NAME

fn_attr_multi_get, FN_multigetlist_t, fn_multigetlist_next, fn_multigetlist_destroy – return multiple attributes associated with named object

SYNOPSIS

cc [*flag* ...] *file* ... **–lxfn** [*library* ...]

#include <xfn/xfn.h>

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*);

MT-LEVEL

Safe.

DESCRIPTION

This set of operations returns one or more attributes associated with the object named by *name* relative to the context *ctx*. If *name* is empty, the attributes associated with *ctx* are returned.

The attributes returned are those specified in *attr_ids.* If the value of *attr_ids* is **0**, all attributes associated with the named object are returned. Any attribute values in *attr_ids* provided by the caller are ignored; only the attribute identifiers are relevant for this operation. Each attribute (identifier, syntax, values) is returned one at a time using an enumeration scheme similar to that for listing a context.

fn_attr_multi_get( ) initiates the enumeration process. It returns a handle to an **FN_multigetlist_t** object that can be used for the enumeration.

The operation **fn_multigetlist_next( )** returns a new **FN_attribute_t** object containing the next attribute (identifiers, syntaxes, and values) requested and updates *ml* to indicate the state of the enumeration.

The operation **fn_multigetlist_destroy( )** releases the resources used during the enumeration. It may be invoked before the enumeration has completed to terminate the enumeration.

RETURN VALUE

fn_attr_multi_get( ) returns a pointer to an **FN_multigetlist_t** object if the enumeration has been initiated successfully; a **NULL** pointer is returned if it failed.

fn_multigetlist_next( ) returns a pointer to an **FN_attribute_t** object if an attribute was returned, a **NULL** pointer (**0**) if no attribute was returned.

In the case of a failure, these operations set *status* to indicate the nature of the failure.

ERRORS

Each call to **fn_multigetlist_next( )** sets *status* as follows:

FN_SUCCESS
If an attribute was returned, there are more attributes to be enumerated. If no attribute was returned, the enumeration has completed successfully.

FN_E_ATTR_NO_PERMISSION

The caller did not have permission to read this attribute.

FN_E_INSUFFICIENT_RESOURCES
Insufficient resources are available to return the attribute's values.

FN_E_INVALID_ATTR_IDENTIFIER
This 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_ENUM_HANDLE
(No attribute should be returned with this status code). The given enumeration handle is not valid. Possible reasons could be that the handle was from another enumeration, or the object being processed no longer accepts the handle (due to such events as handle expiration or updates to the object's attribute set).

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

FN_E_PARTIAL_RESULT
(No attribute should be returned with this status code). The enumeration is not yet complete but cannot be continued.

For **FN_E_ATTR_NO_PERMISSION, FN_E_INVALID_ATTR_IDENTIFIER, FN_E_INSUFFICIENT_RESOURCES,** or **FN_E_NO_SUCH_ATTRIBUTE,** the returned attribute contains only the attribute identifier (no value or syntax). For these four status codes and **FN_SUCCESS** (when an attribute was returned), **fn_multigetlist_next( )** can be called again to return another attribute. All other status codes indicate that no more attributes can be returned by **fn_multigetlist_next( ).**

Other status codes, such as **FN_E_COMMUNICATION_FAILURE,** are also possible, in which case, no attribute is returned. In such cases, *status* is set as described in **FN_status_t**(3N) and **xfn_status_codes**(3N).

EXAMPLES

The following code fragment illustrates to obtain all attributes associated with a given name using the **fn_attr_multi_get( )** operations.

```
/* list all attributes associated with given name */

extern FN_string_t *input_string;
FN_ctx_t  *ctx;
FN_composite_name_t *target_name = fn_composite_name_from_string(input_string);
FN_multigetlist_t *ml;
FN_status_t  *status = fn_status_create();
FN_attribute_t *attr;
int done = 0;

ctx = fn_ctx_handle_from_initial(status);
/* error checking on 'status' */

/* attr_ids == 0 indicates all attributes are to be returned */
if ((ml=fn_attr_multi_get(ctx, target_name, 0, status)) == 0) {
```

```
        /* report 'status' and exit */
}

while ((attr=fn_multigetlist_next(ml, status)) && !done) {
        switch (fn_status_code(status)) {
        case FN_SUCCESS:
                /* do something with 'attr' */
                break;
        case FN_E_ATTR_NO_PERMISSION:
        case FN_E_ATTR_INVALID_ATTR_IDENTIFIER:
        case FN_E_NO_SUCH_ATTRIBUTE:
                /* report error using identifier in 'attr' */
                break;
        default:
                /* other error handling */
                done = 1;
        }
        if (attr)
                fn_attribute_destroy(attr);
}

/* check 'status' for reason for end of enumeration and report if necessary */

/* clean up */
fn_multigetlist_destroy(ml, status);

/* report 'status' */
}
```

APPLICATION USAGE

Implementations are not required to return all attributes requested by *attr_ids*. Some may choose to return only the attributes found successfully, followed by a status of **FN_E_PARTIAL_RESULT;** such implementations may not necessarily return attributes identifying those that could not be read. Implementations are not required to return the attributes in any order.

There may be a relationship between the *ctx* argument supplied to **fn_attr_multi_get( )** and the **FN_multigetlist_t** object it returns. For example, some implementations may store the context handle *ctx* within the **FN_multigetlist_t** object for subsequent **fn_multigetlist_next( )** calls. In general, a **fn_ctx_handle_destroy( )** should not be invoked on *ctx* until the enumeration has terminated.

SEE ALSO

FN_attrset_t(3N), **FN_attribute_t**(3N), **FN_composite_name_t**(3N), **FN_ctx_t**(3N), **FN_identifier_t**(3N), **FN_status_t**(3N), **fn_attr_get**(3N), **fn_ctx_list_names**(3N), **xfn_attributes**(3N), **xfn_status_codes**(3N), **xfn**(3N)