESET</td>
<td>Generic PageSet</td>
</tr>
<tr>
<td>DLA_C_GEN_PAGE</td>
<td>Generic Page</td>
</tr>
<tr>
<td>DLA_C_GEN_RECTO_PAGE</td>
<td>Generic RectoPage</td>
</tr>
<tr>
<td>DLA_C_GEN_VERSO_PAGE</td>
<td>Generic VersoPage</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_HEADER</td>
<td>Generic CompositeHeader</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_FOOTER</td>
<td>Generic CompositeFooter</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_BODY_FIX</td>
<td>Generic FixedCompositeBody</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_BODY_VAR</td>
<td>Generic VariableCompositeBody</td>
</tr>
<tr>
<td>DLA_C_GEN_COL_FIXED</td>
<td>Generic ColumnFixed</td>
</tr>
<tr>
<td>DLA_C_GEN_COL_VARIABLE</td>
<td>Generic ColumnVariable</td>
</tr>
<tr>
<td>DLA_C_GEN_SNAKING_COLUMN</td>
<td>Generic SnakingColumns</td>
</tr>
<tr>
<td>DLA_C_GEN_SYNC_COLUMN</td>
<td>Generic SynchronizedColumns</td>
</tr>
<tr>
<td>DLA_C_GEN_BASIC_FLOAT</td>
<td>Generic BasicFloat</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_FLOAT</td>
<td>Generic CompositeFloat</td>
</tr>
<tr>
<td>DLA_C_GEN_BASIC_COLUMN</td>
<td>Generic BasicColumn</td>
</tr>
<tr>
<td>DLA_C_GEN_FOOTNOTE_AREA</td>
<td>Generic FootnoteArea</td>
</tr>
<tr>
<td>DLA_C_GEN_ARRNGD_CONT_FIXED</td>
<td>Generic ArrangedContentFixed</td>
</tr>
<tr>
<td>DLA_C_GEN_ARRNGD_CONT_VAR</td>
<td>Generic ArrangedContentVariable</td>
</tr>
<tr>
<td>DLA_C_GEN_SRCD_CONT_FIXED</td>
<td>Generic SourcedContentFixed</td>
</tr>
<tr>
<td>DLA_C_GEN_SRCD_CONT_VAR</td>
<td>Generic SourcedContentVariable</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_FIXTURE_VAR</td>
<td>Generic CompositeFixtureVariable</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_FIXTURE_FIXED</td>
<td>Generic CompositeFixtureFixed</td>
</tr>
<tr>
<td>DLA_C_GEN_BASIC_FIXTURE</td>
<td>Generic BasicFixture</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_COL_FIXED</td>
<td>Generic CompositeColumnFixed</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_COL_VARIABLE</td>
<td>Generic CompositeColumnVariable</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_COMMON</td>
<td>Generic CompositeCommon</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_ARTWORK</td>
<td>Generic CompositeArtwork</td>
</tr>
<tr>
<td>Entity type</td>
<td>Meaning</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>DLA_C_GEN_BASIC_HEADER</td>
<td>Generic BasicHeader</td>
</tr>
<tr>
<td>DLA_C_GEN_BASIC/footer</td>
<td>Generic BasicFooter</td>
</tr>
<tr>
<td>DLA_C_GEN_BASIC_BODY</td>
<td>Generic BasicBody</td>
</tr>
<tr>
<td>DLA_C_GEN_GENERIC_BLOCK</td>
<td>Generic GenericBlock</td>
</tr>
<tr>
<td>DLA_C_GEN_FORM_AREA</td>
<td>Generic FormArea</td>
</tr>
<tr>
<td>DLA_C_GEN_ENTRY_GROUP_AREA</td>
<td>Generic EntryGroupArea</td>
</tr>
<tr>
<td>DLA_C_GEN_TABLE_AREA</td>
<td>Generic TableArea</td>
</tr>
<tr>
<td>DLA_C_GEN_TABLE_HEADER</td>
<td>Generic TableHeader</td>
</tr>
<tr>
<td>DLA_C_GEN_TABLE_LABEL</td>
<td>Generic TableLabel</td>
</tr>
<tr>
<td>DLA_C_GEN_COMP_TABLE_LABEL</td>
<td>Generic CompositeTableLabel</td>
</tr>
<tr>
<td>DLA_C_GEN_LABEL_COMPONENT</td>
<td>Generic LabelComponent</td>
</tr>
<tr>
<td>DLA_C_GEN_ROW_AREA</td>
<td>Generic RowArea</td>
</tr>
<tr>
<td>DLA_C_GEN_CELL</td>
<td>Generic Cell</td>
</tr>
<tr>
<td>DLA_C_GEN_SUBROW_GROUP</td>
<td>Generic SubRowGroup</td>
</tr>
<tr>
<td>DLA_C_GEN_SUBROW</td>
<td>Generic SubRow</td>
</tr>
<tr>
<td>DLA_C_GEN_TABLE_LABEL_CONTENT</td>
<td>Generic TableLabelContent</td>
</tr>
<tr>
<td>DLA_C_GEN_FORM_ENTRY_AREA</td>
<td>Generic FormEntryArea</td>
</tr>
</tbody>
</table>

