NAME

xfn_attributes – an overview of XFN attribute operations

DESCRIPTION

XFN assumes the following model for attributes. A set of zero or more attributes is associated with a named object. Each attribute in the set has a unique attribute identifier, an attribute syntax and a (possibly empty) set of distinct data values. Each attribute value has an opaque data type. The attribute identifier serves as a name for the attribute. The attribute syntax indicates the how the value is encoded in the buffer.

The operations of the base attribute interface may be used to examine and modify the settings of attributes associated with existing named objects. These objects may be contexts or other types of objects. The attribute operations do not create names or remove names from contexts.

The range of support for attribute operations may vary widely. Some naming systems may not support any attribute operations. Other naming systems may only support read operations, or operations on attributes whose identifiers are in some fixed set. A naming system may limit attributes to have a single value, or may require at least one value. Some naming systems may only associate attributes with context objects, while others may allow associating attributes with non-context objects.

These are the interfaces:

#include <xfn/xfn.h>

FN_attribute_t *fn_attr_get(FN_ctx_t ****ctx*, const FN_composite_name_t ****name***,**
const FN_identifier_t ****attribute_id*, FN_status_t ****status***);**

int fn_attr_modify(FN_ctx_t ****ctx*, const FN_composite_name_t ****name***,**
unsigned int *mod_op***, const FN_attribute_t ****attr***, FN_status_t ****status***);**

FN_attrset_t *fn_attr_get_ids(FN_ctx_t ****ctx*, const FN_composite_name_t ****name***,**
FN_status_t ****status*);**

FN_valuelist_t *fn_attr_get_values(FN_ctx_t ****ctx*, const FN_composite_name_t ****name***,**
const FN_identifier_t ****attribute_id*, FN_status_t ****status***);**

FN_attrvalue_t *fn_valuelist_next(FN_valuelist_t ****vl*, FN_identifier_t *****attr_syntax***,**
FN_status_t ****status*);**

void fn_valuelist_destroy(FN_valuelist_t ****vl*, FN_status_t ****status***);**

FN_multigetlist_t *fn_attr_multi_get(FN_ctx_t ****ctx*,**
const FN_composite_name_t ****name*, const FN_attrset_t ****attr_ids***,**
FN_status_t ****status*);**

FN_attribute_t *fn_multigetlist_next(FN_multigetlist_t ****ml,* **FN_status_t ****status***);**

void fn_multigetlist_destroy(FN_multigetlist_t ****ml*, FN_status_t ****status***);**

int fn_attr_multi_modify(FN_ctx_t ****ctx*, const FN_composite_name_t ****name***,**
const FN_attrmodlist_t ****mods*, FN_status_t ****status***,**
FN_attrmodlist_t *****unexecuted_mods*);**

FN_attrset_t *fn_ctx_get_syntax_attrs(FN_ctx_t **ctx*,
const FN_composite_name_t **name*, **FN_status_t** **status***);**

The following describes briefly the operations in the base attribute interface. Detailed descriptions are given in the respective reference manual pages for these operations.

fn_attr_get( ) returns the attribute identified. **fn_attr_modify( )** modifies the attribute identified as described by *mod_op.*

fn_attr_get_ids( ) returns the identifiers of the attributes of the named object.

fn_attr_get_values( ) and its set of related operations are used for returning the individual values of an attribute.

fn_attr_multi_get( ) and its set of related operations are used for returning the requested attributes associated with the named object. **fn_attr_multi_modify( )** modifies multiple attributes associated with the named object in a single invocation.

fn_ctx_get_syntax_attrs( ) returns the syntax attributes associated with the named context.

ERRORS

status is set as described in **FN_status_t**(3N) and **xfn_status_codes**(3N). The following status codes are of special relevance to attribute operations:

FN_E_ATTR_VALUE_REQUIRED
The operation attempted to create an attribute without a value, and the specific naming system does not allow this.

FN_E_ATTR_NO_PERMISSION
The caller did not have permission to perform the attempted attribute operation.

FN_E_INSUFFICIENT_RESOURCES
There is insufficient resources to retrieve the requested attribute(s).

FN_E_INVALID_ATTR_IDENTIFIER
The attribute identifier was not in a format acceptable to the naming system, or its contents was not valid for the format specified for the identifier.

FN_E_INVALID_ATTR_VALUE
One of the values supplied was not in the appropriate form for the given attribute.

FN_E_NO_SUCH_ATTRIBUTE
The object did not have an attribute with the given identifier.

FN_E_TOO_MANY_ATTR_VALUES
The operation attempted to associate more values with an attribute than the naming system supported.

APPLICATION USAGE

Except for **fn_ctx_get_syntax_attrs( ) ,** an attribute operation using a composite name is not necessarily equivalent to an independent **fn_ctx_lookup( )** operation followed by an attribute operation in which the caller supplies the resulting reference and an empty name. This is because there are a range of attribute models in which an attribute is associated with a name in a context, or an attribute is associated with the object named, or both. XFN accommodates all of these alternatives. Invoking an attribute operation using the target context and the terminal atomic name accesses either the attributes that are associated with the target name or target named object — this is dependent on the underlying attribute model. This document uses the term *attributes associated with a named object* to refer to all of these cases.

XFN specifies no guarantees about the relationship between the attributes and the reference associated with a given name. Some naming systems may store the reference bound to a name in one or more attributes associated with a name. Attribute operations might affect the information used to construct a reference.

To avoid undefined results, programmers must use the operations in the context interface and not attribute operations when the intention is to manipulate a reference. Programmers should avoid the use of specific knowledge about how an XFN context implementation over a particular naming system constructs references.

SEE ALSO

FN_attribute_t(3N), **FN_attrvalue_t**(3N), **FN_attrset_t**(3N), **FN_composite_name_t**(3N), **FN_ctx_t**(3N), **FN_identifier_t**(3N), **FN_status_t**(3N), **fn_attr_get**(3N), **fn_attr_get_ids**(3N), **fn_attr_get_values**(3N), **fn_attr_modify**(3N), **fn_attr_multi_get**(3N), **fn_attr_multi_modify**(3N), **fn_ctx_get_syntax_attrs**(3N), **fn_ctx_lookup**(3N), **xfn_status_codes**(3N), **xfn**(3N)