Reference number of working document: **NEAOSS/WG3 N111, WG3/SWG1 N096**

Date: 2007-09-12

Reference number of document: **NEAOSS/WD 001-2**

Committee identification: NEAOSS/WG 3

Secretariat: Shizuo Goto
ProjectEditor: Hiroshi Miura

Draft recommendation of Information Technology - the Specification of IM engine Service Provider Interface

Warning

This document is not a NEAOSS official document. It is distributed for review and comment. It is subject to change without notice and may not be referred to as a standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: Draft recommendation
Document subtype:
Document stage: (18) Second draft for Candidate
Document language: E

Copyright notice

Contents

Page

Foreword

NEAOSS Forum (Northeast Asia Open Source Software promotion Forum) was formed by China,Korea and Japan governments and regional organizations for OSS promotion; China OSS Promotion Union, Korea OSS Promotion Forum and Japan OSS Promotion Forum. The Forum intends to promote Open Source Software in the northeast Asia area.

NEAOSS Forum formed “WG3:Standardization and Certification Study” in order to study OpenSource Software standardization and certification in Jul. 2004. The Free Standards Group also takes part in the WG3's work as liaison.

NEAOSS Forum WG3 formed subsidiary group:SWG1 in Dec. 2004. It addresses to study Input Method engine Service Provider Interface that can be used worldwide and propose the specification to ISO/IEC JTC 1 through the Free Standards Group, in order to establish a single world wide standard for interface among Input Method engines, User Interface Components and Input Method framework.The establishment of the standard is to enable Input Method engines to be bound to Input Method framework on demand.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. NEAOSS shall not be held responsible for identifying any or all such patent rights.

NEAOSS/WD 001-2 was prepared by NEAOSS Working Group 3 , *Study of Standardization and certification*.

Introduction

IM engine SPI is designated to serve standard interface to IM engine. Recently technology on Desktop OS is evolved and started several IM framework project on open source software world.

This may helps developer who want to develop desktop environment widget, IM framework and IM engine for every languages. It may also helps graphical system developer such as X window system from a point of integration and abstraction on 'input'.

<blank>

Draft recommendation of Information Technology
- the Specification of IM engine Service Provider Interface

1 Scope

This specification establishes the specifications of Interface specifications standard (here in after call: IMEngine SPI standard) among Input Method Engines, User Interface Components and Input Method Framework.

2 Conformance

This specification has the following 3 conformance, i.e., Input Method Framework conformance, Input Method Engine conformance and Input Method User Interface Component conformance.

In this specification, “shall” is to be interpreted as a requirement on an implementation of one of Input Method Framework, Input Method Engine or Input Method User Interface Component; conversely, “shall not” is to be interpreted as a prohibition.

If a “shall” or “shall not” requirement that appears outside of a constraint is violated, the behavior is undefined. Undefined behavior is otherwise indicated in this specification by the words “undefined behavior” or by the omission of any explicit definition of behavior. There is no difference in emphasis among these three; they all describe “behavior that is undefined”.

2.1 Input Method Framework conformance

A conforming implementation of Input Method Framework shall meet all of the following criteria:

The Input Method Framework shall support all functions and facilities defined clause 6 through 7 of this specification. The interfaces shall support the functional behavior described herein.

The Input Method Framework may provide additional functions or facilities not required by this specification. Non-standard extensions of the functions, or facilities specified in this specification should be identified as such in the Input Method Framework documentation. Non-standard extensions, when used, may change the behavior of functions, or facilities defined by this specification.

2.2 Input Method Engine conformance

The Input Method Engine shall support all functions and facilities defined clause 6.1 through 6.7 of this specification. The interfaces shall support the functional behavior described herein.

The Input Method Engine may support one or more optional functions or facilities in this specification. When an implementation of Input Method Engine claims that an option is supported, all of its constituent parts shall be provided.

The Input Method Engine may provide additional functions or facilities not required by this specification. Non-standard extensions of the functions, or facilities specified in this specification should be identified as such in the Input Method Engine documentation. Non-standard extensions, when used, may change the behavior of functions,or facilities defined by this specification. The behavior of Input Method Engine is undefined when a function that is not supported by the Input Method Engine is invoked by the Input Method Framework.

2.3 Input Method User Interface Component conformance

The Input Method User Interface Component shall support all functions and facilities defined clause 6.1 through 6.7 of this specification. The interfaces shall support the functional behavior described herein. The Input Method User Interface Component may support one or more optional functions or facilities in this specification. When an implementation of Input Method User Interface Component claims that an option is supported, all of its constituent parts shall be provided.

The Input Method User Interface Component may provide additional functions or facilities not required by this specification. Non-standard extensions of the functions, or facilities specified in this specification should be identified as such in the Input Method User Interface Component documentation. Non-standard extensions, when used, may change the behavior of functions,or facilities defined by this specification

The behavior of Input Method User Interface Component is undefined when a function that is not supported by the Input Method User Interface Component is invoked by the Input Method Framework.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[Unicode]

ISO/IEC 10646:2003 Universal MultipleOctet Coded Character Set (UCS) ISO/IEC TR 15285 An operational model for characters and glyphs ISO/IEC JTC1/SC2/WG2 http://www.open-std.org/JTC1/SC2/ Note: ISO/IEC 10646 is also known as the Unicode Standard which is freely available in the Unicode Consortium web site. http://www.unicode.org/standard/standard.html

[ISO C99]

ISO/IEC 9899:1999 Programming Language C ISO/IEC JTC1/SC22/WG14 http://www.openstd.org/JTC1/SC22/WG14/www/standards

[ISO 639]

ISO 639:Codes for the representation of name of languages

[ISO 3166-1]

ISO 3166-1:1997 Codes for the representaion of names of countries and their subdivisions -- Part1: Country codes

[ISO646]

ISO/IEC 646:1991 ISO 7-bit coded character set for information interchange

[SRGB]

IEC 61966-2-1:1999 Multimedia systems and equipment - Colour measurement and management - Part 2-1: Colour management - Default RGB colour space - sRGB. IEC 61966-2-1 (1999-10). ISBN: 2-8318-4989-6 ICS codes: 33.160.60, 37.080 TC 100 51 pp. URL: http://www.iec.ch/nr1899.htm

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
Input data
A data generated by input device such as keyboard, pointing device and microphone.

4.2
IM engine
An implementation of one or more input methods logics.

4.3
IM logic
A logic implemented in IM engines, such as Chewing or Pinyin methods implemented in a Chinese IM engine. IM logics have a functionality that it takes input and compose text.

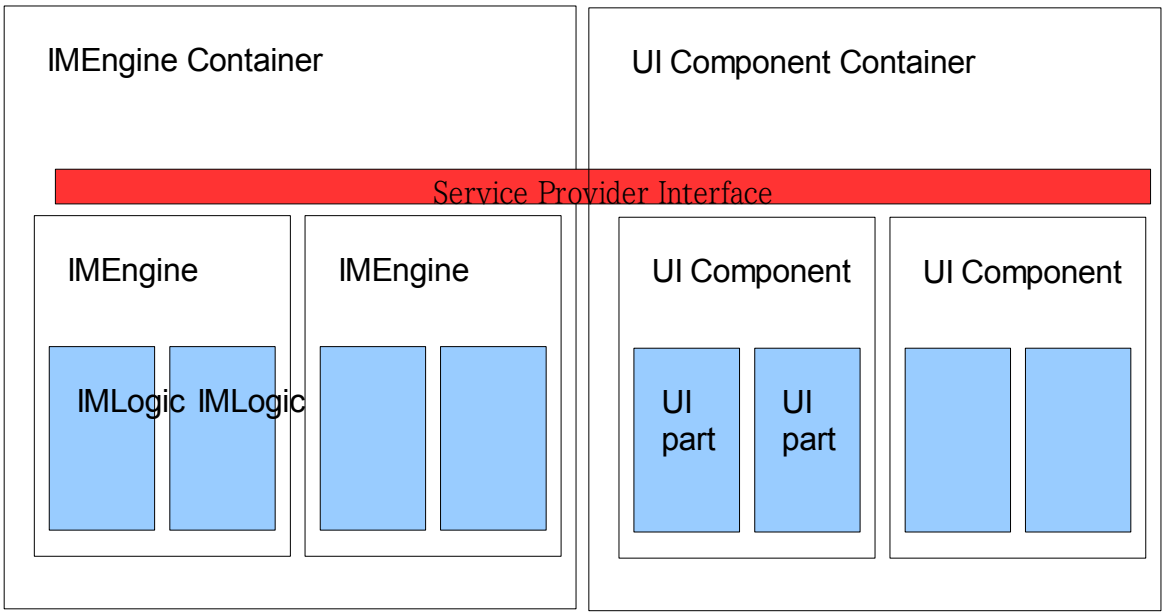
4.4
User Interface Component
An externalized component which handles the user interface related operations on behalf of IM engine, as if it is an extended part of IM engine. User Interface Components and IM engines can be provided and used independently.

4.5
IM framework
A framework which manages IM engines and User Interface Components.

4.6
IM engine SPI
An interface among IM engines, User Interface Components and IM framework.

4.7
IMComponent
An part of implementation which provide functionalities such as UI Component and IM Engine Component.

4.8
IMEngineContainer
An implementation of a kind of IMComponent which provide Service Provider Interface to IM Engine. IM engine Container shall load IM engines and initilize these engines. IM engine Container may be provided as a part of IM framework.



Drawing 1: Relation between IMEngine Container and Engine

4.9
User Interface component container
An implementation of a kind of IMComponent which provide Service Provider Interface to User Interface Component. UI component container shall load UI Component modules and initialize these components. User Interface component container may be provided as a part of IM framework.

4.10
IMEngineModule
module which implement IM engine logics. IMEngineModule may have multiple IM engine logics. IM engine Module may be loaded by IM engine Container. Module should provide internal information to IM engine Container, such as a module name, lanugages that logics provide and character encodings that logics uses.

4.11
Session Identifier
A per-IM Framework unique, non-negative integer used to identify an open session for the purpose of session access. The value of a session identifier is from zero to {XXXX}. An IM Framework can have no more than {XXXX} session identifiers open simultaneously.

4.12
Open Session Description
A record of how an IM Component include IM Engine, or group of IM Components is accessing a session. Each Session Identifier refer to exactly one open session description, but an open session description can be referred to by more than one session identifier. The attributes of an open session description are specified in clause 6.5 of this specification.

4.13
Open Session
A session that is currently associated with a session identifier.

4.14
behavior
External appearance or action

4.15
undefined behavior
Behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which this International Standard imposes no requirements .

4.16
unspecified behavior
Use of an unspecified value, or other behavior where this International Standard provides two or more possibilities and imposes no further requirements on which is chosen in any instance

4.17
implementation-defined behavior
Unspecified behavior where each implementation documents how the choice is made .

4.18
value
precise meaning of the contents of an object when interpreted as having a specific type.

4.19
implementaion-defined value
unspecified value where each implementation documents how the choice is made.

4.20
unspecified value
valid value of the relevant type where this International Standard imposes no requirements on which value is chosen in any instance.

4.21
parameter
object declared as part of a function declaration or definition that acquires a value on entry to the function, or an identifier from the comma-separated list bounded by the parentheses immediately following the macro name in a function-like macro definition

4.22
character
The smallest component of written language that has semantic value; refers to the abstract meaning and/or shape. It is used as the basic unit of encoding for the Unicode character encoding.

4.23
Character encoding
Mapping from a character set definition to the actual code units used to represent the data plus byte serialization.

4.24
IMNegotiables
IMNegotiables is an IMObject which has attributes in the components. Attributes are exchanged between the components.

5 Symbols (and abbreviated terms)

5.1
IM
Abbreviation of Input Method

5.2
XIM
(1)Abbreviation of X Input Method, (2) reference to X11R6 XIM protocol standard and API.

5.3
HRN
Human Readable Name

5.4
UI
User Interface

5.5
SPI
Abbreviation of Service Provider Interface

5.6
IMText
Basic text object which is used in IM engine SPI. It may have strings with attributes.

5.7
UCS
A character encoding defined in ISO10646/Unicode standard. See [Unicode]

5.8
UTF
A character encoding defined in ISO10646/Unicode standard. See [Unicode]

5.9
BOM
A mark for indicating endian defined in ISO10646/Unicode standard. See [Unicode]

6 Service provider Interface

6.1 IMEngineModule

Loading IMEngineModule into IMEngine Containar

im_engine_module_init() function may be invoked after loading IMEngineModule into IMEngineContainer.

The *im_engine_module_init()* function is the first entry point for loading IMEngineModule by IMEngineContainer.

Note There may not be any resource management services through the SPI, IMEngineModule cannot rely on such services when this function invoked .

IMEngine may load the dynamic IMEngine module during IM engine module initilization, or just provide informations of its hardwired IMLogics.

SYNOPSIS

```
IMBool im_engine_module_init();
```

Argument

No argument

Returns

TRUE when IMEngine Module is loaded successfully, otherwise, FALSE.

Texonomy

IMEngineShallProvide, IMEngineContainerShallConsume

6.1.2 Unloading IMEngineModule from IMEngine Containar

After using *IMEngineModule*, *IMEngine Containar* may invoke *im_engine_module_exit* function to notify IMEngine Module that IMEngine Containar no longer uses this IMEngineModule. After calling *im_engine_module_exit*, *IMEngine Containar* may unload *IMEngineModule*.

Although IMEngine Containar shall not invoke *im_engine_module_exit* function before calling *im_engine_module_init* to the same IMEngineModule, IMEngine Containar may invoke *im_engine_module_init* function again after calling *im_engine_module_exit* function.

Note *Upon returning from im_engine_module_exit* function, all resources allocated in the IMEngineModule by shall be released.

Interface

```
IMBool im_engine_module_exit();
```

Argument

No argument.

Returns

TRUE when IMEngineModule is unloaded successfully, otherwise, FALSE.

Texonomy

IMEngineShallProvide,IMEngineContainerShallConsume

6.2 IMEngine

6.2.1 IMEngine Attributes and IMEngine Method

IMEngine consists of IMEngine Attributes and IMEngine Method. IMEngine Attributes have the names, their data types. The value of IMEngine Attributes are associated with IMEngine Attributes' name,that specifies their data types, and what they mean.

IMEngine shall have the following attribute

id

> ‘‘id” is an attribute name that specifies an identifier to distinguish IMEngine. The value of “id” is an IMValue which contains a pointer to IMChar* string. The string shall be composed of character specified by ISO/IEC 646 IRV. Reserved Domain Name points to IMEngine may be used for the value of this attribute such as “com.example.engine.sample”. IMEngine shall provide this attribute.
>
> IMEngineContainerShallConsume, IMEngineShallProvide

cname

> ‘‘cname” is an attribute name that specifies a name of IMEngine.The value of “cname” is an IMValue which contains a pointer to IMChar* string. The string shall be composed of character specified by ISO/IEC 646 IRV. IMEngine shall provide this attribute.
>
> IMEngineContainerShallConsume, IMEngineShallProvide

name

> ‘‘name” is an attribute name that specifies a name of IMEngine. The value of “name” is an IMValue which contains a pointer to an IMString instance. This attribute may be used to display IMEngine's name in the case of IMClient cannot handle. "name" attribute or IMLogic doesn't provide "name" attribute.
>
> IMEngineContainerMayConsume, IMEngineMayProvide

logics

> ‘‘logics” is an attribute name that specifies the IMLogic information that IMEngine. The value of "logics" is an IMValue instance which contains a pointer to an IMObjectArray instance holding an array of IMLogic instance. This arrributes specifies IMLogics that IMEngine supports.
>
> IMEngineContainerShallConsume,IMEngineShallProvide,

Languages

> ‘‘languages” is an attribute name that specifies the languages supported by IMEngine. The value of “languages” is an IMValue instance which contains a pointer to an IMObjectArray instance holding an array of it.
>
> IMEngineContainerMayConsume, IMEngineMayProvide

In addition to that, IMEngine should provide the following attribute.

Keymaps

> “keymaps” is an attribute name that specifies the set of keymaps that works with the IMEngine. The value of “keymaps” is an IMValue instance which contains a pointer to an IMObjectArray instance holding an array of it.
>
> IMEngineContainerMayConsume, IMEngineMayProvide

hotkeys

> “hotkeys” is an attribute name that specifies the combination of keyboard strokes that perform pre-defind action when pressed simultaneously. The value of “hotkeys” is an IMValue instance which contains a pointer to an IMHotkeyProfile instance.
>
> IMEngineContainerMayConsume, IMEngineMayProvide

IMEngine may provide additional attributes, in this case, the behavior is undefined.

6.2.2 Setting IMEngine Attribute

> *im_engine_set_attr* - setting IMEngine attribute

The im_engine_set_attr() function set IMEngine's attribute associated with IMEngine attributes' identifier described in this clause. The im_engine_set_attr() function shall return an boolean value which holds a result whether the attribute is set successfully or not.

The *key* argument specifies the name of the IMEngine attributes described in this clause.

The *value* argument specifies the value of the IMEngine attribute associated with the key.

When IMEngine already has the attribute specified by the *key* argument, the im_engine_set_attr() shall override the previous assigned one with the value argument.

SYNOPSYS

> IMBool im_engine_set_attr(IMEngine *engine, const IMChar* key, IMValue *value)

Argument

engine:
> a pointer to an IMEngine instance that the IMEngine attribute is set to.

key:
> a pointer to a string in type of IMChar* that indicates the name of IMEngine attributes.

value:
> a pointer to a IMValue instance which contains the corresponding data type specified by the key.

Returns

TRUE on success. FALSE otherwise

6.2.3 Getting IMEngine Attribute

im_engine_get_attr - gettine IMEngine attribute

The im_engine_get_attr() function gets the value of IMEngine attribute associated with the name of IMEngine attributes.
The *key* specifies the name of the IMEngine attributes described in this clause.
The *value* specifies an IMValue objects holds a result value associated with *key*.
When IMEngine has not the value associated with *key, value* may be cleared. On success, *value* IMValue instance holds Value of the attribute.

SYNOPSIS

IMBool im_engine_get_attr(IMEngine *engine, const IMChar *key, IMValue *value)

Arguments

engine:
a pointer to an IMEngine instance that the IMEngine attribute is set to.

key:
a poiter to a string in type of IMChar* that indicates the name of IMEngine attributes.

value:
a pointer to a IMValue instance which is used to store the return value of IMEngine attributes specified by the key.

Returns

TRUE on success, FALSE otherwise.

6.2.4 Checking IMEngine Attribute

SYNPOSIS

im_engine_has_attr()

The im_engine_has_attr() function checks whether the value of IMEngine attribute associated with the name of IMEngine attributes are set.

