NAME

xfn_compound_names – XFN compound syntax: an overview of XFN model for compound name parsing

DESCRIPTION

Each naming system in an XFN federation has a naming convention. XFN defines a standard model of expressing compound name syntax that covers a large number of specific name syntaxes and is expressed in terms of syntax properties of the naming convention.

The model uses the attributes in the following table to describe properties of the syntax. Unless otherwise qualified, these syntax attributes have attribute identifiers that use the **FN_ID_STRING** format. A context that supports the XFN standard syntax model has an attribute set containing the **fn_syntax_type** (**FN_ID_STRING** format) attribute with the value "standard" (ASCII attribute syntax).

These are the interfaces:

#include <xfn/xfn.h>

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

FN_compound_name_t *fn_compound_name_from_syntax_attrs(
    const FN_attrset_t **aset*, **const FN_string_t** **name*, **FN_status_t** **status***);**

fn_syntax_type
Its value is the ASCII string "standard" if the context supports the XFN standard syntax model. Its value is an implementation-specific value if another syntax model is supported.

fn_std_syntax_direction
Its value is an ASCII string, one of "left_to_right", "right_to_left", or "flat". This determines whether the order of components in a compound name string goes from left to right, right to left, or whether the namespace is flat (in other words, not hierarchical — all names are atomic).

fn_std_syntax_separator
Its value is the separator string for this name syntax. This attribute is required unless the **fn_std_syntax_direction** is "flat".

fn_std_syntax_escape
If present, its value is the escape string for this name syntax.

fn_std_syntax_case_insensitive
If this attribute is present, it indicates that names that differ only in case are considered identical. If this attribute is absent, it indicates that case is significant. If a value is present, it is ignored.

fn_std_syntax_begin_quote
If present, its value is the begin-quote string for this syntax. There can be multiple values for this attribute.

fn_std_syntax_end_quote
If present, its value is the end-quote string for this syntax. There can be multiple values for this attribute.

fn_std_syntax_ava_separator
If present, its value is the attribute value assertion separator string for this syntax.

fn_std_syntax_typeval_separator
If present, its value is the attribute type-value separator string for this syntax.

fn_std_syntax_code_sets
If present, its value identifies the code sets of the string representation for this syntax. Its value consists of a structure containing an array of code sets supported by the context; the first member of the array is the preferred code set of the context. The values for the code sets are defined in the X/Open code set registry. If this attribute is not present, or if the value is empty, the default code set is **ISO 646** (same encoding as ASCII).

fn_std_syntax_locale_info
If present, identifies locale information, such as character set information, of the string representation for this syntax. The interpretation of its value is implementation-dependent.

The XFN standard syntax attributes are interpreted according to the following rules:

1. In a string without quotes or escapes, any instance of the separator string delimits two atomic names.
2. A separator, quotation or escape string is escaped if preceded immediately (on the left) by the escape string.
3. A non-escaped begin-quote which precedes a component must be matched by a non-escaped end-quote at the end of the component. Quotes embedded in non-quoted names are treated as simple characters and do not need to be matched. An unmatched quotation fails with the status code **FN_E_ILLEGAL_NAME.**
4. If there are multiple values for begin-quote and end-quote, a specific begin-quote value must be matched with its corresponding end-quote value.
5. When the separator appears between a (non-escaped) begin quote and the end quote, it is ignored.
6. When the separator is escaped, it is ignored. An escaped begin-quote or end-quote string is not treated as a quotation mark. An escaped escape string is not treated as an escape string.
7. A non-escaped escape string appearing within quotes is interpreted as an escape string. This can be used to embed an end-quote within a quoted string.

After constructing a compound name from a string, the resulting component atoms have one level of escape strings and quotations interpreted and consumed.

fn_ctx_get_syntax_attrs( ) is used to obtain the syntax attributes associated with a context.

fn_compound_name_from_syntax( ) is used to construct a compound name object using the string form of the name and the syntax attributes of the name.

ERRORS

FN_E_ILLEGAL_NAME
The name supplied to the operation was not a well-formed component according to the name syntax of the context.

FN_E_INCOMPATIBLE_CODE_SETS
Code set mismatches that occur during the construction of the compound name's string form are resolved in an implementation-dependent way. When an implementation discovers that a compound name has components with incompatible code sets, it returns the error code **FN_E_INCOMPATIBLE_CODE_SETS.**

FN_E_INVALID_SYNTAX_ATTRS
The syntax attributes supplied are invalid or insufficient to fully specify the syntax.

FN_E_SYNTAX_NOT_SUPPORTED
The syntax specified is not supported.

APPLICATION USAGE

Most applications treat names as opaque data and hence, the majority of clients of the XFN interface will not need to parse compound names from specific naming systems. Some applications, however, such as browsers, need such capabilities. These applications would use **fn_ctx_get_syntax_attrs( )** to obtain the syntax related attributes of a context and, if the context uses the XFN standard syntax model, it would examine these attributes to determine the name syntax of the context.

SEE ALSO

FN_attribute_t(3N), **FN_attrset_t**(3N), **FN_compound_name_t**(3N), **FN_identifier_t**(3N), **FN_string_t**(3N)**, fn_ctx_get_syntax_attrs**(3N), **xfn**(3N)