**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.
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
- BasicFixture
- 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 dla_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_NOTSAME_TYPE]
The types of the frames given must be identical for this function.
[DLA_E_ALL_COLS_COMPOSITE]
In the FOD36 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.
**dla_move_to_child_spec_entity**

Returns the handle of a child of a specific logical or layout entity.

**C Synopsis**

```c
dla_status   dla_move_to_child_spec_entity(
    dla_document_handle   document_handle,
    dla_entity_handle    current_entity,
    dla_child_type       child_type,
    dla_entity_handle    *child_entity);
```

**Description**

The `dla_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:

<table>
<thead>
<tr>
<th>Child type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_CHILD_FIRST</td>
<td>Move to the first child.</td>
</tr>
<tr>
<td>DLA_C_CHILD_LAST</td>
<td>Move to the last child.</td>
</tr>
</tbody>
</table>

**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**

```c
dia_status  dia_move_to_parent_spec_entity(
    dla_document_handle  document_handle,
    dla_entity_handle    current_entity,
    dla_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.
**dla_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**

```c
dla_status   dla_move_to_root_entity(
    dla_document_handle  document_handle,
    dla_structure_type   structure_type,
    dla_entity_handle    *root_entity);
```

**Description**

The `dla_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:
  - `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.
dla_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.
**dla_move_to_sibling_spec_entity**

Returns the handle of a sibling entity.

**C Synopsis**

```c
dda_status    dla_move_to_sibling_spec_entity(       
    dla_document_handle document_handle, 
    dla_entity_handle    current_entity, 
    dla_sibling_type     sibling_type, 
    dla_entity_handle    *sibling_entity);
```

**Description**

The `dla_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:

<table>
<thead>
<tr>
<th>Sibling type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_SIB_FIRST</td>
<td>Handle of the first sibling</td>
</tr>
<tr>
<td>DLA_C_SIB_LAST</td>
<td>Handle of the last sibling</td>
</tr>
<tr>
<td>DLA_C_SIB_PREV</td>
<td>Handle of the previous sibling</td>
</tr>
<tr>
<td>DLA_C_SIB_NEXT</td>
<td>Handle of the next sibling</td>
</tr>
</tbody>
</table>

**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.
Results

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

Errors

[DLA_EENTITY_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.
dla_read_document

Reads an ODIF data stream and creates the corresponding ODA document.

C Synopsis

dla_status       dla_read_document(
    dla_toolkit_handle toolkit_handle,
    char             *file_specification,
    dla_length       file_specification_length,
    dla_document_options document_options,
    char             *store_file,
    dla_length       store_file_length,
    dla_dap_level    *dap_level,
    dla_document_handle *document_handle);

Description

The dla_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.
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.

<table>
<thead>
<tr>
<th>DAP level</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_DAP_LEVEL_1</td>
<td>Document at DAP level 1</td>
</tr>
<tr>
<td>DLA_C_DAP_LEVEL_2</td>
<td>Document at DAP level 2</td>
</tr>
<tr>
<td>DLA_C_DAP_LEVEL_3</td>
<td>Document at DAP level 3</td>
</tr>
</tbody>
</table>

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.
dl_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

```c
#include "dl.h"

dl_status  dla_read_generic_doc(  
    dl_document_handle   document_handle,  
    char                *file_specification,  
    dl_length           file_specification_length,  
    dl_generic_doc_type  generic_doc_type,  
    dl_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 fulfill; one of the following:

<table>
<thead>
<tr>
<th>Type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_EXTERNAL_DOC</td>
<td>External document</td>
</tr>
<tr>
<td>DLA_CRESOURCE_DOC</td>
<td>Resource document</td>
</tr>
</tbody>
</table>

**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_LAYOUT_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_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.