Interfaces

IMBool im_engine_has_attr(IMEngine *engine, const IMChar* key)

Arguments

engine:
a pointer to an IMEngine instance that the IMEngine attribute is set to.

key:
a poiter to a string in type of IMChar* that indicates the name of IMEngine attributes.

Returns

TRUE when the attribute is already set, FALSE otherwise.

6.3 IMLogic

IMLogic is an object which provides a way to input characters. IMLogic consists of IMLogic Attribute and IMLogic Method.

6.3.1 Registering IMLogic specific functionalities

im_logic_register_func() - register function to IMLogic instance

The im_logic_register_func() is used to change the action taken by a IMLogic instance on receipt of a specific event assosiated with functionality identifier. The im_logic_register_func() shall return a pointer to a previous assigned function assigned with key by im_logic_register_func(). A null pointer is returned when no function is assigned. When the corresponding event happens, registered function should be invoked. All functionality identified which begin with prefix "imlogic_" except for "imlogic_ext_" are reserved for future purpose.

Interfaces

typedef IMBool (*im_logic_context_handler_t)(IMContext *context)

typedef IMBool (*im_logic_event_handler_t)(IMContext *context, IMEvent *event)

IMPointer im_logic_register_func(IMLogic *logic, const IMChar* key, IMPointer function)

Arguments

logic:

a pointer to IMLogic instance

key:

a pointer to a string in type of IMChar* that indicates functionality's identifier.

Functionality's identifier are the followings:

imlogic_initialize

function registered with the identifier "imlogic_initialize" may be invoked by IMEngine Container when the IMLogic is about to be used by client. this function shall not be invoked before initializing IMEngine bound to the corresponding IMLogic.

imlogic_finalize

function registerd with the identifier "imlogic_finalize" may be invoked by IMEngine Container when IMContext, that the IMLogic is bound to, about to be free'd. the function may also be invoked by IMEngine Container before calling im_engine_finalize() function.

function:

a pointer to the function to be registered to IMLogic. when NULL is passed as a function value, the previous value of the function is removed.

Returns

the im_logic_register_func() function return the previous value of the registerd function, or IMLOGIC_REGISTER_FUNC_ERR on error.

Texonomy

IMEngineContainerShallProvide,IMEngineMayConsume

6.3.2 IMLogic Attribute

This section describes the names of the IMLogic attributes, their data types, and what they mean. The IMLogic attributes which is not described in this section, are implementation-defined.

id:

IMValue which contains a pointer to a string in type with IMChar* indicating IMLogic's id. IMLogic's id, which shall be composed only with character specified by ISO/IEC 646 IRV. "id" attribute shal be used to distinguish each IMLogic, so it recomends to use reversed domain name such as "com.example.logic.sample"

cname:

IMValue whith contains a pointer to a string in type with IMChar* indicating IMLogic's cname. IMLogic's cname shall be composed only with character specified by ISO/IEC 646 IRV. This attribute may be used to display IMLogic's name in the case of IMClient cannot handle "name" attribute or IMLogic doesn't provide "name" attribute.

name:

IMValue which contains a pointer to an IMString instance.

6.3.3 Setting IMLogic's Attributes

SYNPOSIS

im_logic_set_attr - setting IMLogic's attribute

The im_logic_set_attr() function sets IMLogic's attribute associated with IMLogic's identifier described in [6.3.2 IMLogic Attributes] The im_logic_set_attr() function shall return an boolean value which holds a result whether the attribute is set successfully or not. The key argument specifies the name of the IMLogic attributes described in [IMLogic Attributes Meanings]. The value argument specifies the value of the IMLogic attribute associated with the key. When IMLogic already has the attribute specified by the key argument, the value of the attribute shall be overwritten the previous assigned one.

Interfaces

IMBool im_logic_set_attr(IMLogic *logic, const IMChar *key, IMValue *value, IMLogicError *error);

Arguments

logic:

a pointer to an IMLogic instance that the IMLogic attribute is set to.

key:

a pointer to a string in type of IMChar* that indicates the name of IMLogic attributes.

value:

a pointer to an IMValue which contains the corresponding data type specified by the key.

error:

In the case of failure, status of error is stored.

Returns

TRUE on success, FALSE otherwise. In the case of failure, following error code is stored on error.

6.3.4 Getting IMLogic's Attribute

im_logic_get_attr()

The im_logic_get_attr() function gets the value of IMLogic attribute associated with the name of IMLogic attributes.

Interfaces

IMBool im_logic_get_attr(IMLogic *logic, const IMChar *key, IMValue *value, IMLogicError *error);

Arguments

logic:

an pointer to an IMLogic instance

key:

a pointer to a string in type of IMChar* that indicates IMLogic information's identifier

value:

a pointer to an IMValue which contains the corresponding data type specified by key.

error:

In the case of failure, status of error is stored.

Returns

TRUE on success, FALSE otherwise. In the case of failure, following error code is stored on error

IMLOGIC_ERR_EMPTY when an information is not set yet with specified identifier.

6.4 IMContext

IMContext is an object which provides the state information flow between an application and IMComponent.

6.4.1 Event Handling

im_dispatch_event() - Dispathes an event to the corresponding IMContext.

im_dispatch_event() function dispatches an event to the corresponding IMContext.

Interfaces

IMBool im_dispatch_event(IMContext* context, IMEvent *event)

Arguemnts

context: a pointer to an instance of IMContext which the event is sent to.

event: a pointer to an instance of IMEvent which contains an arbitary data.

Returns

TRUE if event is queued , FALSE otherwise.

Texonomy

IMEngineContainerShallProvide

6.5 Attribute lists with behavior

6.5.1 The `im_negotiables_get_info` function

Synopsis

IMBool im_negotiables_get_info(IMNegotiables* negotiables, const char* key, IMValue** value);

Description

The im_negotiables_get_info() function shall get the value associated with the key. The argument *value* will be filled with null pointer if negotiables does not have any value associated with the key.

The argument *negotiables* points to an instance of IMObject which can get or set some information for itself or others. IMNegotiable types are IMEngine, IMLogic, IMContext and IMSession.

The argument *key* points to the string which represents any property of IMNegotiable. This can be a UNIX path like string with hierarchical representation.

The argument *value* is location to store IMValue type pointer, which shall be filled with the new pointer to an instance of IMValue containing the information associated with the key, or shall be a null pointer if there is no key associated with the key.

Return Value

If this function succeed, it returns TRUE and **value* shall have valid pointer to new IMValue object or FALSE and fill the *value with null pointer if it failed, should be unreferenced with im_object_unref() when no longer need.

Taxonomy information:

EngineMayConsume, UICMayConsume, EngineContainerShallProvide, UICContainerShallProvide

6.5.2 The `im_negotiable_set_info` function

Synopsis

IMBool im_negotiable_set_info(IMNegotiable* negotiable, const char* key, const IMValue* value);

Description

The im_negotiable_set_info() function shall set the value associated with the key. IMNegotiable types are IMEngine, IMLogic, IMContext and IMSession. Valid attributes of each objects are described in their section.

The argument negotiable points to an instance of IMObject which can get or set some information for itself.

The argument key points to the string which represents any property of IMNegotiable.

The argument value points to an IMValue to be copied to the IMValue associated with the key inside IMNegotiable.

Return value

It returns TRUE if it succeed, or FALSE.

Taxonomy information:

EngineMayConsume, UICMayConsume, EngineContainerShallProvide, UICContainerShallProvide

Note: im_negotiables_get_info() and im_negotiable_set_info(), these two functions are provided to query or set IMNegotiables attributes. IMEngine, IMLogic, IMSession and IMContext can have some attributes and exchange them with each other. And this functions can be used for exchanging extra information which is not supported by IMFramework.

6.6. Event Listener interface

6.6.1 The `im_session_register_event_filter` function

Interfaces

typedef IMBool (*IMEventFilter)(IMComponent*, IMEvent*, IMPointer);

IMInt im_session_register_event_filter(IMSession* session,

IMComponent* component,

IMInt priority,

IMEventFilter filter,

IMPointer user_data);

Description

The im_session_register_event_filter() function shall register event filter callback function for the events specified when the component registered itself to IM-BUS. When the events are generated, IM framework shall call this filter callback function with the event and the user_data. But if the event does not match the event role, IM Framework shall not call the callback function.

The argument session points an instance of IMSession which you want to register this filter.

The argument component points to an instance of IMObject which want to listen any events.

The argument priority is an integer value from 0 to 100 indicating the degree of priority. 0 is the highest and 100 is lowest priority. If another filter function was already registered with the same priority, the last registered filter function shall be called after the previous registered filter function.

The argument filter is the function pointer to the user-defined event filter function. When a new event is generated, IM framework shall call this filter function with the event. The first argument of the filter function is a pointer to component, the second argument is an event and the third argument is a user defined data pointer which is passed when this callback function is registered.

The argument user_data points to user specific data which is passed when the filter function is called.

Return Value

The new registered event filter id is returned, or -1 if it fails. The filter id is non-negative integer value. The returned filter id can be used with im_session_unregister_event_filter function.

Taxonomy information:

EngineMayConsume, UICMayConsume, EngineContainerShallProvide, UICContainerShallProvide

6.6.2 The `im_session_unregister_event_filter` function

Interfaces

IMBool im_session_unregister_event_filter(IMSession* session, IMInt filter_id);

Description

The im_session_unregister_event_filter() function shall deletes previously registered event filter function by filter_id. If there is no such filter id, this function shall do nothing.

The argument session points to an instance of IMSession which you want to unregister this filter.

The argument filter_id is the event filter id returned when the filter is registered.

Return Value

It returns TRUE if it succeed, or FALSE.

Taxonomy information:

EngineMayConsume, UICMayConsume, EngineContainerShallProvide, UICContainerShallProvide

Note: Event Listener interface provide a mechanism for registering event filter functions for specific events in order for the components to receive the events they want to process.

6.7. Object handling and convenient functions

Modules which provide public convenient functions:

- Primitive Data Types
- Macros
- Memory handling
- IMMain
- IMUnicode
- IMUtilities
- IMQuark
- IMKeyCodes

Modules which provide public object handling functions:

- IMAttribute
- IMEvent
- IMEventRoles
- IMEventTypes
- IMHotkey and IMHotkeyProfile

- IMLookupTable and IMCandidate
- IMObject
- IMProperty
- IMString
- IMText
- IMTypeClass
- IMValue

6.7.1 Convenient functions:

Primitive Data Types

Defines

```
#define IM_MININT16  ((IMInt16) 0x8000)
#define IM_MAXINT16  ((IMInt16) 0x7fff)
#define IM_MAXUINT16  ((IMInt16) 0xffff)
#define IM_MININT32  ((IMInt32) 0x80000000)
#define IM_MAXINT32  ((IMInt32) 0x7fffffff)
#define IM_MAXUINT32  ((IMInt32) 0xffffffff)
#define IM_MININT64  ((IMInt64) IM_IMINT64_CONSTANT(0x8000000000000000))
#define IM_MAXINT64  IM_IMINT64_CONSTANT(0x7fffffffffffffff)
#define IM_MAXUINT64  IM_IMINT64_CONSTANT(0xffffffffffffffffU)
#define IM_POINTER_TO_INT(p)  ((IMInt) (p))
#define IM_POINTER_TO_UINT(p)  ((IMUInt) (p))
#define IM_INT_TO_POINTER(i)  ((IMPointer) (i))
#define IM_UINT_TO_POINTER(u)  ((IMPointer) (u))
#define IM_POINTER_TO_SIZE(p)  ((IMSize) (p))
#define IM_SIZE_TO_POINTER(s)  ((IMPointer) (s))
#define IM_LITTLE_ENDIAN  1234
#define IM_BIG_ENDIAN  4321
#define IMUINT16_SWAP_LE_BE_CONSTANT(val)
#define IMUINT32_SWAP_LE_BE_CONSTANT(val)
#define IMUINT64_SWAP_LE_BE_CONSTANT(val)
#define IMUINT16_SWAP_LE_BE(val)  (IMUINT16_SWAP_LE_BE_CONSTANT (val))
#define IMUINT32_SWAP_LE_BE(val)  (IMUINT32_SWAP_LE_BE_CONSTANT (val))
#define IMUINT64_SWAP_LE_BE(val)  (IMUINT64_SWAP_LE_BE_CONSTANT (val))
#define IMINT16_FROM_LE(val)  (IMINT16_TO_LE (val))
#define IMUINT16_FROM_LE(val)  (IMUINT16_TO_LE (val))
#define IMINT16_FROM_BE(val)  (IMINT16_TO_BE (val))
#define IMUINT16_FROM_BE(val)  (IMUINT16_TO_BE (val))
#define IMINT32_FROM_LE(val)  (IMINT32_TO_LE (val))
#define IMUINT32_FROM_LE(val)  (IMUINT32_TO_LE (val))
#define IMINT32_FROM_BE(val)  (IMINT32_TO_BE (val))
#define IMUINT32_FROM_BE(val)  (IMUINT32_TO_BE (val))
#define IMINT64_FROM_LE(val)  (IMINT64_TO_LE (val))
```

33 #define **IMUINT64_FROM_LE**(val) (IMUINT64_TO_LE (val))

34 #define **IMINT64_FROM_BE**(val) (IMINT64_TO_BE (val))

35 #define **IMUINT64_FROM_BE**(val) (IMUINT64_TO_BE (val))

Typedefs

36 typedef unsigned int **IMType**

37 typedef size_t **IMSize**

A data type which holds the size of a specified data or array, or an index.

38 typedef char **IMChar**

A data type which holds a single signed byte.

39 typedef unsigned char **IMUChar**

A data type which holds a single unsigned byte.

40 typedef int **IMInt**

A data type which holds a single signed integer.

41 typedef unsigned int **IMUInt**

A data type which holds a single unsigned integer.

42 typedef int **IMBool**

A data type which holds a single boolean value.

43 typedef double **IMDouble**

A data type which holds a single double value.

44 typedef void * **IMPointer**

A data type which holds a single pointer.

45 typedef const void * **IMConstPointer**

A data type which holds a single const pointer.

46 typedef **IMBool**(* **IMEqualFunc**)(**IMConstPointer** a, **IMConstPointer** b)

Function to compare two data to see whether they are equal or not.

47 typedef **IMInt**(* **IMCompareFunc**)(**IMConstPointer** a, **IMConstPointer** b)

Function to compare two data.

48 typedef **IMInt**(* **IMCompareDataFunc**)(**IMConstPointer** a, **IMConstPointer** b, **IMPointer** user_data)

Function to compare two data by using a specified user_data.

49 typedef void(* **IMFunc**)(**IMPointer** data, **IMPointer** user_data)

Function to manipulate a specified data.

50 typedef void(* **IMDestroyFunc**)(**IMPointer** data)

Function to destroy a specified data.

Define Documentation

#define IM_BIG_ENDIAN 4321

#define IM_INT_TO_POINTER(i) ((IMPointer) (i))

#define IM_LITTLE_ENDIAN 1234

#define IM_MAXINT16 ((IMInt16) 0x7fff)

#define IM_MAXINT32 ((IMInt32) 0x7fffffff)

#define IM_MAXINT64 IM_IMINT64_CONSTANT(0x7fffffffffffffff)

#define IM_MAXUINT16 ((IMInt16) 0xffff)

#define IM_MAXUINT32 ((IMInt32) 0xffffffff)

#define IM_MAXUINT64 IM_IMINT64_CONSTANT(0xffffffffffffffffU)

#define IM_MININT16 ((IMInt16) 0x8000)

#define IM_MININT32 ((IMInt32) 0x80000000)

#define IM_MININT64 ((IMInt64) IM_IMINT64_CONSTANT(0x8000000000000000))

#define IM_POINTER_TO_INT(p) ((IMInt) (p))

#define IM_POINTER_TO_SIZE(p) ((IMSize) (p))

#define IM_POINTER_TO_UINT(p) ((IMUInt) (p))

#define IM_SIZE_TO_POINTER(s) ((IMPointer) (s))

#define IM_UINT_TO_POINTER(u) ((IMPointer) (u))

#define IMINT16_FROM_BE(val) (IMINT16_TO_BE (val))

#define IMINT16_FROM_LE(val) (IMINT16_TO_LE (val))

#define IMINT32_FROM_BE(val) (IMINT32_TO_BE (val))

#define IMINT32_FROM_LE(val) (IMINT32_TO_LE (val))

#define IMINT64_FROM_BE(val) (IMINT64_TO_BE (val))

#define IMINT64_FROM_LE(val) (IMINT64_TO_LE (val))

#define IMUINT16_FROM_BE(val) (IMUINT16_TO_BE (val))

#define IMUINT16_FROM_LE(val) (IMUINT16_TO_LE (val))

#define IMUINT16_SWAP_LE_BE(val) (IMUINT16_SWAP_LE_BE_CONSTANT (val))

#define IMUINT16_SWAP_LE_BE_CONSTANT(val)

Value:
```
((IMUInt16) (   \
        (IMUInt16) ((IMUInt16) (val) >> 8) |                    \
        (IMUInt16) ((IMUInt16) (val) << 8)))
```

#define IMUINT32_FROM_BE(val) (IMUINT32_TO_BE (val))

#define IMUINT32_FROM_LE(val) (IMUINT32_TO_LE (val))

#define IMUINT32_SWAP_LE_BE(val) (IMUINT32_SWAP_LE_BE_CONSTANT (val))

#define IMUINT32_SWAP_LE_BE_CONSTANT(val)

Value:
```
((IMUInt32) (   \
        (((IMUInt32) (val) & (IMUInt32) 0x000000ffU) << 24) |   \
        (((IMUInt32) (val) & (IMUInt32) 0x0000ff00U) <<  8) |   \
        (((IMUInt32) (val) & (IMUInt32) 0x00ff0000U) >>  8) |   \
        (((IMUInt32) (val) & (IMUInt32) 0xff000000U) >> 24)))
```

#define IMUINT64_FROM_BE(val) (IMUINT64_TO_BE (val))

#define IMUINT64_FROM_LE(val) (IMUINT64_TO_LE (val))

#define IMUINT64_SWAP_LE_BE(val) (IMUINT64_SWAP_LE_BE_CONSTANT (val))

#define IMUINT64_SWAP_LE_BE_CONSTANT(val)

Value:
```
((IMUInt64) (                \
        (((IMUInt64) (val) &                                                \
            (IMUInt64) IM_IMINT64_CONSTANT (0x00000000000000ffU)) << 56) |  \
        (((IMUInt64) (val) &                                                \
            (IMUInt64) IM_IMINT64_CONSTANT (0x000000000000ff00U)) << 40) |  \
        (((IMUInt64) (val) &                                                \
            (IMUInt64) IM_IMINT64_CONSTANT (0x0000000000ff0000U)) << 24) |  \
        (((IMUInt64) (val) &                                                \
            (IMUInt64) IM_IMINT64_CONSTANT (0x00000000ff000000U)) <<  8) |  \
        (((IMUInt64) (val) &                                                \
            (IMUInt64) IM_IMINT64_CONSTANT (0x000000ff00000000U)) >>  8) |  \
        (((IMUInt64) (val) &                                                \
            (IMUInt64) IM_IMINT64_CONSTANT (0x0000ff0000000000U)) >> 24) |  \
        (((IMUInt64) (val) &                                                \
            (IMUInt64) IM_IMINT64_CONSTANT (0x00ff000000000000U)) >> 40) |  \
        (((IMUInt64) (val) &                                                \
            (IMUInt64) IM_IMINT64_CONSTANT (0xff00000000000000U)) >> 56)))
```

Typedef Documentation

typedef int IMBool

A data type which holds a single boolean value.

an IMBool variable shall only take two different values: TRUE or FALSE.

typedef char IMChar

A data type which holds a single signed byte.

typedef IMInt(* IMCompareDataFunc)(IMConstPointer a, IMConstPointer b, IMPointer user_data)

Function to compare two data by using a specified user_data.

