NAME

xfn_links – XFN links: an overview of XFN links

DESCRIPTION

An *XFN link* is a special form of reference that contains a composite name, the *link name*, and that may be bound to an atomic name in an XFN context. Because the link name is a composite name, it may span multiple namespaces.

Normal resolution of names in context operations always follows XFN links. If the first composite name component of the link name is the atomic name ".", the link name is resolved relative to the same context in which the link is bound, otherwise, the link name is resolved relative to the XFN Initial Context of the client. The link name may itself cause resolution to pass through other XFN links. This gives rise to the possibility of a cycle of links whose resolution could not terminate normally. As a simple means to avoid such non-terminating resolutions, implementations may define limits on the number of XFN links that may be resolved in any single operation invoked by the caller.

These are the interfaces:

#include <xfn/xfn.h>

FN_ref_t *fn_ref_create_link(const FN_composite_name_t **link_name***);**

int fn_ref_is_link(const FN_ref_t **ref***);**

FN_composite_name_t *fn_ref_link_name(const FN_ref_t **link_ref***);**

FN_ref_t *fn_ctx_lookup_link(FN_ctx_t **ctx***, const FN_composite_name_t** **name***,**
FN_status_t **status***);**

unsigned int fn_status_link_code(const FN_status_t **stat***);**

const FN_composite_name_t *fn_status_link_remaining_name(
const FN_status_t **stat***);**

const FN_composite_name_t *fn_status_link_resolved_name(const FN_status_t **stat***);**

const FN_ref_t *fn_status_link_resolved_ref(const FN_status_t **stat***);**

int fn_status_set_link_code(FN_status_t **stat***, unsigned int** *code***);**

int fn_status_set_link_remaining_name(FN_status_t **stat***,**
const FN_composite_name_t **name***);**

int fn_status_set_link_resolved_name(FN_status_t **stat***,**
const FN_composite_name_t **name***);**

int fn_status_set_link_resolved_ref(FN_status_t **stat***, const FN_ref_t** **ref***);**

Links are bound to names using the normal **fn_ctx_bind( )** and unbound using the normal **fn_ctx_unbind( )** operation. The operation **fn_ref_create_link( )** is provided for constructing a link reference from a composite name. Since normal resolution always follows links, a separate operation, **fn_ctx_lookup_link( )** is provided to lookup the link itself.

In the case that an error occurred while resolving an XFN link, the status object set by the operation contains additional information about that error and sets the corresponding link status fields using **fn_status_set_link_code( )**, **fn_status_set_link_remaining_name( )**, **fn_status_set_link_resolved_name( )** and

fn_status_set_link_resolved_ref( ). The link status fields can be retrieved using **fn_status_link_code( )**, **fn_status_link_remaining_name( )**, **fn_status_link_resolved_name( )** and **fn_status_link_resolved_ref( )**.

ERRORS

The following status codes are of special relevance when performing operations involving XFN links:

FN_E_LINK_ERROR
There was an error encountered resolving an XFN link encountered during resolution of the supplied name. Check the link part of the status object to determine cause of the link error.

FN_E_LINK_LOOP_LIMIT
A non-terminating loop (cycle) in the resolution can arise due to XFN links encountered during the resolution of a composite name. This code indicates either the definite detection of such a cycle, or that resolution exceeded an implementation-defined limit on the number of XFN links allowed for a single operation invoked by the caller.

FN_E_MALFORMED_LINK
A malformed link reference was encountered. For the **fn_ctx_lookup_link( )** operation, the name supplied resolved to a reference that was not a link.

APPLICATION USAGE

For the **fn_ctx_bind( )**, **fn_ctx_unbind( )**, **fn_ctx_rename( )**, **fn_ctx_lookup_link( )**, **fn_ctx_create_subcontext( )** and **fn_ctx_destroy_subcontext( )** operations, resolution of the given name continues to the target context — that named by all but the terminal atomic part of the given name; the terminal atomic name is not resolved. Consequently, for operations that involve unbinding the terminal atomic part such as **fn_ctx_unbind( ) ,** if the terminal atomic name is bound to a link, the link is not followed and the link itself is unbound from the terminal atomic name.

Many naming systems support a native notion of link that may be used within the naming system itself. XFN does not determine whether there is any relationship between such native links and XFN links.

SEE ALSO

FN_composite_name_t(3N), **FN_ref_t**(3N), **FN_status_t**(3N), **fn_ctx_bind**(3N), **fn_ctx_destroy_subcontext**(3N), **fn_ctx_lookup**(3N), **fn_ctx_lookup_link**(3N), **fn_ctx_rename**(3N), **fn_ctx_unbind**(3N), **xfn_status_codes**(3N), **xfn**(3N)