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_NO_OPTIONS</td>
<td>Equivalent to DLA_C_MODE_1</td>
</tr>
<tr>
<td>DLA_C_MODE_1</td>
<td>Toolkit creates generic logical structure</td>
</tr>
<tr>
<td>DLA_C_AUTO_GEN</td>
<td>Toolkit creates generic logical structure (Equivalent to DLA_C_MODE_1)</td>
</tr>
<tr>
<td>DLA_C_MODE_2</td>
<td>Application creates generic logical structure</td>
</tr>
<tr>
<td>DLA_C_WITH_EXTERNAL</td>
<td>Application will read generic document from ODIF</td>
</tr>
<tr>
<td>DLA_C_FOR_EXTERNAL</td>
<td>Application will supply ODIF identifiers for all entities</td>
</tr>
<tr>
<td>DLA_C_FOR_RESOURCE</td>
<td>Application will supply ODIF identifiers for all entities and the resources property</td>
</tr>
</tbody>
</table>
**Results**

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

**Errors**

**[DLA_E_DOC_READONLY]**
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.
dlaregistercallbackfunction

dlaregistercallbackfunction

Specifies a function in the application's code space that is invoked periodically during
dlaraddocument and dlawritedocument.

C Synopsis

dl_status dlaregistercallbackfunction(
    dl_toolkit_handle toolkit_handle,
    dlacallback_function callback_function);

Description

The dlaregistercallbackfunction 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
dlaraddocument and dlawritedocument.

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
dlawritedocument or dlareaddocument, 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.
**dl_set_entity_handle**

Specifies a value for a property on an entity, for a property of entity handle type.

**C Synopsis**

```c
dl_set_entity_handle(dla_document_handle document_handle, 
                     dla_entity_handle entity_handle, 
                     dla_structure_handle structure_handle, 
                     dla_property_code property_code, 
                     dla_entity_handle property_entity_handle);
```

**Description**

The `dl_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_PRES_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**

```c
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:

<table>
<thead>
<tr>
<th>Index operation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_INDEX_REPLACE</td>
<td>Replace the indexed entry.</td>
</tr>
<tr>
<td>DLA_C_INDEX_INSERT</td>
<td>Insert before the indexed entry.</td>
</tr>
</tbody>
</table>

**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_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.
dla_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   dla_set_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_index_operation index_operation,
    dla_integer         integer_value);

Description

The dla_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.
Index_operation
Indicates whether the entry must be inserted before, or replace, the entry existing at the indexed position. One of the enumerated values:

<table>
<thead>
<tr>
<th>Index operation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_INDEX_REPLACE</td>
<td>Replace the indexed entry.</td>
</tr>
<tr>
<td>DLA_C_INDEX_INSERT</td>
<td>Insert before the indexed entry.</td>
</tr>
</tbody>
</table>

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_APROPERTY_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.
**dla_set_index_null_value**

Specifies the value for an indexed property of type null-valued.

**C Synopsis**

```c
dla_status  dla_set_index_null_value(
    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);
```

**Description**

The `dla_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.
**Index_operation**
Indicates whether the entry must be inserted before, or replace, the entry existing at the indexed position. One of the enumerated values:

<table>
<thead>
<tr>
<th>Index Operation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_INDEX_REPLACE</td>
<td>Replace the indexed entry.</td>
</tr>
<tr>
<td>DLA_C_INDEX_INSERT</td>
<td>Insert before the indexed entry.</td>
</tr>
</tbody>
</table>

**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_RNG]
The index is out of the range of valid indexes for 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.
dl_a_set_index_string

Specifies the value for an indexed property of type string.

C Synopsis