Returns:

-1 if a < b, 0 if a == b, 1 if a > b

typedef IMInt(* IMCompareFunc)(IMConstPointer a, IMConstPointer b)

Function to compare two data.

Returns:

-1 if a < b, 0 if a == b, 1 if a > b

typedef const void* IMConstPointer

A data type which holds a single const pointer.

typedef void(* IMDestroyFunc)(IMPointer data)

Function to destroy a specified data.

For example it can be **im_object_unref()** to release an specified object.

typedef double IMDouble

A data type which holds a single double value.

typedef IMBool(* IMEqualFunc)(IMConstPointer a, IMConstPointer b)

Function to compare two data to see whether they are equal or not.

typedef void(* IMFunc)(IMPointer data, IMPointer user_data)

Function to manipulate a specified data.

typedef int IMInt

A data type which holds a single signed integer.

typedef void* IMPointer

A data type which holds a single pointer.

An IMPointer variable can be pointed to any type of data, but in most case, it may point to an **IMObject** instance.

typedef size_t IMSize

A data type which holds the size of a specified data or array, or an index.

typedef unsigned int IMType

typedef unsigned char IMUChar

A data type which holds a single unsigned byte.

typedef unsigned int IMUInt

A data type which holds a single unsigned integer.

Macros

Defines

```
51  #define IM_BEGIN_DECLS
52  #define IM_END_DECLS
53  #define NULL  ((void*) 0)
54  #define FALSE  (0)
55  #define TRUE  (!FALSE)
56  #define IM_MAX(a, b)  (((a) > (b)) ? (a) : (b))
57  #define IM_MIN(a, b)  (((a) < (b)) ? (a) : (b))
58  #define IM_ABS(a)  (((a) < 0) ? -(a) : (a))
59  #define IM_CLAMP(x, low, high)  (((x) > (high)) ? (high) : (((x) < (low)) ? (low) : (x)))
60  #define IM_N_ELEMENTS(arr)  (sizeof (arr) / sizeof ((arr)[0]))
61  #define IM_SWAP(type, a, b)
62  #define IM_GNUC_WARN_UNUSED_RESULT
63  #define IM_GNUC_EXTENSION
64  #define IM_GNUC_PRINTF(format_idx, arg_idx)
65  #define IM_GNUC_NORETURN
```

Define Documentation

#define FALSE (0)

#define IM_ABS(a) (((a) < 0) ? -(a) : (a))

#define IM_BEGIN_DECLS

#define IM_CLAMP(x, low, high) (((x) > (high)) ? (high) : (((x) < (low)) ? (low) : (x)))

#define IM_END_DECLS

#define IM_GNUC_EXTENSION

#define IM_GNUC_NORETURN

used to tell gcc about functions that never return, such as abort()

#define IM_GNUC_PRINTF(format_idx, arg_idx)

used to tell gcc about printf format strings

#define IM_GNUC_WARN_UNUSED_RESULT

#define IM_MAX(a, b) (((a) > (b)) ? (a) : (b))

#define IM_MIN(a, b) (((a) < (b)) ? (a) : (b))

#define IM_N_ELEMENTS(arr) (sizeof (arr) / sizeof ((arr)[0]))

#define IM_SWAP(type, a, b)

Value:
```
do {                    \
        type tmp = (a);     \
        (a) = (b);          \
        (b) = tmp;          \
    } while(0)
```

#define NULL ((void*) 0)

#define TRUE (!FALSE)

Memory handling

Defines

66 #define **im_new**(struct_type, n_structs) ((struct_type *) im_malloc (((**IMSize**) sizeof (struct_type)) * ((**IMSize**) (n_structs))))

67 #define **im_new0**(struct_type, n_structs) ((struct_type *) im_malloc0 (((**IMSize**) sizeof (struct_type)) * ((**IMSize**) (n_structs))))

68 #define **im_renew**(struct_type, mem, n_structs) ((struct_type *) im_realloc ((mem), ((**IMSize**) sizeof (struct_type)) * ((**IMSize**) (n_structs))))

69 #define **im_slice_new**(struct_type) ((struct_type *) im_slice_alloc ((**IMSize**) sizeof (struct_type)))

70 #define **im_slice_new0**(struct_type) ((struct_type *) im_slice_alloc0 ((**IMSize**) sizeof (struct_type)))

71 #define **im_slice_delete**(struct_type, mem)

Functions

72 IM_BEGIN_DECLS void **im_slice_init** ()

Initialize the slice allocation system.

73 **IMPointer im_slice_alloc** (**IMSize** block_size)

Allocate a memory slice with given size.

74 **IMPointer im_slice_alloc0** (**IMSize** block_size)

Allocate a memory slice with given size, and fill it with zero.

75 void **im_slice_free** (**IMSize** block_size, **IMPointer** block)

*Free a memory slice with specific size, which was allocated by **im_slice_alloc()** or **im_slice_alloc0()**.*

76 **IMPointer im_malloc** (**IMSize** n_bytes)

Allocate a memory block with given size.

77 **IMPointer im_malloc0** (**IMSize** n_bytes)

Allocate a memory block with given size, and fill it with zero.

78 **IMPointer im_realloc** (**IMPointer** mem, **IMSize** n_bytes)

*Resize a memory block which was allocated by **im_malloc()** or **im_malloc0()**.*

79 void **im_free** (**IMPointer** mem)

*Free a memory block which is allocated by **im_malloc()**, **im_malloc0()** or **im_realloc()**.*

Define Documentation

#define im_new(struct_type, n_structs) ((struct_type *) im_malloc (((IMSize) sizeof (struct_type)) * ((IMSize) (n_structs))))

#define im_new0(struct_type, n_structs) ((struct_type *) im_malloc0 (((IMSize) sizeof (struct_type)) * ((IMSize) (n_structs))))

#define im_renew(struct_type, mem, n_structs) ((struct_type *) im_realloc ((mem), ((IMSize) sizeof (struct_type)) * ((IMSize) (n_structs))))

#define im_slice_delete(struct_type, mem)

Value:
```
do {        \
        if (1) im_slice_free((IMSize) sizeof (struct_type), mem);   \
        else (void)((struct_type*) 0 == (mem));                     \
        } while (0)
```

#define im_slice_new(struct_type) ((struct_type *) im_slice_alloc ((IMSize) sizeof (struct_type)))

#define im_slice_new0(struct_type) ((struct_type *) im_slice_alloc0 ((IMSize) sizeof (struct_type)))

Function Documentation

void im_free (IMPointer *mem*)

Free a memory block which is allocated by **im_malloc()**, **im_malloc0()** or **im_realloc()**.

IMPointer im_malloc (IMSize *n_bytes*)

Allocate a memory block with given size.

It's optimized for allocating large memory block.

Unlike **im_slice_alloc()**, the block size will be recorded, so it's not necessary to specify the block size when freeing it.

Currently, it's just a wrapper of system malloc.

IMPointer im_malloc0 (IMSize *n_bytes*)

Allocate a memory block with given size, and fill it with zero.

IMPointer im_realloc (IMPointer *mem*, IMSize *n_bytes*)

Resize a memory block which was allocated by **im_malloc()** or **im_malloc0()**.

Parameters:

mem Pointer to the memory block, if it's 0, a new memory block will be allocated.
n_bytes New size of the memory block.

Returns:

A pointer to resized or newly created memory block.

IMPointer im_slice_alloc (IMSize *block_size*)

Allocate a memory slice with given size.

IMPointer im_slice_alloc0 (IMSize *block_size*)

Allocate a memory slice with given size, and fill it with zero.

void im_slice_free (IMSize *block_size*, IMPointer *block*)

Free a memory slice with specific size, which was allocated by **im_slice_alloc()** or **im_slice_alloc0()**.

IM_BEGIN_DECLS void im_slice_init ()

Initialize the slice allocation system.

It'll be called inside im_init (), so don't call it directly.

IMMain

Functions

80 IM_BEGIN_DECLS void **im_init** (int *argc, char ***argv)

Initialize IMFramework.

Function Documentation

IM_BEGIN_DECLS void im_init (int * *argc*, char * *argv*)**

Initialize IMFramework.

It initializes internal data of IMFramework. It must be called as soon as the main program runs, before calling any other functions of IMFramework.

IMUnicode

Defines

81 #define **IM_UNICHAR_INVALID** ((**IMUniChar**)(-1))
82 #define **IM_UNICHAR_REPLACEMENT** ((**IMUniChar**)0xFFFD)
83 #define **im_utf8_char_length**(p) (**IMSize**)(**im_utf8_skip** [*(**IMUChar** *)(p)])
84 #define **im_utf8_next_char**(p) (**IMChar** *)((p) + im_utf8_char_length(p))
85 #define **im_utf16_char_length**(p)

86 #define **im_utf16_next_char**(p) (((**IMUInt16***)(p)) + im_utf16_char_length(p))

Functions

87 **IMUniChar im_utf8_get_char** (const **IMChar** *src)

Get an unicode char from an utf8 char sequence.

88 **IMSize im_utf8_strlen** (const **IMChar** *src, **IMInt** maxlen)

Return how many unicode chars in an utf8 string.

89 **IMBool im_utf8_string_validate** (const **IMChar** *str, **IMInt** maxlen, const **IMChar** **end)

*Validate a UTF8 string, return TRUE if valid, put pointer to first invalid char in **end.*

90 **IMUniChar im_utf16_get_char** (const **IMUInt16** *src)

Get an unicode char from an utf8 char sequence.

91 **IMSize im_utf16_strlen** (const **IMUInt16** *src, **IMInt** maxlen)

Return how many unicode chars in a given utf16 sequence.

92 **IMBool im_utf16_string_validate** (const **IMUInt16** *str, **IMInt** maxlen, const **IMUInt16** **end)

*Validate a UTF16 string, return TRUE if valid, put pointer to first invalid char in **end.*

93 **IMSize im_unichar_from_utf8** (**IMUniChar** *dest, const **IMChar** *src, **IMInt** src_len)

Convert an utf8 char sequence to ucs4.

94 **IMSize im_unichar_to_utf8** (**IMChar** *dest, **IMUniChar** src, **IMSize** dest_size)

Convert an ucs4 code to utf8 char sequence.

95 **IMSize im_unichar_from_utf16** (**IMUniChar** *dest, const **IMUInt16** *src, **IMInt** src_len)

Convert an utf16 char sequence to ucs4.

96 **IMSize im_unichar_to_utf16** (**IMUInt16** *dest, **IMUniChar** src, **IMSize** dest_size)

Convert an ucs4 code to utf16 char sequence.

97 **IMBool im_unichar_validate** (**IMUniChar** ch)

Validate an unicode character.

98 **IMSize im_utf8_string_to_ucs4** (**IMUniChar** *dest, **IMSize** dest_size, const **IMChar** *src, **IMInt** src_len)

Convert an utf8 string to ucs4 string.

99 **IMSize im_utf16_string_to_ucs4** (**IMUniChar** *dest, **IMSize** dest_size, const **IMUInt16** *src, **IMInt** src_len)

Convert an utf16 string to ucs4 string.

100 **IMSize im_ucs4_strlen** (const **IMUniChar** *src, **IMInt** maxlen)

Return how many unicode chars in a given ucs4 sequence.

101 **IMSize im_ucs4_string_utf8_length** (const **IMUniChar** *str, **IMInt** maxlen)

Return the minimum dest buffer size when converting an ucs4 string to utf8 string.

102 **IMSize im_ucs4_string_utf16_length** (const **IMUniChar** *str, **IMInt** maxlen)

Return the minimum dest buffer size when converting an ucs4 string to utf16 string.

103 **IMSize im_ucs4_string_to_utf8** (**IMChar** *dest, **IMSize** dest_size, const **IMUniChar** *src, **IMInt** src_len)

Convert an ucs4 string to utf8 string.

104 **IMSize im_ucs4_string_to_utf16** (**IMUInt16** *dest, **IMSize** dest_size, const **IMUniChar** *src, **IMInt** src_len)

Convert an ucs4 string to utf16 string.

Variables

105 const **IMChar** *const **im_utf8_skip**

Define Documentation

#define IM_UNICHAR_INVALID ((IMUniChar)(-1))

#define IM_UNICHAR_REPLACEMENT ((IMUniChar)0xFFFD)

#define im_utf16_char_length(p)

Value:
```
((p) ?                                  \
      (((*(IMUInt16*)(p)) >= 0xd800 &&       \
        (*(IMUInt16*)(p)) < 0xdc00 &&        \
        (*(IMUInt16*)((p)+1)) >= 0xdc00 &&   \
        (*(IMUInt16*)((p)+1)) < 0xe000) ?    \
       2 : 1) : 0)
```

#define im_utf16_next_char(p) (((IMUInt16*)(p)) + im_utf16_char_length(p))

#define im_utf8_char_length(p) (IMSize)(im_utf8_skip [*(IMUChar *)(p)])

#define im_utf8_next_char(p) (IMChar *)((p) + im_utf8_char_length(p))

Function Documentation

IMSize im_ucs4_string_to_utf16 (IMUInt16 * *dest*, IMSize *dest_size*, const IMUniChar * *src*, IMInt *src_len*)

Convert an ucs4 string to utf16 string.

Parameters:

dest Buffer to store result utf16 string.
dest_size Size of the dest buffer, in bytes.
src Source ucs4 string to be converted.
src_len Length of source ucs4 string, in number of IMUniChar elements, -1 means until 0.

Returns:

How many elements actually written to dest, if it's less than dest_size, then dest will be terminated by 0.

IMSize im_ucs4_string_to_utf8 (IMChar * *dest*, IMSize *dest_size*, const IMUniChar * *src*, IMInt *src_len*)

Convert an ucs4 string to utf8 string.

Parameters:

dest Buffer to store result utf8 string.
dest_size Size of the dest buffer, in bytes.
src Source ucs4 string to be converted.
src_len Length of source ucs4 string, in number of IMUniChar elements, -1 means until 0.

Returns:

How many bytes actually written to dest, if it's less than dest_size, then dest will be terminated by 0.

IMSize im_ucs4_string_utf16_length (const IMUniChar * *str*, IMInt *maxlen*)

Return the minimum dest buffer size when converting an ucs4 string to utf16 string.

Parameters:

see function im_ucs4_strlen

IMSize im_ucs4_string_utf8_length (const IMUniChar * *str*, IMInt *maxlen*)

Return the minimum dest buffer size when converting an ucs4 string to utf8 string.

Parameters:

see function im_ucs4_strlen

IMSize im_ucs4_strlen (const IMUniChar * *src*, IMInt *maxlen*)

Return how many unicode chars in a given ucs4 sequence.

Parameters:

src The zero terminated ucs4 sequence to be examined.
maxlen Maximum length of src, -1 means until 0.

IMSize im_unichar_from_utf16 (IMUniChar * *dest*, const IMUInt16 * *src*, IMInt *src_len*)

Convert an utf16 char sequence to ucs4.

Parameters:

dest destination buffer to store the ucs4 code.
src source buffer contains the utf16 char sequence.
src_len the size of source buffer.

Returns:

number of uint16 elements were actually converted.

IMSize im_unichar_from_utf8 (IMUniChar * *dest*, const IMChar * *src*, IMInt *src_len*)

Convert an utf8 char sequence to ucs4.

Parameters:

dest destination buffer to store the ucs4 code.
src source buffer contains the utf8 char sequence.
src_len the size of source buffer.

Returns:

number of chars in src actually converted.

IMSize im_unichar_to_utf16 (IMUInt16 * *dest*, IMUniChar *src*, IMSize *dest_size*)

Convert an ucs4 code to utf16 char sequence.

Parameters:

dest destination buffer to store utf16 char sequence.
src the ucs4 code to be converted.
dest_size the size of destination buffer.

Returns:

the number of uint16 elements actually written into dest.

IMSize im_unichar_to_utf8 (IMChar * *dest*, IMUniChar *src*, IMSize *dest_size*)

Convert an ucs4 code to utf8 char sequence.

Parameters:

dest destination buffer to store utf8 char sequence.
src the ucs4 code to be converted.
dest_size the size of destination buffer.

Returns:

the number of bytes actually written into dest.

IMBool im_unichar_validate (IMUniChar *ch*)

Validate an unicode character. This fucntion return true if succeeded, otherwise return false.

IMUniChar im_utf16_get_char (const IMUInt16 * *src*)

Get an unicode char from an utf8 char sequence.

IMSize im_utf16_string_to_ucs4 (IMUniChar * *dest*, IMSize *dest_size*, const IMUInt16 * *src*, IMInt *src_len*)

Convert an utf16 string to ucs4 string.

Parameters:

dest Buffer to store result ucs4 string.
dest_size Size of the dest buffer.
src Source utf8 string to be converted.
src_len Length of source utf8 string, in number of uint16 elements, -1 means until 0.

Returns:

How many unicode chars actually converted, if it's less than dest_size, then dest will be terminated by 0.

IMBool im_utf16_string_validate (const IMUInt16 * *str*, IMInt *maxlen*, const IMUInt16 ** *end*)

Validate a UTF16 string, return TRUE if valid, put pointer to first invalid char in **end.

IMSize im_utf16_strlen (const IMUInt16 * *src*, IMInt *maxlen*)

Return how many unicode chars in a given utf16 sequence.

Parameters:

src The utf16 sequence to be examined.
maxlen Maximum length of src, -1 means until 0.

IMUniChar im_utf8_get_char (const IMChar * *src*)

Get an unicode char from an utf8 char sequence.

IMSize im_utf8_string_to_ucs4 (IMUniChar * *dest*, IMSize *dest_size*, const IMChar * *src*, IMInt *src_len*)

Convert an utf8 string to ucs4 string.

Parameters:

dest Buffer to store result ucs4 string.
dest_size Size of the dest buffer.
src Source utf8 string to be converted.
src_len Length of source utf8 string, in number of bytes, -1 means until 0.

Returns:

How many unicode chars actually converted, if it's less than dest_size, then dest will be terminated by 0.

IMBool im_utf8_string_validate (const IMChar * *str*, IMInt *maxlen*, const IMChar ** *end*)

Validate a UTF8 string, return TRUE if valid, put pointer to first invalid char in **end.

IMSize im_utf8_strlen (const IMChar * *src*, IMInt *maxlen*)

Return how many unicode chars in an utf8 string.

Parameters:

src The utf8 string to be examined.
maxlen Maximum length of src, -1 means until 0.

Variable Documentation

const IMChar* const im_utf8_skip

Functions

106 IM_BEGIN_DECLS void **im_qsort_with_data** (**IMConstPointer** base, **IMSize** total_elems, **IMSize** size, **IMCompareDataFunc** compare_func, **IMPointer** user_data)

qsort an array, similar with qsort().

107 **IMPointer im_lower_bound** (**IMConstPointer** base, **IMSize** total_elems, **IMSize** size, **IMConstPointer** val, **IMCompareFunc** compare_func)

Finds the first position in which val could be inserted without changing the ordering.

108 **IMPointer im_upper_bound** (**IMConstPointer** base, **IMSize** total_elems, **IMSize** size, **IMConstPointer** val, **IMCompareFunc** compare_func)

Finds the last position in which val could be inserted without changing the ordering.

109 **IMPointer im_lower_bound_with_data** (**IMConstPointer** base, **IMSize** total_elems, **IMSize** size, **IMConstPointer** val, **IMCompareDataFunc** compare_func, **IMPointer** user_data)

Finds the first position in which val could be inserted without changing the ordering.

110 **IMPointer im_upper_bound_with_data** (**IMConstPointer** base, **IMSize** total_elems, **IMSize** size, **IMConstPointer** val, **IMCompareDataFunc** compare_func, **IMPointer** user_data)

Finds the last position in which val could be inserted without changing the ordering.

111 **IMUInt im_spaced_primes_closest** (**IMUInt** num)

Returns prime numbers spaced by approximately 1.5-2.0.

112 **IMUInt32 im_calculate_crc32** (const **IMChar** *buf, **IMSize** nbytes)

Calculate CRC32 checksum of a buffer.

113 **IMUInt64 im_get_current_time_in_milliseconds** ()

Function Documentation

IMUInt32 im_calculate_crc32 (const IMChar * *buf*, IMSize *nbytes*)

Calculate CRC32 checksum of a buffer.

IMUInt64 im_get_current_time_in_milliseconds ()

IMPointer im_lower_bound (IMConstPointer *base*, IMSize *total_elems*, IMSize *size*, IMConstPointer *val*, IMCompareFunc *compare_func*)

Finds the first position in which val could be inserted without changing the ordering.

Parameters:

base Pointer of the array.
total_elems Total elements in the array.
size Size of each element.
val The element to be inserted.
compare_func Compare function.

Returns:

A pointer to the first element "not less than" val, or end of the array if every element is less than val.

IMPointer im_lower_bound_with_data (IMConstPointer *base*, IMSize *total_elems*, IMSize *size*, IMConstPointer *val*, IMCompareDataFunc *compare_func*, IMPointer *user_data*)

Finds the first position in which val could be inserted without changing the ordering.

Parameters:

base Pointer of the array.
total_elems Total elements in the array.
size Size of each element.
val The element to be inserted.
compare_func Compare function.
user_data Arbitrary user data.

Returns:

A pointer to the first element "not less than" val, or end of the array if every element is less than val.

IM_BEGIN_DECLS void im_qsort_with_data (IMConstPointer *base*, IMSize *total_elems*, IMSize *size*, IMCompareDataFunc *compare_func*, IMPointer *user_data*)

qsort an array, similar with qsort().

Parameters:

base Pointer of the array.
total_elems Total elements in the array.
size Size of each element.
compare_func Compare function.
user_data Arbitrary user data.

im_qsort_with_data: : start of array to sort : elements in the array : size of each element : function to compare elements : data to pass to

This is just like the standard C qsort() function, but the comparison routine accepts a user data argument.

IMUInt im_spaced_primes_closest (IMUInt *num*)

Returns prime numbers spaced by approximately 1.5-2.0.

It's for use in resizing data structures which prefer prime-valued sizes, such as **IMHashTable**.

Returns:

the next largest prime, or the highest it knows about which is about MAXINT/4.

IMPointer im_upper_bound (IMConstPointer *base*, IMSize *total_elems*, IMSize *size*, IMConstPointer *val*, IMCompareFunc *compare_func*)

Finds the last position in which val could be inserted without changing the ordering.

Parameters:

base Pointer of the array.
total_elems Total elements in the array.
size Size of each element.
val The element to be inserted.
compare_func Compare function.

Returns:

A pointer to the first element greater than val, or end of the array if no elements are greater than val.

IMPointer im_upper_bound_with_data (IMConstPointer *base*, IMSize *total_elems*, IMSize *size*, IMConstPointer *val*, IMCompareDataFunc *compare_func*, IMPointer *user_data*)

Finds the last position in which val could be inserted without changing the ordering.

Parameters:

base Pointer of the array.
total_elems Total elements in the array.
size Size of each element.
val The element to be inserted.
compare_func Compare function.
user_data Arbitrary user data.

Returns:

A pointer to the first element greater than val, or end of the array if no elements are greater than val.

IMQuark

Functions

114 **IMQuark im_quark_try_string** (const **IMChar** *string)

Gets the IMQuark associated with the given string, or 0 if the string has no associated IMQuark.

115 **IMQuark im_quark_from_static_string** (const **IMChar** *string)

Gets the IMQuark identifying the given (static) string. If the string does not currently have an associated IMQuark, a new IMQuark is created, linked to the given string.

116 **IMQuark im_quark_from_string** (const **IMChar** *string)

Gets the IMQuark identifying the given string. If the string does not currently have an associated IMQuark, a new IMQuark is created, using a copy of the string.

117 const **IMChar** * **im_quark_to_string** (**IMQuark** quark)

Gets the string associated with the given IMQuark.

118 const **IMChar** * **im_intern_string** (const **IMChar** *string)

Returns a canonical representation for a string. Interned strings can be compared for equality by comparing the pointers, instead of using strcmp().

119 const **IMChar** * **im_intern_static_string** (const **IMChar** *string)

Returns a canonical representation for a string. Interned strings can be compared for equality by comparing the pointers, instead of using strcmp().

Variables

120 IM_BEGIN_DECLS typedef **IMUInt32 IMQuark**

IMQuark is used to associate strings to unique ids.

Function Documentation

const IMChar* im_intern_static_string (const IMChar * *string*)

Returns a canonical representation for a string. Interned strings can be compared for equality by comparing the pointers, instead of using strcmp().

im_intern_static_string() does not copy the string, therefore string must not be freed or modified.

Parameters:

string a string

Returns:

a canonical representation for the string.

const IMChar* im_intern_string (const IMChar * *string*)

Returns a canonical representation for a string. Interned strings can be compared for equality by comparing the pointers, instead of using strcmp().

Parameters:

string a string

Returns:
a canonical representation for the string.

IMQuark im_quark_from_static_string (const IMChar * *string*)

Gets the IMQuark identifying the given (static) string. If the string does not currently have an associated IMQuark, a new IMQuark is created, linked to the given string.

Note that this function is identical to **im_quark_from_string()** except that if a new IMQuark is created the string itself is used rather than a copy. This saves memory, but can only be used if the string will always exist. It can be used with statically allocated strings in the main program, but not with statically allocated memory in dynamically loaded modules, if you expect to ever unload the module again.

Parameters:
string a string.

Returns:
the IMQuark identifying the string.

IMQuark im_quark_from_string (const IMChar * *string*)

Gets the IMQuark identifying the given string. If the string does not currently have an associated IMQuark, a new IMQuark is created, using a copy of the string.

Parameters:
string a string.

Returns:
the IMQuark identifying the string.

const IMChar* im_quark_to_string (IMQuark *quark*)

Gets the string associated with the given IMQuark.

Parameters:
quark a IMQuark.

Returns:
the string associated with the IMQuark.

IMQuark im_quark_try_string (const IMChar * *string*)

Gets the IMQuark associated with the given string, or 0 if the string has no associated IMQuark.

If you want the IMQuark to be created if it doesn't already exist, use **im_quark_from_string()** or **im_quark_from_static_string()**

Parameters:
string a string.

Returns:
the IMQuark associated with the string, or 0 if there is no IMQuark associated with the string.

Variable Documentation

IM_BEGIN_DECLS typedef IMUInt32 IMQuark

IMQuark is used to associate strings to unique ids.

IMQuark

IMKeyCodes

Defines

121 #define **im_key_new**(code, mask) (((code) & IM_KEY_CODE_MASK) | ((mask) & IM_KEY_ALL_MASKS))

Create a key value from a key code and a mask.

122 #define **im_key_get_code**(val) ((val) & IM_KEY_CODE_MASK)

Get key code of a key value.

123 #define **im_key_get_masks**(val) ((val) & IM_KEY_ALL_MASKS)

Get key masks of a key value.

124 #define **im_key_check_mask**(val, mask) (((val) & (mask)) == (mask))

Check whether a mask is set for a key value or not.

125 #define **im_key_set_code**(val, code) ((val) = (im_key_get_masks(val) | ((code) & IM_KEY_CODE_MASK)))

Set the key code of a key value.

126 #define **im_key_set_mask**(val, mask) ((val) |= ((mask) & IM_KEY_ALL_MASKS))

Set a mask of a key value.

127 #define **im_key_clear_mask**(val, mask) ((val) &= ~((mask) & IM_KEY_ALL_MASKS))

Clear a mask of a key value.

128 #define **im_key_is_release**(val) im_key_check_mask((val),IM_KEY_RELEASE_MASK)

Check whether a key is a key release event.

129 #define **im_key_is_shift_down**(val) im_key_check_mask((val),IM_KEY_SHIFT_MASK)

130 #define **im_key_is_caps_lock_down**(val) im_key_check_mask((val),IM_KEY_CAPS_LOCK_MASK)

131 #define **im_key_is_ctrl_down**(val) im_key_check_mask((val),IM_KEY_CTRL_MASK)

132 #define **im_key_is_alt_down**(val) im_key_check_mask((val),IM_KEY_ALT_MASK)

133 #define **im_key_is_meta_down**(val) im_key_check_mask((val),IM_KEY_META_MASK)

134 #define **im_key_is_super_down**(val) im_key_check_mask((val),IM_KEY_SUPER_MASK)

135 #define **im_key_is_hyper_down**(val) im_key_check_mask((val),IM_KEY_HYPER_MASK)

136 #define **im_key_is_num_lock_down**(val) im_key_check_mask((val),IM_KEY_NUM_LOCK_MASK)

137 #define **im_key_is_unicode**(val)

Enumerations

138 enum **IMKeyMask – See Annex A for details.**

139 enum **IMKeyCode – See Annex A for details.**

Functions

140 **IMString** * **im_key_get_name** (**IMUInt32** keyval)

Get the name of a key value.

141 **IMUInt32 im_key_from_name** (const **IMString** *name)

Get a key value by parsing a specified key name.

142 **IMUInt32 im_key_code_to_x11keysym** (**IMUInt32** keycode)

Convert a key code to X11 keysym.

143 **IMUInt32 im_key_code_from_x11keysym** (**IMUInt32** x11keysym)

Convert a X11 keysym to key code.

Define Documentation

#define im_key_check_mask(val, mask) (((val) & (mask)) == (mask))

Check whether a mask is set for a key value or not.

#define im_key_clear_mask(val, mask) ((val) &= ~((mask) & IM_KEY_ALL_MASKS))

Clear a mask of a key value.

#define im_key_get_code(val) ((val) & IM_KEY_CODE_MASK)

Get key code of a key value.

#define im_key_get_masks(val) ((val) & IM_KEY_ALL_MASKS)

Get key masks of a key value.

#define im_key_is_alt_down(val) im_key_check_mask((val),IM_KEY_ALT_MASK)

#define im_key_is_caps_lock_down(val) im_key_check_mask((val),IM_KEY_CAPS_LOCK_MASK)

#define im_key_is_ctrl_down(val) im_key_check_mask((val),IM_KEY_CTRL_MASK)

#define im_key_is_hyper_down(val) im_key_check_mask((val),IM_KEY_HYPER_MASK)

#define im_key_is_meta_down(val) im_key_check_mask((val),IM_KEY_META_MASK)

#define im_key_is_num_lock_down(val) im_key_check_mask((val),IM_KEY_NUM_LOCK_MASK)

#define im_key_is_release(val) im_key_check_mask((val),IM_KEY_RELEASE_MASK)

Check whether a key is a key release event.

#define im_key_is_shift_down(val) im_key_check_mask((val),IM_KEY_SHIFT_MASK)

#define im_key_is_super_down(val) im_key_check_mask((val),IM_KEY_SUPER_MASK)

#define im_key_is_unicode(val)

Value:
```
((im_key_get_code(val) < IM_KEY_SPECIAL_START) ||        \
    ((im_key_get_code(val) > IM_KEY_SPECIAL_END)  &&      \
     (im_key_get_code(val) <= IM_KEY_MAX_VALID_CODE)))
```

#define im_key_new(code, mask) (((code) & IM_KEY_CODE_MASK) | ((mask) & IM_KEY_ALL_MASKS))

Create a key value from a key code and a mask.

#define im_key_set_code(val, code) ((val) = (im_key_get_masks(val) | ((code) & IM_KEY_CODE_MASK)))

Set the key code of a key value.

#define im_key_set_mask(val, mask) ((val) |= ((mask) & IM_KEY_ALL_MASKS))

Set a mask of a key value.

Enumeration Type Documentation

enum IMKeyCode

See Annex.A

enum IMKeyMask

A IMUInt32 is used to represent a key press or release code.

The format of a key code is: Bit 31 (1bit) : Key release mask. Set to indicate a key release event. Bit 21-30 (10bit): Modifiers' masks, such as shift, capslock, ctrl, alt, etc. Bit 0-20 (21bit): Key code.

Normally a Key code is a UCS-4 code (21bit). The key events that do not have corresponding UCS-4 code, such as modifier keys and function keys, are mapped into 0x110000-0x11ffff area to prevent from interfering with valid UCS-4 codes. All codes in 0x120000-0x1fffff are reserved for internal usage.

Enumerator:

IM_KEY_SHIFT_MASK
IM_KEY_CAPS_LOCK_MASK
IM_KEY_CTRL_MASK
IM_KEY_ALT_MASK
IM_KEY_META_MASK
IM_KEY_SUPER_MASK
IM_KEY_HYPER_MASK
IM_KEY_NUM_LOCK_MASK
IM_KEY_RELEASE_MASK
IM_KEY_ALL_MASKS

Function Documentation

IMUInt32 im_key_code_from_x11keysym (IMUInt32 *x11keysym*)

Convert a X11 keysym to key code.

IMUInt32 im_key_code_to_x11keysym (IMUInt32 *keycode*)

Convert a key code to X11 keysym.

IMUInt32 im_key_from_name (const IMString * *name*)

Get a key value by parsing a specified key name.

IMString* im_key_get_name (IMUInt32 *keyval*)

Get the name of a key value.

Returns:

A newly created **IMString** object, must be released by caller.

6.2.2 Object handling functions:

IMAttribute

Data Structures

144 struct **_IMAttribute**
145 struct **_IMAttributeClass**
146 struct **_IMAttrColor**
147 struct **_IMAttrInt**
148 struct **_IMAttrString**
149 struct **_IMAttrObject**

150 struct **IMAttribute**

It's the common base class of all text attribute types.

151 struct **IMAttrColor**

It is used to represent attributes that are colors.

152 struct **IMAttrInt**

It is used to represent attributes with an integer or enumeration value.

153 struct **IMAttrString**

It is used to represent attributes with a string value.

154 struct **IMAttrObject**

It is used to represent attributes with an object.

155 struct **IMAttrList**

A structure to hold a list of attributes that apply to a section of text.

156 struct **IMAttrIterator**

*A structure to represent an Iterator through a **IMAttrList**.*

Defines

157 #define **IM_ATTRIBUTE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTRIBUTE,**IMAttribute**))

cast the pointer to IMAttribute, return NULL if failed.*

158 #define **IM_CONST_ATTRIBUTE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTRIBUTE,**IMAttribute**))

cast the pointer to const IMAttribute, return NULL if failed.*

159 #define **IM_ATTRIBUTE_CLASS**(c)
(IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_ATTRIBUTE,**IMAttributeClass**))

cast the pointer to IMAttributeClass, return NULL if failed.*

160 #define **IM_IS_ATTRIBUTE**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTRIBUTE))

*tell if the pointer is an instance of **IMAttribute***

161 #define **IM_IS_ATTRIBUTE_CLASS**(c)
(IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_ATTRIBUTE))

tell if the pointer is an IMAttributeClass

162 #define **IM_ATTRIBUTE_GET_CLASS**(p)
(IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_ATTRIBUTE,**IMAttributeClass**))

*get the IMAttributeClass from an instance of **IMAttribute***

163 #define **IM_ATTRIBUTE_TYPE**(p) (im_object_get_type (p))

*get the type id from an instance of **IMAttribute***

164 #define **IM_ATTR_LIST**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_LIST,**IMAttrList**))

cast the pointer to IMAttrList, return NULL if failed.*

165 #define **IM_CONST_ATTR_LIST**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_LIST,**IMAttrList**))

cast the pointer to const IMAttrList, return NULL if failed.*

166 #define **IM_ATTR_LIST_CLASS**(c)
(IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_ATTR_LIST,**IMAttrListClass**))

cast the pointer to IMAttrListClass, return NULL if failed.*

167 #define **IM_IS_ATTR_LIST**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_LIST))

*tell if the pointer is an instance of **IMAttrList***

168 #define **IM_IS_ATTR_LIST_CLASS**(c)
(IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_ATTR_LIST))

tell if the pointer is an IMAttrListClass

169 #define **IM_ATTR_LIST_GET_CLASS**(p)
(IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_ATTR_LIST,**IMAttrListClass**))

*get the type id from an instance of **IMAttrList***

170 #define **IM_ATTR_LANGUAGE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_LANGUAGE,**IMAttrString**))

cast the pointer to a language attribute object, return NULL if failed.