```c
dl_status dl_a_set_index_string(
    dl_a_document_handle document_handle,
    dl_a_entity_handle entity_handle,
    dl_a_structure_handle structure_handle,
    dl_a_property_code property_code,
    dl_a_integer index,
    dl_a_index_operation index_operation,
    dl_a_octet_string string_value,
    dl_a_length string_value_length);
```

Description

The `dl_a_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:

<table>
<thead>
<tr>
<th>Index operation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_INDEX_REPLACE</td>
<td>Replace the indexed entry.</td>
</tr>
<tr>
<td>DLA_C_INDEX_INSERT</td>
<td>Insert before the indexed entry.</td>
</tr>
</tbody>
</table>

**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.
**dia_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**

```c
dia_status     dla_set_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_index_operation    index_operation,
    dia_structure_handle   *new_structure_handle);
```

**Description**

The `dia_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:

<table>
<thead>
<tr>
<th>Index operation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_INDEX_REPLACE</td>
<td>Replace the indexed entry.</td>
</tr>
<tr>
<td>DLA_C_INDEX_INSERT</td>
<td>Insert before the indexed entry.</td>
</tr>
</tbody>
</table>

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_SHAREDPROP_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_OPERATOR]
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_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.
**dla_set_integer**

**dla_set_integer**

Specifies the value for a property of type integer.

**C Synopsis**

```c
#include <dla.h>

dla_status dla_set_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);
```

**Description**

The `dla_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_SEG_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.
dla_set_null_value

Specifies the value for a property of type null-valued.

C Synopsis

dla_status    dla_set_null_value(
    dla_document_handle  document_handle,
    dla_entity_handle    entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code    property_code);

Description

The dla_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.
**dla_set_status**

Sets the status of a property of an entity.

**C Synopsis**

```c
dla_status  dla_set_status(
    dla_document_handle  document_handle,
    dla_entity_handle    entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code    property_code,
    dla_property_status  property_status);
```

**Description**

The `dla_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 `dla_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_PGE_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.

<table>
<thead>
<tr>
<th>Status</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_PROP_NULL</td>
<td>The property status is null.</td>
</tr>
<tr>
<td>DLA_C_PROP_UNSPECIFIED</td>
<td>The property status is unspecified</td>
</tr>
</tbody>
</table>

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.
[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_string**

Specifies the value for a property of type string.

**C Synopsis**

```c
dla_status dla_set_string(
    dla_document_handle document_handle,
    dla_entity_handle entity_handle,
    dla_structure_handle structure_handle,
    dla_property_code property_code,
    dla_octet_string string_value,
    dla_length string_value_length);
```

**Description**

The `dla_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.
Results

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

Errors

**[DLA_E_PROPFIXED]**
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.
The `dla_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 BMUs.
alignment
The required alignment type. An enumerated value that is one of:

<table>
<thead>
<tr>
<th>Alignment type</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLA_C_TAB_ALIGN_START</td>
<td>Start aligned</td>
</tr>
<tr>
<td>DLA_C_TAB_ALIGN_END</td>
<td>End aligned</td>
</tr>
<tr>
<td>DLA_C_TAB_ALIGN_CENTRED</td>
<td>Center aligned</td>
</tr>
<tr>
<td>DLA_C_TAB_ALIGN_AROUND</td>
<td>Align around</td>
</tr>
</tbody>
</table>

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**

```c
dia_status    dia_term_toolkit(
    dla_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.
dlunfix_properties

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.

tentity_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

Returns the version of the DAP Level API toolkit.

C Synopsis

```c
char *dla_version(
  **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;ODAC;2` represents Release 2.0 of Version 1.0.

Arguments

`version [write]`

Received 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.
**dla_write_document**

Writes a document as an ODIF data stream.

**C Synopsis**

```c
int8_t *dla_write_document(
    dla_document_handle document_handle,
    char *file_specification,
    uint16_t file_specification_length,
    dla_document_options document_options);
```

**Description**

The `dla_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 `DLA_C_NO_OPTIONS`.

**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_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 dla_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_NOSPEC_LOG_STRUCT]
The document (including any referenced external document class) does not contain a specific logical structure. This is deemed mandatory.