171 #define **IM_ATTR_FONT_FAMILY**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_FAMILY,**IMAttrString**))

cast the pointer to a font family attribute object, return NULL if failed.

172 #define **IM_ATTR_FONT_STYLE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_STYLE,**IMAttrInt**))

cast the pointer to a font style attribute object, return NULL if failed.

173 #define **IM_ATTR_FONT_WEIGHT**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_WEIGHT,**IMAttrInt**))

cast the pointer to a font weight attribute object, return NULL if failed.

174 #define **IM_ATTR_FONT_SIZE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_SIZE,**IMAttrInt**))

cast the pointer to a font size attribute object, return NULL if failed.

175 #define **IM_ATTR_FONT_SCALE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_SCALE,**IMAttrInt**))

cast the pointer to a font scale attribute object, return NULL if failed.

176 #define **IM_ATTR_FOREGROUND**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FOREGROUND,**IMAttrColor**))

cast the pointer to a foreground attribute object, return NULL if failed.

177 #define **IM_ATTR_BACKGROUND**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_BACKGROUND,**IMAttrColor**))

cast the pointer to a background attribute object, return NULL if failed.

178 #define **IM_ATTR_UNDERLINE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_UNDERLINE,**IMAttrInt**))

cast the pointer to a underline attribute object, return NULL if failed.

179 #define **IM_ATTR_UNDERLINE_COLOR**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_UNDERLINE_COLOR,**IMAttrColor**)
)

cast the pointer to a underline color attribute object, return NULL if failed.

180 #define **IM_ATTR_STRIKETHROUGH**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_STRIKETHROUGH,**IMAttribute**))

cast the pointer to a strikethrough attribute object, return NULL if failed.

181 #define **IM_ATTR_STRIKETHROUGH_COLOR**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_STRIKETHROUGH_COLOR,**IMAttr
Color**))

cast the pointer to a strikethrough color attribute object, return NULL if failed.

182 #define **IM_ATTR_HIGHLIGHT**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_HIGHLIGHT,**IMAttribute**))

cast the pointer to a highlight attribute object, return NULL if failed.

183 #define **IM_ATTR_REVERSE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_REVERSE,**IMAttribute**))

cast the pointer to a reverse attribute object, return NULL if failed.

184 #define **IM_ATTR_STRING**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_STRING,**IMAttrString**))

cast the pointer to a string attribute object, return NULL if failed.

185 #define **IM_ATTR_OBJECT**(p)
(IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_OBJECT,**IMAttrObject**))

cast the pointer to an attribute object holds another object, return NULL if failed.

186 #define **IM_CONST_ATTR_LANGUAGE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_LANGUAGE,**IMAttrString**)
)

cast the pointer to a const reference of language attribute object, return NULL if failed.

187 #define **IM_CONST_ATTR_FONT_FAMILY**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_FAMILY,**IMAttrStrin**
g))

cast the pointer to a const reference of font family attribute object, return NULL if failed.

188 #define **IM_CONST_ATTR_FONT_STYLE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_STYLE,**IMAttrInt**))

cast the pointer to a const reference of font style attribute object, return NULL if failed.

189 #define **IM_CONST_ATTR_FONT_WEIGHT**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_WEIGHT,**IMAttrInt**)
)

cast the pointer to a const reference of font weight attribute object, return NULL if failed.

190 #define **IM_CONST_ATTR_FONT_SIZE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_SIZE,**IMAttrInt**))

cast the pointer to a const reference of font size attribute object, return NULL if failed.

191 #define **IM_CONST_ATTR_FONT_SCALE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_SCALE,**IMAttrInt**))

cast the pointer to a const reference of font scale attribute object, return NULL if failed.

192 #define **IM_CONST_ATTR_FOREGROUND**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FOREGROUND,**IMAttrCol**
or))

cast the pointer to a const reference of foreground attribute object, return NULL if failed.

193 #define **IM_CONST_ATTR_BACKGROUND**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_BACKGROUND,**IMAttrCol**
or))

cast the pointer to a const reference of background attribute object, return NULL if failed.

194 #define **IM_CONST_ATTR_UNDERLINE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_UNDERLINE,**IMAttrInt**))

cast the pointer to a const reference of underline attribute object, return NULL if failed.

195 #define **IM_CONST_ATTR_UNDERLINE_COLOR**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_UNDERLINE_COLOR,**IMA**
ttrColor))

cast the pointer to a const reference of underline color attribute object, return NULL if failed.

196 #define **IM_CONST_ATTR_STRIKETHROUGH**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_STRIKETHROUGH,**IMAttr**
ibute))

cast the pointer to a const reference of strikethrough attribute object, return NULL if failed.

197 #define **IM_CONST_ATTR_STRIKETHROUGH_COLOR**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_STRIKETHROUGH_COLO
R,**IMAttrColor**))

cast the pointer to a const reference of strikethrough color attribute object, return NULL if failed.

198 #define **IM_CONST_ATTR_HIGHLIGHT**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_HIGHLIGHT,**IMAttribute**))

cast the pointer to a const reference of highlight attribute object, return NULL if failed.

199 #define **IM_CONST_ATTR_REVERSE**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_REVERSE,**IMAttribute**))

cast the pointer to a const reference of reverse attribute object, return NULL if failed.

200 #define **IM_CONST_ATTR_STRING**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_STRING,**IMAttrString**))

cast the pointer to a const reference of string attribute object, return NULL if failed.

201 #define **IM_CONST_ATTR_OBJECT**(p)
(IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_OBJECT,**IMAttrObject**))

cast the pointer to a const reference of an attribute object holds another object, return NULL if failed.

202 #define **IM_IS_ATTR_LANGUAGE**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_LANGUAGE))

tell if the pointer is an instance of language attribute

203 #define **IM_IS_ATTR_FONT_FAMILY**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_FAMILY))

tell if the pointer is an instance of font family attribute

204 #define **IM_IS_ATTR_FONT_STYLE**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_STYLE))

tell if the pointer is an instance of font size attribute

205 #define **IM_IS_ATTR_FONT_WEIGHT**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_WEIGHT))

tell if the pointer is an instance of font weight attribute

206 #define **IM_IS_ATTR_FONT_SIZE**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_SIZE))

tell if the pointer is an instance of font size attribute

207 #define **IM_IS_ATTR_FONT_SCALE**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_SCALE))

tell if the pointer is an instance of font scale attribute

208 #define **IM_IS_ATTR_FOREGROUND**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FOREGROUND))

tell if the pointer is an instance of foreground attribute

209 #define **IM_IS_ATTR_BACKGROUND**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_BACKGROUND))

tell if the pointer is an instance of background attribute

210 #define **IM_IS_ATTR_UNDERLINE**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_UNDERLINE))

tell if the pointer is an instance of underline attribute

211 #define **IM_IS_ATTR_UNDERLINE_COLOR**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_UNDERLINE_COLOR))

tell if the pointer is an instance of underline color attribute

212 #define **IM_IS_ATTR_STRIKETHROUGH**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_STRIKETHROUGH))

tell if the pointer is an instance of strikethrough attribute

213 #define **IM_IS_ATTR_STRIKETHROUGH_COLOR**(p)
(IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_STRIKETHROUGH_COLOR))

tell if the pointer is an instance of strikethrough color attribute

214 #define **IM_IS_ATTR_HIGHLIGHT**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_HIGHLIGHT))

tell if the pointer is an instance of highlight attribute

215 #define **IM_IS_ATTR_REVERSE**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_REVERSE))

tell if the pointer is an instance of reverse attribute

216 #define **IM_IS_ATTR_STRING**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_STRING))

tell if the pointer is an instance of string attribute

217 #define **IM_IS_ATTR_OBJECT**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_OBJECT))

tell if the pointer is an instance of attribute object holds another object

218 #define **im_attribute_get_start**(attr) (((**IMAttribute***)(attr))->start)

helper macro to get start position of an attribute.

219 #define **im_attribute_get_end**(attr) (((**IMAttribute***)(attr))->end)

helper macro to get end position of an attribute.

220 #define **im_attribute_get_length**(attr) (im_attribute_end(attr) - im_attribute_start(attr))

helper macro to get length of an attribute.

Typedefs

221 typedef struct **_IMAttribute IMAttribute**

222 typedef struct **_IMAttributeClass IMAttributeClass**

223 typedef struct **_IMAttrColor IMAttrColor**

224 typedef struct **_IMAttrInt IMAttrInt**

225 typedef struct **_IMAttrString IMAttrString**

226 typedef struct **_IMAttrObject IMAttrObject**

227 typedef struct **_IMAttrList IMAttrList**

228 typedef struct **_IMAttrListClass IMAttrListClass**

229 typedef struct **_IMAttrIterator IMAttrIterator**

Enumerations

230 enum **IMAttrUnderline** { **IM_ATTR_UNDERLINE_NONE**, **IM_ATTR_UNDERLINE_SINGLE**, **IM_ATTR_UNDERLINE_DOUBLE**, **IM_ATTR_UNDERLINE_WAVE** }

Types of underline.

231 enum **IMAttrFontStyle** { **IM_ATTR_FONT_STYLE_NORMAL**, **IM_ATTR_FONT_STYLE_OBLIQUE**, **IM_ATTR_FONT_STYLE_ITALIC** }

Possible font styles.

232 enum **IMAttrFontWeight** { **IM_ATTR_FONT_WEIGHT_NORMAL**, **IM_ATTR_FONT_WEIGHT_BOLD** }

Possible font weights.

Functions

233 **IMAttribute** * **im_attribute_new** (**IMType** type)

*Create an empty **IMAttribute** object with a specific attribute type.*

234 void **im_attribute_clear** (**IMAttribute** *attr)

Clear the content of an attribute.

235 void **im_attribute_set_start** (**IMAttribute** *attr, **IMUInt** start)

Set the start position of an attribute.

236 void **im_attribute_set_end** (**IMAttribute** *attr, **IMUInt** end)

Set the end position of an attribute.

237 void **im_attribute_set_length** (**IMAttribute** *attr, **IMUInt** length)

Set the length of an attribute.

238 void **im_attribute_get_range** (const **IMAttribute** *attr, **IMUInt** *start, **IMUInt** *end)

Get the range of an attribute.

239 void **im_attribute_set_range** (**IMAttribute** *attr, **IMUInt** start, **IMUInt** end)

Set the range of an attribute.

240 **IMBool im_attribute_equal** (const **IMAttribute** *attr1, const **IMAttribute** *attr2)

Check whether two attributes are equal.

241 **IMAttribute** * **im_attr_language_new** (const **IMChar** *lang)

Create a new language attribute object.

242 const **IMChar** * **im_attr_language_get** (const **IMAttribute** *attr)

Get the language string from the given attribute object.

243 void **im_attr_language_set** (**IMAttribute** *attr, const **IMChar** *lang)

Set the language in the given attribute object.

244 void **im_attr_language_set_static** (**IMAttribute** *attr, const **IMChar** *lang)

Set the language in the given attribute object with a static string.

245 **IMAttribute** * **im_attr_font_family_new** (const **IMChar** *family)

Create a new font family attribute object.

246 const **IMChar** * **im_attr_font_family_get** (const **IMAttribute** *attr)

Get the font family string from the given attribute object.

247 void **im_attr_font_family_set** (**IMAttribute** *attr, const **IMChar** *family)

Set the font family in the given attribute object.

248 void **im_attr_font_family_set_static** (**IMAttribute** *attr, const **IMChar** *family)

Set the font family in the given attribute object with a static string.

249 **IMAttribute** * **im_attr_font_style_new** (**IMAttrFontStyle** style)

Create a new font style attribute object.

250 **IMAttrFontStyle im_attr_font_style_get** (const **IMAttribute** *attr)

Get the font style from the given attribute object.

251 void **im_attr_font_style_set** (**IMAttribute** *attr, **IMAttrFontStyle** style)

Set the font style in the given attribute object.

252 **IMAttribute** * **im_attr_font_weight_new** (**IMAttrFontWeight** weight)

Create a new font weight attribute object.

253 **IMAttrFontWeight im_attr_font_weight_get** (const **IMAttribute** *attr)

Get the font weight from the given attribute object.

254 void **im_attr_font_weight_set** (**IMAttribute** *attr, **IMAttrFontWeight** weight)

Set the font weight in the given attribute object.

255 **IMAttribute** * **im_attr_font_size_new** (**IMInt** size)

Create a new font size attribute object.

256 **IMInt im_attr_font_size_get** (const **IMAttribute** *attr)

Get the font size from the given attribute object.

257 void **im_attr_font_size_set** (**IMAttribute** *attr, **IMInt** size)

Set the font size in the given attribute object.

258 **IMAttribute** * **im_attr_font_scale_new** (**IMInt** scale_factor)

Create a new font scale attribute object.

259 **IMInt im_attr_font_scale_get** (const **IMAttribute** *attr)

Get the font scale factor from the given attribute object.

260 void **im_attr_font_scale_set** (**IMAttribute** *attr, **IMInt** scale_factor)

Set the font scale factor in the given attribute object.

261 **IMAttribute** * **im_attr_foreground_new** (**IMUInt16** red, **IMUInt16** green, **IMUInt16** blue)

Create a new foreground attribute object with RGB color.

262 void **im_attr_foreground_get** (const **IMAttribute** *attr, **IMUInt16** *red, **IMUInt16** *green, **IMUInt16** *blue)

Get the foreground color (in RGB) from the given attribute object.

263 void **im_attr_foreground_set** (**IMAttribute** *attr, **IMUInt16** red, **IMUInt16** green, **IMUInt16** blue)

Set the foreground color (in RGB) in the given attribute object.

264 **IMAttribute** * **im_attr_background_new** (**IMUInt16** red, **IMUInt16** green, **IMUInt16** blue)

Create a background attribute object with RGB color.

265 void **im_attr_background_get** (const **IMAttribute** *attr, **IMUInt16** *red, **IMUInt16** *green, **IMUInt16** *blue)

Get the background color (in RGB) from the given attribute object.

266 void **im_attr_background_set** (**IMAttribute** *attr, **IMUInt16** red, **IMUInt16** green, **IMUInt16** blue)

Set the background color (in RGB) in the given attribute object.

267 **IMAttribute** * **im_attr_underline_new** (**IMAttrUnderline** type)

Create a new underline attribute object.

268 **IMAttrUnderline im_attr_underline_get** (const **IMAttribute** *attr)

Get the underline type from the given attribute object.

269 void **im_attr_underline_set** (**IMAttribute** *attr, **IMAttrUnderline** type)

Set the underline type in the given attribute object.

270 **IMAttribute** * **im_attr_underline_color_new** (**IMUInt16** red, **IMUInt16** green, **IMUInt16** blue)

Create a new underline color attribute object.

271 void **im_attr_underline_color_get** (const **IMAttribute** *attr, **IMUInt16** *red, **IMUInt16** *green, **IMUInt16** *blue)

Get the underline color (in RGB) from the given attribute object.

272 void **im_attr_underline_color_set** (**IMAttribute** *attr, **IMUInt16** red, **IMUInt16** green, **IMUInt16** blue)

Set the underline color (in RGB) in the given attribute object.

273 **IMAttribute** * **im_attr_strikethrough_new** ()

Create a new strikethrough attribute object.

274 **IMAttribute** * **im_attr_strikethrough_color_new** (**IMUInt16** red, **IMUInt16** green, **IMUInt16** blue)

Create a new strikethrough color attribute object.

275 void **im_attr_strikethrough_color_get** (const **IMAttribute** *attr, **IMUInt16** *red, **IMUInt16** *green, **IMUInt16** *blue)

Get the strikethrough color (in RGB) from the given attribute object.

276 void **im_attr_strikethrough_color_set** (**IMAttribute** *attr, **IMUInt16** red, **IMUInt16** green, **IMUInt16** blue)

Set the strikethrough color (in RGB) in the given attribute object.

277 **IMAttribute** * **im_attr_highlight_new** ()

Create a new hightlight attribute object.

278 **IMAttribute** * **im_attr_reverse_new** ()

Create a new reverse attribute object.

279 **IMAttribute** * **im_attr_string_new** (const **IMChar** *str)

Create a new string attribute object.

280 const **IMChar** * **im_attr_string_get** (const **IMAttribute** *attr)

Get the string from the given attribute object.

281 void **im_attr_string_set** (**IMAttribute** *attr, const **IMChar** *str)

Set the string in the given attribute object.

282 void **im_attr_string_set_static** (**IMAttribute** *attr, const **IMChar** *str)

Set the string in the given attribute object with a static string.

283 **IMAttribute** * **im_attr_object_new** (**IMPointer** object)

Create a new attribute which holds an object.

284 **IMPointer im_attr_object_get** (const **IMAttribute** *attr)

Get the object hold by an attribute.

285 **IMPointer im_attr_object_dup** (const **IMAttribute** *attr)

Return a clone of the object hold by this attribute.

286 void **im_attr_object_set** (**IMAttribute** *attr, **IMPointer** object)

287 **IMAttrList** * **im_attr_list_new** (void)

Create an empty attribute list object.

288 void **im_attr_list_clear** (**IMAttrList** *list)

Remove all attributes hold by the list.

289 **IMBool im_attr_list_empty** (const **IMAttrList** *list)

*Check if a **IMAttrList** object is empty.*

290 void **im_attr_list_get_range** (const **IMAttrList** *list, **IMUInt** *start, **IMUInt** *end)

Get the range covered by all attributes in a list.

291 void **im_attr_list_insert** (**IMAttrList** *list, **IMAttribute** *attr)

*Insert the given attribute into the **IMAttrList**.*

292 void **im_attr_list_insert_before** (**IMAttrList** *list, **IMAttribute** *attr)

*Insert the given attribute into the **IMAttrList**.*

293 void **im_attr_list_change** (**IMAttrList** *list, **IMAttribute** *attr)

*Insert the given attribute into the **IMAttrList** while maintaining consistency of the list.*

294 void **im_attr_list_splice** (**IMAttrList** *list, **IMAttrList** *other, **IMUInt** pos, **IMUInt** len)

Splices attribute list other into list.

295 void **im_attr_list_stretch** (**IMAttrList** *list, **IMUInt** pos, **IMUInt** len)

Stretch a list at given position with given amount of length.

296 void **im_attr_list_shrink** (**IMAttrList** *list, **IMUInt** pos, **IMUInt** len)

Shrink a list at given position with given amount of length.

297 void **im_attr_list_clear_range** (**IMAttrList** *list, **IMUInt** pos, **IMUInt** len)

Clear a given range of a list.

298 void **im_attr_list_remove** (**IMAttrList** *list, **IMUInt** pos, **IMUInt** len, **IMType** attrtype)

Delete all attributes that match given type in given range.

299 **IMAttrList** * **im_attr_list_get_sub_list** (const **IMAttrList** *list, **IMUInt** pos, **IMUInt** len)

*Get a sub portion of an **IMAttrList**.*

300 void **im_attr_list_foreach** (const **IMAttrList** *list, **IMFunc** func, **IMPointer** user_data)

Calls the given function against all attributes in a list one by one.

301 void **im_attr_list_optimize** (**IMAttrList** *list)

Optimize a list.

302 **IMAttrIterator** * **im_attr_list_get_iterator** (**IMAttrList** *list)

Create an iterator for a given list points to the first element.

303 **IMAttrIterator** * **im_attr_iterator_clone** (const **IMAttrIterator** *iterator)

*Clone an **IMAttrIterator**.*

304 void **im_attr_iterator_destroy** (**IMAttrIterator** *iterator)

*Destroy an **IMAttrIterator**.*

305 **IMBool im_attr_iterator_next** (**IMAttrIterator** *iterator)

Advance an list iterator to the next attribute in the list.

306 **IMAttribute** * **im_attr_iterator_get** (const **IMAttrIterator** *iterator)

Get the attribute pointed by an list iterator.

307 **IMBool im_attr_iterator_valid** (const **IMAttrIterator** *iterator)

Check whether an list iterator is valid or not.

308 **IMBool im_attr_iterator_equal** (const **IMAttrIterator** *iterator1, const **IMAttrIterator** *iterator2)

Check whether two iterators point to the same attribute.

309 void **im_attr_iterator_remove** (**IMAttrIterator** *iterator)

Remove an attribute pointed by a given iterator.

Define Documentation

#define IM_ATTR_BACKGROUND(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_BACKGROUND,IMAttrColor))

cast the pointer to a background attribute object, return NULL if failed.

#define IM_ATTR_FONT_FAMILY(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_FAMILY,IMAttrString))

cast the pointer to a font family attribute object, return NULL if failed.

#define IM_ATTR_FONT_SCALE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_SCALE,IMAttrInt))

cast the pointer to a font scale attribute object, return NULL if failed.

#define IM_ATTR_FONT_SIZE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_SIZE,IMAttrInt))

cast the pointer to a font size attribute object, return NULL if failed.

#define IM_ATTR_FONT_STYLE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_STYLE,IMAttrInt))

cast the pointer to a font style attribute object, return NULL if failed.

#define IM_ATTR_FONT_WEIGHT(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FONT_WEIGHT,IMAttrInt))

cast the pointer to a font weight attribute object, return NULL if failed.

#define IM_ATTR_FOREGROUND(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_FOREGROUND,IMAttrColor))

cast the pointer to a foreground attribute object, return NULL if failed.

#define IM_ATTR_HIGHLIGHT(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_HIGHLIGHT,IMAttribute))

cast the pointer to a highlight attribute object, return NULL if failed.

#define IM_ATTR_LANGUAGE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_LANGUAGE,IMAttrString))

cast the pointer to a language attribute object, return NULL if failed.

#define IM_ATTR_LIST(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_LIST,IMAttrList))

cast the pointer to IMAttrList*, return NULL if failed.

#define IM_ATTR_LIST_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_ATTR_LIST,IMAttrListClass))

cast the pointer to IMAttrListClass*, return NULL if failed.

#define IM_ATTR_LIST_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_ATTR_LIST,IMAttrListClass))

get the type id from an instance of **IMAttrList**

#define IM_ATTR_OBJECT(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_OBJECT,IMAttrObject))

cast the pointer to an attribute object holds another object, return NULL if failed.

#define IM_ATTR_REVERSE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_REVERSE,IMAttribute))

cast the pointer to a reverse attribute object, return NULL if failed.

#define IM_ATTR_STRIKETHROUGH(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_STRIKETHROUGH,IMAttribute))

cast the pointer to a strikethrough attribute object, return NULL if failed.

#define IM_ATTR_STRIKETHROUGH_COLOR(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_STRIKETHROUGH_COLOR,IMAttrColor))

cast the pointer to a strikethrough color attribute object, return NULL if failed.

#define IM_ATTR_STRING(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_STRING,IMAttrString))

cast the pointer to a string attribute object, return NULL if failed.

#define IM_ATTR_UNDERLINE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_UNDERLINE,IMAttrInt))

cast the pointer to a underline attribute object, return NULL if failed.

#define IM_ATTR_UNDERLINE_COLOR(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTR_UNDERLINE_COLOR,IMAttrColor))

cast the pointer to a underline color attribute object, return NULL if failed.

#define IM_ATTRIBUTE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_ATTRIBUTE,IMAttribute))

cast the pointer to IMAttribute*, return NULL if failed.

#define IM_ATTRIBUTE_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_ATTRIBUTE,IMAttributeClass))

cast the pointer to IMAttributeClass*, return NULL if failed.

#define IM_ATTRIBUTE_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_ATTRIBUTE,IMAttributeClass))

get the IMAttributeClass from an instance of **IMAttribute**

```
#define im_attribute_get_end(attr)  (((IMAttribute*)(attr))->end)
```

helper macro to get end position of an attribute.

```
#define im_attribute_get_length(attr)  (im_attribute_end(attr) - im_attribute_start(attr))
```

helper macro to get length of an attribute.

```
#define im_attribute_get_start(attr)  (((IMAttribute*)(attr))->start)
```

helper macro to get start position of an attribute.

```
#define IM_ATTRIBUTE_TYPE(p)  (im_object_get_type (p))
```

get the type id from an instance of **IMAttribute**

```
#define IM_CONST_ATTR_BACKGROUND(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_BACKGROUND,IMAttrColor))
```

cast the pointer to a const reference of background attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_FONT_FAMILY(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_FAMILY,IMAttrString))
```

cast the pointer to a const reference of font family attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_FONT_SCALE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_SCALE,IMAttrInt))
```

cast the pointer to a const reference of font scale attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_FONT_SIZE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_SIZE,IMAttrInt))
```

cast the pointer to a const reference of font size attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_FONT_STYLE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_STYLE,IMAttrInt))
```

cast the pointer to a const reference of font style attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_FONT_WEIGHT(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FONT_WEIGHT,IMAttrInt))
```

cast the pointer to a const reference of font weight attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_FOREGROUND(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_FOREGROUND,IMAttrColor))
```

cast the pointer to a const reference of foreground attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_HIGHLIGHT(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_HIGHLIGHT,IMAttribute))
```

cast the pointer to a const reference of highlight attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_LANGUAGE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_LANGUAGE,IMAttrString))
```

cast the pointer to a const reference of language attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_LIST(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_LIST,IMAttrList))
```

cast the pointer to const IMAttrList*, return NULL if failed.

```
#define IM_CONST_ATTR_OBJECT(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_OBJECT,IMAttrObject))
```

cast the pointer to a const reference of an attribute object holds another object, return NULL if failed.

```
#define IM_CONST_ATTR_REVERSE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_REVERSE,IMAttribute))
```

cast the pointer to a const reference of reverse attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_STRIKETHROUGH(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_STRIKETHROUGH,IMAttribute))
```

cast the pointer to a const reference of strikethrough attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_STRIKETHROUGH_COLOR(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_STRIKETHROUGH_COLOR,IMAttrColor))
```

cast the pointer to a const reference of strikethrough color attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_STRING(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_STRING,IMAttrString))
```

cast the pointer to a const reference of string attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_UNDERLINE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_UNDERLINE,IMAttrInt))
```

cast the pointer to a const reference of underline attribute object, return NULL if failed.

```
#define IM_CONST_ATTR_UNDERLINE_COLOR(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTR_UNDERLINE_COLOR,IMAttrColor))
```

cast the pointer to a const reference of underline color attribute object, return NULL if failed.

```
#define IM_CONST_ATTRIBUTE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_ATTRIBUTE,IMAttribute))
```

cast the pointer to const IMAttribute*, return NULL if failed.

```
#define IM_IS_ATTR_BACKGROUND(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_BACKGROUND))
```

tell if the pointer is an instance of background attribute

```
#define IM_IS_ATTR_FONT_FAMILY(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_FAMILY))
```

tell if the pointer is an instance of font family attribute

```
#define IM_IS_ATTR_FONT_SCALE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_SCALE))
```

tell if the pointer is an instance of font scale attribute

```
#define IM_IS_ATTR_FONT_SIZE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_SIZE))
```

tell if the pointer is an instance of font size attribute

```
#define IM_IS_ATTR_FONT_STYLE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_STYLE))
```

tell if the pointer is an instance of font size attribute

```
#define IM_IS_ATTR_FONT_WEIGHT(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FONT_WEIGHT))
```

tell if the pointer is an instance of font weight attribute

```
#define IM_IS_ATTR_FOREGROUND(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_FOREGROUND))
```

tell if the pointer is an instance of foreground attribute

#define IM_IS_ATTR_HIGHLIGHT(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_HIGHLIGHT))

tell if the pointer is an instance of highlight attribute

#define IM_IS_ATTR_LANGUAGE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_LANGUAGE))

tell if the pointer is an instance of language attribute

#define IM_IS_ATTR_LIST(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_LIST))

tell if the pointer is an instance of **IMAttrList**

#define IM_IS_ATTR_LIST_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_ATTR_LIST))

tell if the pointer is an IMAttrListClass

#define IM_IS_ATTR_OBJECT(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_OBJECT))

tell if the pointer is an instance of attribute object holds another object

#define IM_IS_ATTR_REVERSE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_REVERSE))

tell if the pointer is an instance of reverse attribute

#define IM_IS_ATTR_STRIKETHROUGH(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_STRIKETHROUGH))

tell if the pointer is an instance of strikethrough attribute

#define IM_IS_ATTR_STRIKETHROUGH_COLOR(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_STRIKETHROUGH_COLOR))

tell if the pointer is an instance of strikethrough color attribute

#define IM_IS_ATTR_STRING(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_STRING))

tell if the pointer is an instance of string attribute

#define IM_IS_ATTR_UNDERLINE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_UNDERLINE))

tell if the pointer is an instance of underline attribute

#define IM_IS_ATTR_UNDERLINE_COLOR(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTR_UNDERLINE_COLOR))

tell if the pointer is an instance of underline color attribute

#define IM_IS_ATTRIBUTE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_ATTRIBUTE))

tell if the pointer is an instance of **IMAttribute**

#define IM_IS_ATTRIBUTE_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_ATTRIBUTE))

tell if the pointer is an IMAttributeClass

Typedef Documentation

typedef struct _IMAttrColor IMAttrColor

typedef struct _IMAttribute IMAttribute

typedef struct _IMAttributeClass IMAttributeClass

typedef struct _IMAttrInt IMAttrInt

typedef struct _IMAttrIterator IMAttrIterator

typedef struct _IMAttrList IMAttrList

typedef struct _IMAttrListClass IMAttrListClass

typedef struct _IMAttrObject IMAttrObject

typedef struct _IMAttrString IMAttrString

Enumeration Type Documentation

enum IMAttrFontStyle

Possible font styles.

310 IM_ATTR_FONT_STYLE_NORMAL normal font style
311 IM_ATTR_FONT_STYLE_OBLIQUE slanted with roman style
312 IM_ATTR_FONT_STYLE_ITALIC slanted with italic style

Enumerator:

IM_ATTR_FONT_STYLE_NORMAL
IM_ATTR_FONT_STYLE_OBLIQUE
IM_ATTR_FONT_STYLE_ITALIC

enum IMAttrFontWeight

Possible font weights.

313 IM_ATTR_FONT_WEIGHT_NORMAL normal weight
314 IM_ATTR_FONT_WEIGHT_BOLD bold weight

Enumerator:

IM_ATTR_FONT_WEIGHT_NORMAL
IM_ATTR_FONT_WEIGHT_BOLD

enum IMAttrUnderline

Types of underline.

315 IM_ATTR_UNDERLINE_SINGLE single underline
316 IM_ATTR_UNDERLINE_DOUBLE double underline
317 IM_ATTR_UNDERLINE_WAVE wavy underline

Enumerator:

IM_ATTR_UNDERLINE_NONE
IM_ATTR_UNDERLINE_SINGLE
IM_ATTR_UNDERLINE_DOUBLE
IM_ATTR_UNDERLINE_WAVE

Function Documentation

void im_attr_background_get (const IMAttribute * *attr*, IMUInt16 * *red*, IMUInt16 * *green*, IMUInt16 * *blue*)

Get the background color (in RGB) from the given attribute object. See [SRGB]

IMAttribute* im_attr_background_new (IMUInt16 *red*, IMUInt16 *green*, IMUInt16 *blue*)

Create a background attribute object with RGB color. See [SRGB]

void im_attr_background_set (IMAttribute * *attr*, IMUInt16 *red*, IMUInt16 *green*, IMUInt16 *blue*)

Set the background color (in RGB) in the given attribute object. See [SRGB]

const IMChar* im_attr_font_family_get (const IMAttribute * *attr*)

Get the font family string from the given attribute object.

The returned font family string should not be released by caller.

IMAttribute* im_attr_font_family_new (const IMChar * *family*)

Create a new font family attribute object.

void im_attr_font_family_set (IMAttribute * *attr*, const IMChar * *family*)

Set the font family in the given attribute object.

In the attribute object, the original font family string would be released, and replaced by a new copy of the given string.

void im_attr_font_family_set_static (IMAttribute * *attr*, const IMChar * *family*)

Set the font family in the given attribute object with a static string.

IMInt im_attr_font_scale_get (const IMAttribute * *attr*)

Get the font scale factor from the given attribute object.

IMAttribute* im_attr_font_scale_new (IMInt *scale_factor*)

Create a new font scale attribute object.

void im_attr_font_scale_set (IMAttribute * *attr*, IMInt *scale_factor*)

Set the font scale factor in the given attribute object.

IMInt im_attr_font_size_get (const IMAttribute * *attr*)

Get the font size from the given attribute object.

IMAttribute* im_attr_font_size_new (IMInt *size*)

Create a new font size attribute object.

void im_attr_font_size_set (IMAttribute * *attr*, IMInt *size*)

Set the font size in the given attribute object.

IMAttrFontStyle im_attr_font_style_get (const IMAttribute * *attr*)

Get the font style from the given attribute object.

IMAttribute* im_attr_font_style_new (IMAttrFontStyle *style*)

Create a new font style attribute object.

void im_attr_font_style_set (IMAttribute * *attr*, IMAttrFontStyle *style*)

Set the font style in the given attribute object.

IMAttrFontWeight im_attr_font_weight_get (const IMAttribute * *attr*)

Get the font weight from the given attribute object.

IMAttribute* im_attr_font_weight_new (IMAttrFontWeight *weight*)

Create a new font weight attribute object.

void im_attr_font_weight_set (IMAttribute * *attr*, IMAttrFontWeight *weight*)

Set the font weight in the given attribute object.

void im_attr_foreground_get (const IMAttribute * *attr*, IMUInt16 * *red*, IMUInt16 * *green*, IMUInt16 * *blue*)

Get the foreground color (in RGB) from the given attribute object. See [SRGB]

IMAttribute* im_attr_foreground_new (IMUInt16 *red*, IMUInt16 *green*, IMUInt16 *blue*)

Create a new foreground attribute object with RGB color. See [SRGB]

void im_attr_foreground_set (IMAttribute * *attr*, IMUInt16 *red*, IMUInt16 *green*, IMUInt16 *blue*)

Set the foreground color (in RGB) in the given attribute object. See [SRGB]

IMAttribute* im_attr_highlight_new ()

Create a new hightlight attribute object.

IMAttrIterator* im_attr_iterator_clone (const IMAttrIterator * *iterator*)

Clone an **IMAttrIterator**.

void im_attr_iterator_destroy (IMAttrIterator * *iterator*)

Destroy an **IMAttrIterator**.

IMBool im_attr_iterator_equal (const IMAttrIterator * *iterator1*, const IMAttrIterator * *iterator2*)

Check whether two iterators point to the same attribute.

This fucntion return true if succeeded, otherwise return false.

IMAttribute* im_attr_iterator_get (const IMAttrIterator * *iterator*)

Get the attribute pointed by an list iterator.

IMBool im_attr_iterator_next (IMAttrIterator * *iterator*)

Advance an list iterator to the next attribute in the list.

This fucntion return true if succeeded, otherwise return false.

void im_attr_iterator_remove (IMAttrIterator * *iterator*)

Remove an attribute pointed by a given iterator.

Parameters:

iterator an iterator points to an attribute.

IMBool im_attr_iterator_valid (const IMAttrIterator * *iterator*)

Check whether an list iterator is valid or not.

This fucntion return true if succeeded, otherwise return false.

const IMChar* im_attr_language_get (const IMAttribute * *attr*)

Get the language string from the given attribute object.

The returned language string should not be released by caller.

IMAttribute* im_attr_language_new (const IMChar * *lang*)

Create a new language attribute object.

void im_attr_language_set (IMAttribute * *attr*, const IMChar * *lang*)

Set the language in the given attribute object.

In the attribute object, the original language string would be released, and replaced by a new copy of the given string.

void im_attr_language_set_static (IMAttribute * *attr*, const IMChar * *lang*)

Set the language in the given attribute object with a static string.

void im_attr_list_change (IMAttrList * *list*, IMAttribute * *attr*)

Insert the given attribute into the **IMAttrList** while maintaining consistency of the list.

It will replace any attributes of the same type on that segment and be merged with any adjoining attributes that are identical.

This function is slower than **im_attr_list_insert()** for creating a attribute list in order (potentially much slower for large lists). However, **im_attr_list_insert()** is not suitable for continually changing a set of attributes since it never removes or combines existing attributes.

void im_attr_list_clear (IMAttrList * *list*)

Remove all attributes hold by the list.

void im_attr_list_clear_range (IMAttrList * *list*, IMUInt *pos*, IMUInt *len*)

Clear a given range of a list.

All attributes in this range will be removed/clipped accordingly. Total range of the list won't be affected, unless the clear range is at the end of list range.

Parameters:

list an **IMAttrList**.
pos the position of the range to be cleared.
len the length of the range to be cleared.

IMBool im_attr_list_empty (const IMAttrList * *list*)

Check if a **IMAttrList** object is empty. This fucntion return true if succeeded, otherwise return false.

void im_attr_list_foreach (const IMAttrList * *list*, IMFunc *func*, IMPointer *user_data*)

Calls the given function against all attributes in a list one by one.

IMAttrIterator* im_attr_list_get_iterator (IMAttrList * *list*)

Create an iterator for a given list points to the first element.

void im_attr_list_get_range (const IMAttrList * *list*, IMUInt * *start*, IMUInt * *end*)

Get the range covered by all attributes in a list.

IMAttrList* im_attr_list_get_sub_list (const IMAttrList * *list*, IMUInt *pos*, IMUInt *len*)

Get a sub portion of an **IMAttrList**.

Parameters:

list an **IMAttrList**.
pos the position of sub list to be get.
len the length of sub lits to be get.

Returns:

the newly created **IMAttrList** object which holds the specific sub portion of original list. It should be released by caller.

void im_attr_list_insert (IMAttrList * *list*, IMAttribute * *attr*)

Insert the given attribute into the **IMAttrList**.

It will be inserted after all other attributes with a matching

Parameters:

list a **IMAttrList**
attr the attribute to insert. Ownership of this attribute is assumed by the list.

void im_attr_list_insert_before (IMAttrList * *list*, IMAttribute * *attr*)

Insert the given attribute into the **IMAttrList**.

It will be inserted before all other attributes with a matching

Parameters:

list a **IMAttrList**
attr the attribute to insert. Ownership of this attribute is assumed by the list.

IMAttrList* im_attr_list_new (void)

Create an empty attribute list object.

void im_attr_list_optimize (IMAttrList * *list*)

Optimize a list.

It actully just uses **im_attr_list_change()** to re-insert all attributes in the list.

void im_attr_list_remove (IMAttrList * *list*, IMUInt *pos*, IMUInt *len*, IMType *attrtype*)

Delete all attributes that match given type in given range.

Attributes that match given attrtype in this range will be removed/clipped accordingly.

Parameters:

list an **IMAttrList**.
pos the position of the range to be deleted.
len the length of the range to be deleted.
attrtype The type of attribute to be deleted.

void im_attr_list_shrink (IMAttrList * *list*, IMUInt *pos*, IMUInt *len*)

Shrink a list at given position with given amount of length.

All attributes will be moved/removed/clipped accordingly. Total range of the list will be truncated by len.

Parameters:

list an **IMAttrList**.
pos the position of the range to be shrunk.
len the length of the range to be shrunk.

void im_attr_list_splice (IMAttrList * *list*, IMAttrList * *other*, IMUInt *pos*, IMUInt *len*)

Splices attribute list other into list.

This operation is equivalent to stretching every attribute that applies at position pos in list by an amount len, by calling **im_attr_list_stretch()**, and then calling **im_attr_list_change()** with a copy of each attribute in other in sequence (offset in position by pos).

Parameters:

list a **IMAttrList**
other another **IMAttrList**
pos the position in list at which to insert other
len the length of the spliced segment. (Note that this must be specified since the attributes in other may only be present at some subsection of this range)

void im_attr_list_stretch (IMAttrList * *list*, IMUInt *pos*, IMUInt *len*)

Stretch a list at given position with given amount of length.

All attributes at right side of the pos will be moved right accordingly. All attributes which include the pos will be stretched accordingly.

IMPointer im_attr_object_dup (const IMAttribute * *attr*)

Return a clone of the object hold by this attribute.

The returned object must be released with im_object_unref.

IMPointer im_attr_object_get (const IMAttribute * *attr*)

Get the object hold by an attribute.

The returned object is owned by the attribute, it shouldn't be modified or released by caller.

IMAttribute* im_attr_object_new (IMPointer *object*)

Create a new attribute which holds an object.

Ownership of given object will be assumed by the newly created attribute object.

void im_attr_object_set (IMAttribute * *attr*, IMPointer *object*)

IMAttribute* im_attr_reverse_new ()

Create a new reverse attribute object.

void im_attr_strikethrough_color_get (const IMAttribute * *attr*, IMUInt16 * *red*, IMUInt16 * *green*, IMUInt16 * *blue*)

Get the strikethrough color (in RGB) from the given attribute object. See [SRGB]

IMAttribute* im_attr_strikethrough_color_new (IMUInt16 *red*, IMUInt16 *green*, IMUInt16 *blue*)

Create a new strikethrough color attribute object.

void im_attr_strikethrough_color_set (IMAttribute * *attr*, IMUInt16 *red*, IMUInt16 *green*, IMUInt16 *blue*)

Set the strikethrough color (in RGB) in the given attribute object. See [SRGB]

IMAttribute* im_attr_strikethrough_new ()

Create a new strikethrough attribute object.

const IMChar* im_attr_string_get (const IMAttribute * *attr*)

Get the string from the given attribute object.

The returned string should not be released by caller.

IMAttribute* im_attr_string_new (const IMChar * *str*)

Create a new string attribute object.

void im_attr_string_set (IMAttribute * *attr*, const IMChar * *str*)

Set the string in the given attribute object.

In the attribute object, the original string would be released, and replaced by a new copy of the given string.

void im_attr_string_set_static (IMAttribute * *attr*, const IMChar * *str*)

Set the string in the given attribute object with a static string.

void im_attr_underline_color_get (const IMAttribute * *attr*, IMUInt16 * *red*, IMUInt16 * *green*, IMUInt16 * *blue*)

Get the underline color (in RGB) from the given attribute object. See [SRGB]

IMAttribute* im_attr_underline_color_new (IMUInt16 *red*, IMUInt16 *green*, IMUInt16 *blue*)

Create a new underline color attribute object. See [SRGB]

void im_attr_underline_color_set (IMAttribute * *attr*, IMUInt16 *red*, IMUInt16 *green*, IMUInt16 *blue*)

Set the underline color (in RGB) in the given attribute object. See [SRGB]

IMAttrUnderline im_attr_underline_get (const IMAttribute * *attr*)

Get the underline type from the given attribute object.

IMAttribute* im_attr_underline_new (IMAttrUnderline *type*)

Create a new underline attribute object.

void im_attr_underline_set (IMAttribute * *attr*, IMAttrUnderline *type*)

Set the underline type in the given attribute object.

void im_attribute_clear (IMAttribute * *attr*)

Clear the content of an attribute.

IMBool im_attribute_equal (const IMAttribute * *attr1*, const IMAttribute * *attr2*)

Check whether two attributes are equal. This fucntion return true if succeeded, otherwise return false.

Returns:

TRUE if the two attributes are the same type, and have the same value.

Only content of the attributes will be compared. Their range will not be taken into account.

void im_attribute_get_range (const IMAttribute * *attr*, IMUInt * *start*, IMUInt * *end*)

Get the range of an attribute.

IMAttribute* im_attribute_new (IMType *type*)

Create an empty **IMAttribute** object with a specific attribute type.

Parameters:

type The type of newly created attribute object, it must be derived from IM_TYPE_ATTRIBUTE.

void im_attribute_set_end (IMAttribute * *attr*, IMUInt *end*)

Set the end position of an attribute.

void im_attribute_set_length (IMAttribute * *attr*, IMUInt *length*)

Set the length of an attribute.

The end position of this attribute will be adjusted accordingly.

void im_attribute_set_range (IMAttribute * *attr*, IMUInt *start*, IMUInt *end*)

Set the range of an attribute.

void im_attribute_set_start (IMAttribute * *attr*, IMUInt *start*)

Set the start position of an attribute.

IMEvent

Data Structures

318 struct **_IMEventInfo**

319 struct **IMEvent**

***IMEvent** class is used to hold specified information.*

320 struct **IMEventInfo**

This structure holds various information of an predefined event type. Such as number data and the type of each data that the event should contain.

Defines

321 #define **IM_EVENT**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_EVENT,**IMEvent**))

cast the pointer to IMEvent, return NULL if failed.*

322 #define **IM_CONST_EVENT**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_EVENT,**IMEvent**))

cast the pointer to const IMEvent, return NULL if failed.*

323 #define **IM_EVENT_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_EVENT,**IMEventClass**))

cast the pointer to IMEventClass, return NULL if failed.*

324 #define **IM_IS_EVENT**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_EVENT))

*tell if the pointer is an instance of **IMEvent**.*

325 #define **IM_IS_EVENT_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_EVENT))

tell if the pointer is an IMEventClass.

326 #define **IM_EVENT_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_EVENT,**IMEventClass**))

*get the type id from an instance of **IMEvent**.*

Typedefs

327 typedef struct **_IMEvent IMEvent**

328 typedef struct **_IMEventClass IMEventClass**

329 typedef struct **_IMEventInfo IMEventInfo**

Enumerations

330 enum { **IM_EVENT_SOURCE_IMBUS** = 0, **IM_EVENT_SOURCE_INVALID** = IM_MAXUINT32 }

331 enum { **IM_EVENT_DEST_IMBUS** = 0, **IM_EVENT_DEST_DEFAULT** = 0, **IM_EVENT_DEST_BROADCAST** = IM_MAXUINT32 }

332 enum { **IM_EVENT_INPUT_CONTEXT_NONE** = 0, **IM_EVENT_INPUT_CONTEXT_FOCUSED** = IM_MAXUINT32 }

Functions

333 **IMEvent** * **im_event_new** (**IMUInt32** type, **IMBool** reply)

*Create a new **IMEvent** object with the given type.*

334 **IMEvent** * **im_event_new_full** (**IMUInt32** type, **IMUInt32** src_id, **IMUInt32** dest_id, **IMUInt32** ic_id, **IMBool** observable, **IMBool** reply)

*Create a new **IMEvent** object with full parameters.*

335 void **im_event_clear_data** (**IMEvent** *event)

Clear the data of an event.

336 void **im_event_set_type** (**IMEvent** *event, **IMUInt32** type)

Set event type of an event object.

337 **IMUInt32 im_event_get_type** (const **IMEvent** *event)

Get event type of an event object.

338 void **im_event_set_source** (**IMEvent** *event, **IMUInt32** src_id)

Set source id of an event object.

339 **IMUInt32 im_event_get_source** (const **IMEvent** *event)

Get source id of an event object.

340 void **im_event_set_destination** (**IMEvent** *event, **IMUInt32** dest_id)

Set destination id of an event object.

341 **IMUInt32 im_event_get_destination** (const **IMEvent** *event)

Get destination id of an event object.

342 void **im_event_set_serial_number** (**IMEvent** *event, **IMUInt32** serial)

Set serial number of an event object.

343 **IMUInt32 im_event_get_serial_number** (const **IMEvent** *event)

Get serial number of an event object.

344 void **im_event_set_input_context_id** (**IMEvent** *event, **IMUInt32** icid)

Set InputContext ID of an event object.

345 **IMUInt32 im_event_get_input_context_id** (const **IMEvent** *event)

Get InputContext ID of an event object.

346 void **im_event_update_timestamp** (**IMEvent** *event)

Update timestamp of an event object to the current time.

347 void **im_event_set_timestamp** (**IMEvent** *event, **IMUInt64** timestamp)

Set timestamp of an event object to a specified time. in milliseconds.

348 **IMUInt64 im_event_get_timestamp** (const **IMEvent** *event)

Get timestamp of an event object.

349 void **im_event_set_observable** (**IMEvent** *event, **IMBool** observable)

Set observable flag of an event object.

350 **IMBool im_event_get_observable** (const **IMEvent** *event)

Get observable flag of an event object.

351 void **im_event_set_is_reply** (**IMEvent** *event, **IMBool** reply)

Set is reply flag of an event object.

352 **IMBool im_event_get_is_reply** (const **IMEvent** *event)

Get is reply flag of an event object.

353 **IMSize im_event_get_n_data** (const **IMEvent** *event)

Get the number of data that a specified event contains.

354 **IMType im_event_get_data_type** (const **IMEvent** *event, **IMSize** index)

Get the type of a data in an event at specified index.

355 **IMType im_event_get_data** (const **IMEvent** *event, **IMSize** index, **IMValue** *value)

Get a data in an event at specified index.

356 void **im_event_append_data** (**IMEvent** *event, **IMValue** *value)

*Append a data ,which is stored in a specified **IMValue** object, into a specified event.*

357 void **im_event_set_data** (**IMEvent** *event, **IMSize** index, **IMValue** *value)

Set a data in an event at specified index.

358 **IMChar im_event_get_char** (const **IMEvent** *event, **IMSize** index)

get a char at the given index in the data of an event

359 **IMUChar im_event_get_uchar** (const **IMEvent** *event, **IMSize** index)

get an unsigned char at the given index in the data of an event

360 **IMBool im_event_get_bool** (const **IMEvent** *event, **IMSize** index)

get a boolean at the given index in the data of an event

361 **IMInt16 im_event_get_int16** (const **IMEvent** *event, **IMSize** index)

get an int16 (short) at the given index in the data of an event

362 **IMUInt16 im_event_get_uint16** (const **IMEvent** *event, **IMSize** index)

get an unsigned int16 (short) at the given index in the data of an event

363 **IMInt32 im_event_get_int32** (const **IMEvent** *event, **IMSize** index)

get an int32 (long) at the given index in the data of an event

364 **IMUInt32 im_event_get_uint32** (const **IMEvent** *event, **IMSize** index)

get an unsigned int32 (long) at the given index in the data of an event

365 **IMInt64 im_event_get_int64** (const **IMEvent** *event, **IMSize** index)

get an int64 (long long) at the given index in the data of an event

366 **IMUInt64 im_event_get_uint64** (const **IMEvent** *event, **IMSize** index)

get an unsigned int64 (long long) at the given index in the data of an event

367 **IMDouble im_event_get_double** (const **IMEvent** *event, **IMSize** index)

get a doulbe value at the given index in the data of an event

368 const **IMChar** * **im_event_get_c_string** (const **IMEvent** *event, **IMSize** index)

get a c-string at the given index in the data of an event

369 **IMChar** * **im_event_dup_c_string** (const **IMEvent** *event, **IMSize** index)

get a duplicated c-string at the given index in the data of an event

370 **IMPointer im_event_get_object** (const **IMEvent** *event, **IMSize** index)

get an object at the given index in the data of an event

371 **IMPointer im_event_dup_object** (const **IMEvent** *event, **IMSize** index)

get a duplicated object at the given index in the data of an event

372 **IMBool im_event_validate** (const **IMEvent** *event)

Validate the type of each data contained by an event according to the event type id, to see whether they conform to the standard of predefined event types.

Define Documentation

#define IM_CONST_EVENT(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_EVENT,IMEvent))

cast the pointer to const IMEvent*, return NULL if failed.

#define IM_EVENT(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_EVENT,IMEvent))

cast the pointer to IMEvent*, return NULL if failed.

#define IM_EVENT_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_EVENT,IMEventClass))

cast the pointer to IMEventClass*, return NULL if failed.

#define IM_EVENT_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_EVENT,IMEventClass))

get the type id from an instance of **IMEvent**.

#define IM_IS_EVENT(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_EVENT))

tell if the pointer is an instance of **IMEvent**.

#define IM_IS_EVENT_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_EVENT))

tell if the pointer is an IMEventClass.

Typedef Documentation

typedef struct _IMEvent IMEvent

typedef struct _IMEventClass IMEventClass

typedef struct _IMEventInfo IMEventInfo

Enumeration Type Documentation

anonymous enum

Enumerator:

IM_EVENT_SOURCE_IMBUS
IM_EVENT_SOURCE_INVALID

anonymous enum

Enumerator:

IM_EVENT_DEST_IMBUS
IM_EVENT_DEST_DEFAULT
IM_EVENT_DEST_BROADCAST

anonymous enum

Enumerator:

IM_EVENT_INPUT_CONTEXT_NONE
IM_EVENT_INPUT_CONTEXT_FOCUSED

Function Documentation

void im_event_append_data (IMEvent * *event*, IMValue * *value*)

Append a data ,which is stored in a specified **IMValue** object, into a specified event.

void im_event_clear_data (IMEvent * *event*)

Clear the data of an event.

All fixed event information won't be cleared. Only the arbitrary data of the event will be cleared.

IMChar* im_event_dup_c_string (const IMEvent * *event*, IMSize *index*)

get a duplicated c-string at the given index in the data of an event

IMPointer im_event_dup_object (const IMEvent * *event*, IMSize *index*)

get a duplicated object at the given index in the data of an event

IMBool im_event_get_bool (const IMEvent * *event*, IMSize *index*)

get a boolean at the given index in the data of an event

const IMChar* im_event_get_c_string (const IMEvent * *event*, IMSize *index*)

get a c-string at the given index in the data of an event

IMChar im_event_get_char (const IMEvent * *event*, IMSize *index*)

get a char at the given index in the data of an event

IMType im_event_get_data (const IMEvent * *event*, IMSize *index*, IMValue * *value*)

Get a data in an event at specified index.

The data will be stored into given **IMValue** object, and the data type will be returned.

IMType im_event_get_data_type (const IMEvent * *event*, IMSize *index*)

Get the type of a data in an event at specified index.

IMUInt32 im_event_get_destination (const IMEvent * *event*)

Get destination id of an event object.

IMDouble im_event_get_double (const IMEvent * *event*, IMSize *index*)

get a doulbe value at the given index in the data of an event

IMUInt32 im_event_get_input_context_id (const IMEvent * *event*)

Get InputContext ID of an event object.

IMInt16 im_event_get_int16 (const IMEvent * *event*, IMSize *index*)

get an int16 (short) at the given index in the data of an event

IMInt32 im_event_get_int32 (const IMEvent * *event*, IMSize *index*)

get an int32 (long) at the given index in the data of an event

IMInt64 im_event_get_int64 (const IMEvent * *event*, IMSize *index*)

get an int64 (long long) at the given index in the data of an event

IMBool im_event_get_is_reply (const IMEvent * *event*)

Get is reply flag of an event object.

IMSize im_event_get_n_data (const IMEvent * *event*)

Get the number of data that a specified event contains.

IMPointer im_event_get_object (const IMEvent * *event*, IMSize *index*)

get an object at the given index in the data of an event

IMBool im_event_get_observable (const IMEvent * *event*)

Get observable flag of an event object.

IMUInt32 im_event_get_serial_number (const IMEvent * *event*)

Get serial number of an event object.

IMUInt32 im_event_get_source (const IMEvent * *event*)

Get source id of an event object.

IMUInt64 im_event_get_timestamp (const IMEvent * *event*)

Get timestamp of an event object.

IMUInt32 im_event_get_type (const IMEvent * *event*)

Get event type of an event object.

IMUChar im_event_get_uchar (const IMEvent * *event*, IMSize *index*)

get an unsigned char at the given index in the data of an event

IMUInt16 im_event_get_uint16 (const IMEvent * *event*, IMSize *index*)

get an unsigned int16 (short) at the given index in the data of an event

IMUInt32 im_event_get_uint32 (const IMEvent * *event*, IMSize *index*)

get an unsigned int32 (long) at the given index in the data of an event

IMUInt64 im_event_get_uint64 (const IMEvent * *event*, IMSize *index*)

get an unsigned int64 (long long) at the given index in the data of an event

IMEvent* im_event_new (IMUInt32 *type*, IMBool *reply*)

Create a new **IMEvent** object with the given type.

Parameters:

type event type id

Returns:

the new **IMEvent**

IMEvent* im_event_new_full (IMUInt32 *type*, IMUInt32 *src_id*, IMUInt32 *dest_id*, IMUInt32 *ic_id*, IMBool *observable*, IMBool *reply*)

Create a new **IMEvent** object with full parameters.

Parameters:

type event type id
src_id source id
dest_id destination id
ic_id InputContext id

Returns:

the new **IMEvent**

void im_event_set_data (IMEvent * *event*, IMSize *index*, IMValue * *value*)

Set a data in an event at specified index.

void im_event_set_destination (IMEvent * *event*, IMUInt32 *dest_id*)

Set destination id of an event object.

void im_event_set_input_context_id (IMEvent * *event*, IMUInt32 *icid*)

Set InputContext ID of an event object.

void im_event_set_is_reply (IMEvent * *event*, IMBool *reply*)

Set is reply flag of an event object.

void im_event_set_observable (IMEvent * *event*, IMBool *observable*)

Set observable flag of an event object.

void im_event_set_source (IMEvent * *event*, IMUInt32 *src_id*)

Set source id of an event object.

void im_event_set_timestamp (IMEvent * *event*, IMUInt64 *timestamp*)

Set timestamp of an event object to a specified time. in milliseconds.

void im_event_set_type (IMEvent * *event*, IMUInt32 *type*)

Set event type of an event object.

All data owned by the event will be cleared,

void im_event_update_timestamp (IMEvent * *event*)

Update timestamp of an event object to the current time.

IMBool im_event_validate (const IMEvent * *event*)

Validate the type of each data contained by an event according to the event type id, to see whether they conform to the standard of predefined event types. This fucntion return true if succeeded, otherwise return false.

Data Structures

373 struct **IMEventRoles**

Class to hold one or more role informations of arbitrary event types.

Defines

374 #define **IM_EVENT_ROLES**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_EVENT_ROLES,**IMEventRoles**))

cast the pointer to IMEventRoles, return NULL if failed.*

375 #define **IM_CONST_EVENT_ROLES**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_EVENT_ROLES,**IMEventRoles**))

cast the pointer to const IMEventRoles, return NULL if failed.*

376 #define **IM_EVENT_ROLES_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_EVENT_ROLES,**IMEventRolesClass**))

cast the pointer to IMEventRolesClass, return NULL if failed.*

377 #define **IM_IS_EVENT_ROLES**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_EVENT_ROLES))

*tell if the pointer is an instance of **IMEventRoles**.*

378 #define **IM_IS_EVENT_ROLES_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_EVENT_ROLES))

tell if the pointer is IMEventRolesClass.

379 #define **IM_EVENT_ROLES_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_EVENT_ROLES,**IMEventRolesClass**))

*get the type id from an instance of **IMEventRoles**.*

Typedefs

380 typedef struct **_IMEventRoles IMEventRoles**

381 typedef struct **_IMEventRolesClass IMEventRolesClass**

Enumerations

382 enum **IMEventRoleType** { **IM_EVENT_PRODUCER** = (1 << 0), **IM_EVENT_CONSUMER** = (1 << 1), **IM_EVENT_OBSERVER** = (1 << 2) }

Functions

383 **IMEventRoles** * **im_event_roles_new** ()

*Create a new **IMEventRoles** object.*

384 void **im_event_roles_clear** (**IMEventRoles** *event_roles)

*Clear a specified **IMEventRoles** object.*

385 void **im_event_roles_set** (**IMEventRoles** *event_roles, **IMUInt32** event_type, **IMUInt** roles)

*Set the roles of a specified event type in a **IMEventRoles** object.*

386 void **im_event_roles_add** (**IMEventRoles** *event_roles, **IMUInt32** event_type, **IMUInt** roles)

*Add a role to a specified event type in a **IMEventRoles** object.*

387 void **im_event_roles_remove** (**IMEventRoles** *event_roles, **IMUInt32** event_type)

Remove role information of a event type.

388 **IMUInt im_event_roles_get** (const **IMEventRoles** *event_roles, **IMUInt32** event_type)

Get roles for a specified event type.

389 **IMBool im_event_roles_check** (const **IMEventRoles** *event_roles, **IMUInt32** event_type, **IMEventRoleType** role)

Check whether a role is set to a specified event type.

Define Documentation

#define IM_CONST_EVENT_ROLES(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_EVENT_ROLES,IMEventRoles))

cast the pointer to const IMEventRoles*, return NULL if failed.

#define IM_EVENT_ROLES(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_EVENT_ROLES,IMEventRoles))

cast the pointer to IMEventRoles*, return NULL if failed.

#define IM_EVENT_ROLES_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_EVENT_ROLES,IMEventRolesClass))

cast the pointer to IMEventRolesClass*, return NULL if failed.

#define IM_EVENT_ROLES_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_EVENT_ROLES,IMEventRolesClass))

get the type id from an instance of **IMEventRoles**.

#define IM_IS_EVENT_ROLES(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_EVENT_ROLES))

tell if the pointer is an instance of **IMEventRoles**.

#define IM_IS_EVENT_ROLES_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_EVENT_ROLES))

tell if the pointer is IMEventRolesClass.

Typedef Documentation

typedef struct _IMEventRoles IMEventRoles

typedef struct _IMEventRolesClass IMEventRolesClass

Enumeration Type Documentation

enum IMEventRoleType

Enumerator:

IM_EVENT_PRODUCER
IM_EVENT_CONSUMER
IM_EVENT_OBSERVER

Function Documentation

void im_event_roles_add (IMEventRoles * *event_roles*, IMUInt32 *event_type*, IMUInt *roles*)

Add a role to a specified event type in a **IMEventRoles** object.

The new role will be added to the specified event type. Any available roles for the event won't be touched.

Parameters:

event_type Type id of the event.
role Type of the role.

IMBool im_event_roles_check (const IMEventRoles * *event_roles*, IMUInt32 *event_type*, IMEventRoleType *role*)

Check whether a role is set to a specified event type.

Parameters:

event_type Type id of the event.
role Type of the role to be checked.

Returns:

True if the role is set to the event.

void im_event_roles_clear (IMEventRoles * *event_roles*)

Clear a specified **IMEventRoles** object.

IMUInt im_event_roles_get (const IMEventRoles * *event_roles*, IMUInt32 *event_type*)

Get roles for a specified event type.

Parameters:

event_type Type id of the event.

Returns:

The bit masks of available roles, 0 if there is no role set to this event.

IMEventRoles* im_event_roles_new ()

Create a new **IMEventRoles** object.

void im_event_roles_remove (IMEventRoles * *event_roles*, IMUInt32 *event_type*)

Remove role information of a event type.

All role information of specified event type will be removed.

void im_event_roles_set (IMEventRoles * *event_roles*, IMUInt32 *event_type*, IMUInt *roles*)

Set the roles of a specified event type in a **IMEventRoles** object.

Old roles information for that event type will be replaced by the new one.

Parameters:

event_type Type id of the event.
roles Roles, bit mask of one or more of IM_EVENT_PRODUCER, IM_EVENT_CONSUMER or IM_EVENT_OBSERVER.

IMEventTypes

Enumerations

```
390 enum { IM_EVENT_INVALID = 0, IM_EVENT_REGISTER_COMPONENT,
    IM_EVENT_DEREGISTER_COMPONENT, IM_EVENT_QUERY_COMPONENTS,
    IM_EVENT_UPDATE_EVENT_ROLES, IM_EVENT_REGISTER_PROPERTIES,
    IM_EVENT_DEREGISTER_PROPERTIES, IM_EVENT_UPDATE_PROPERTIES,
    IM_EVENT_QUERY_PROPERTIES, IM_EVENT_REGISTER_HOTKEY_PROFILES,
```

IM_EVENT_DEREGISTER_HOTKEY_PROFILES,
IM_EVENT_SET_ACTIVE_HOTKEY_PROFILES,
IM_EVENT_UNSET_ACTIVE_HOTKEY_PROFILES, **IM_EVENT_SET_ACTIVE_COMPONENT**,
IM_EVENT_GET_ACTIVE_COMPONENT, **IM_EVENT_CREATE_INPUT_CONTEXT**,
IM_EVENT_DELETE_INPUT_CONTEXT, **IM_EVENT_QUERY_INPUT_CONTEXTS**,
IM_EVENT_FOCUS_IN_INPUT_CONTEXT, **IM_EVENT_FOCUS_OUT_INPUT_CONTEXT**,
IM_EVENT_RESET_INPUT_CONTEXT,
IM_EVENT_SET_INPUT_CONTEXT_IME_COMPONENT,
IM_EVENT_SWITCH_INPUT_CONTEXT_IME_COMPONENT,
IM_EVENT_PROCESS_KEY_EVENT, **IM_EVENT_TRIGGER_PROPERTY**,
IM_EVENT_CLICK_ON_PREEDIT, **IM_EVENT_CLICK_ON_CANDIDATE**,
IM_EVENT_CLICK_ON_LOOKUP_TABLE_NEXT_PAGE,
IM_EVENT_CLICK_ON_LOOKUP_TABLE_PREV_PAGE,
IM_EVENT_CLICK_ON_LOOKUP_TABLE_SCROLL_BAR,
IM_EVENT_ADJUST_LOOKUP_TABLE_PAGE_SIZE,
IM_EVENT_UPDATE_SPOT_LOCATION, **IM_EVENT_UPDATE_SCREEN**,
IM_EVENT_SHOW_PREEDIT_STRING, **IM_EVENT_HIDE_PREEDIT_STRING**,
IM_EVENT_UPDATE_PREEDIT_STRING, **IM_EVENT_SHOW_AUXILIARY_STRING**,
IM_EVENT_HIDE_AUXILIARY_STRING, **IM_EVENT_UPDATE_AUXILIARY_STRING**,
IM_EVENT_SHOW_LOOKUP_TABLE, **IM_EVENT_HIDE_LOOKUP_TABLE**,
IM_EVENT_UPDATE_LOOKUP_TABLE, **IM_EVENT_SHOW_COMPONENTS_LIST**,
IM_EVENT_COMMIT_STRING, **IM_EVENT_FORWARD_KEY_EVENT**,
IM_EVENT_GET_SURROUNDING_TEXT, **IM_EVENT_DELETE_SURROUNDING_TEXT**,
IM_EVENT_STORAGE_STORE, **IM_EVENT_STORAGE_RETRIEVE**,
IM_EVENT_STORAGE_REMOVE, **IM_EVENT_STORAGE_LIST**,
IM_EVENT_STORAGE_ADD_WATCH, **IM_EVENT_STORAGE_REMOVE_WATCH**,
IM_EVENT_STORAGE_WATCH_NOTIFY, **IM_EVENT_STORAGE_FLUSH**,
IM_EVENT_STORAGE_RELOAD, **IM_EVENT_SET_TIMEOUT_EVENT**,
IM_EVENT_CANCEL_TIMEOUT_EVENT, **IM_EVENT_USER_DEFINED_START** = 32768 }

Enumeration Type Documentation

anonymous enum

Enumerator:

IM_EVENT_INVALID Indicates invalid event.

IM_EVENT_REGISTER_COMPONENT Register a component into IMFramework

An IM client should send an event with this type as soon as it connects to IMFramework. If the client has more than one components, then multiple events with this type should be sent to IMFramework to register each component.

Source ID: shall be IM_EVENT_SOURCE_INVALID Dest ID: shall be IM_EVENT_DEST_IMBUS ICID: shall be IM_EVENT_INPUT_CONTEXT_NONE

Data:

391 IMComponentInfo object containing necessary information of the component.

Has reply event: Yes Reply data:

392 IMUInt32 Unique ID for corresponding component, allocated by IMFramework. If IMFramework failed to process this event, then IM_EVENT_SOURCE_INVALID will be returned.

IM_EVENT_DEREGISTER_COMPONENT Deregister a component from IMFramework

An event with this type shall be sent to IMFramework if an IM client wants to deregister its component from IMFramework.

The source component, which sends this event, will be deregistered by IMFramework.

Source ID: ID of the component, which sends this event Dest ID: shall be IM_EVENT_DEST_IMBUS ICID: shall be IM_EVENT_INPUT_CONTEXT_NONE

Data: no data Has reply event: No

IM_EVENT_QUERY_COMPONENTS Query information of one or more components

An event with this type shall be sent to IMFramework, if an IM client wants to query the information of one or more components registered in IMFramework.

Source ID: ID of the component, which sends this event Dest ID: shall be IM_EVENT_DEST_IMBUS ICID: shall be IM_EVENT_INPUT_CONTEXT_NONE

Data: One of following data types:

393 IMUInt32 (one or more) One or more component IDs to be queried. If want to query the information of all components, IM_EVENT_DEST_BROADCAST shall be used.

394 **IMEventRoles** object Query the information of all components which support the specified event roles.

Has reply event: Yes

395 IMComponentInfo object (one or more) The information of each component that meet the criteria. If the specified component ID is invalid, then an IMBool with value FALSE will be returned.

IM_EVENT_UPDATE_EVENT_ROLES Update event roles supported by a component.

IM_EVENT_REGISTER_PROPERTIES Register properties of a component

IM_EVENT_DEREGISTER_PROPERTIES Deregister properties of a component

IM_EVENT_UPDATE_PROPERTIES Update information of one or more properties

IM_EVENT_QUERY_PROPERTIES Query information of all properties of a component

IM_EVENT_REGISTER_HOTKEY_PROFILES Register hotkey profiles of a component

IM_EVENT_DEREGISTER_HOTKEY_PROFILES Deregister hotkey profiles of a component

IM_EVENT_SET_ACTIVE_HOTKEY_PROFILES Set active local and/or global hotkey profiles

IM_EVENT_UNSET_ACTIVE_HOTKEY_PROFILES Unset active local and/or global hotkey profiles

IM_EVENT_SET_ACTIVE_COMPONENT Set active component for specified event roles

IM_EVENT_GET_ACTIVE_COMPONENT Get active component for specified event roles

IM_EVENT_CREATE_INPUT_CONTEXT Create an input context with active component which support this event

IM_EVENT_DELETE_INPUT_CONTEXT Delete an input context

IM_EVENT_QUERY_INPUT_CONTEXTS Query information of one or more input contexts

IM_EVENT_FOCUS_IN_INPUT_CONTEXT Focus in an input context

IM_EVENT_FOCUS_OUT_INPUT_CONTEXT Focus out an input context

IM_EVENT_RESET_INPUT_CONTEXT Reset an input context

IM_EVENT_SET_INPUT_CONTEXT_IME_COMPONENT Set the component which is bound to specified input contexts

IM_EVENT_SWITCH_INPUT_CONTEXT_IME_COMPONENT Switch the component which is bound to specified input contexts between the default and active one.

IM_EVENT_PROCESS_KEY_EVENT A key event

IM_EVENT_TRIGGER_PROPERTY Trigger a property

IM_EVENT_CLICK_ON_PREEDIT Click on the preedit string

IM_EVENT_CLICK_ON_CANDIDATE Click on a candidate

IM_EVENT_CLICK_ON_LOOKUP_TABLE_NEXT_PAGE Click on next page button of lookup table

IM_EVENT_CLICK_ON_LOOKUP_TABLE_PREV_PAGE Click on previous page button of lookup table

IM_EVENT_CLICK_ON_LOOKUP_TABLE_SCROLL_BAR Click on scroll bar of lookup table

IM_EVENT_ADJUST_LOOKUP_TABLE_PAGE_SIZE Page size of lookup table was adjusted by UI component

IM_EVENT_UPDATE_SPOT_LOCATION
IM_EVENT_UPDATE_SCREEN
IM_EVENT_SHOW_PREEDIT_STRING

IM_EVENT_HIDE_PREEDIT_STRING
IM_EVENT_UPDATE_PREEDIT_STRING
IM_EVENT_SHOW_AUXILIARY_STRING
IM_EVENT_HIDE_AUXILIARY_STRING
IM_EVENT_UPDATE_AUXILIARY_STRING
IM_EVENT_SHOW_LOOKUP_TABLE
IM_EVENT_HIDE_LOOKUP_TABLE
IM_EVENT_UPDATE_LOOKUP_TABLE
IM_EVENT_SHOW_COMPONENTS_LIST
IM_EVENT_COMMIT_STRING
IM_EVENT_FORWARD_KEY_EVENT
IM_EVENT_GET_SURROUNDING_TEXT
IM_EVENT_DELETE_SURROUNDING_TEXT
IM_EVENT_STORAGE_STORE
IM_EVENT_STORAGE_RETRIEVE
IM_EVENT_STORAGE_REMOVE
IM_EVENT_STORAGE_LIST
IM_EVENT_STORAGE_ADD_WATCH
IM_EVENT_STORAGE_REMOVE_WATCH
IM_EVENT_STORAGE_WATCH_NOTIFY
IM_EVENT_STORAGE_FLUSH
IM_EVENT_STORAGE_RELOAD
IM_EVENT_SET_TIMEOUT_EVENT Set an event which will be sent out after a certain timeout
IM_EVENT_CANCEL_TIMEOUT_EVENT Cancel a timeout event
IM_EVENT_USER_DEFINED_START

IMHotkey and IMHotkeyProfile

Data Structures

396 struct **IMHotkey**

An object to associate a key code to one or more action events.

397 struct **IMHotkeyProfile**

*An object to hold a set of **IMHotkey** objects.*

Defines

398 #define **IM_HOTKEY**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_HOTKEY,**IMHotkey**))

399 #define **IM_CONST_HOTKEY**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_HOTKEY,**IMHotkey**))

400 #define **IM_HOTKEY_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_HOTKEY,**IMHotkeyClass**))

401 #define **IM_IS_HOTKEY**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_HOTKEY))

402 #define **IM_IS_HOTKEY_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_HOTKEY))

403 #define **IM_HOTKEY_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_HOTKEY,**IMHotkeyClass**))

404 #define **IM_HOTKEY_PROFILE**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_HOTKEY_PROFILE,**IMHotkeyProfile**))

405 #define **IM_CONST_HOTKEY_PROFILE**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_HOTKEY_PROFILE,**IMHotkeyProfile**))

406 #define **IM_HOTKEY_PROFILE_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_HOTKEY_PROFILE,**IMHotkeyProfileClass**))

407 #define **IM_IS_HOTKEY_PROFILE**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_HOTKEY_PROFILE))

408 #define **IM_IS_HOTKEY_PROFILE_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_HOTKEY_PROFILE))

409 #define **IM_HOTKEY_PROFILE_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_HOTKEY_PROFILE,**IMHotkeyProfileClass**))

Typedefs

410 typedef struct **_IMHotkey IMHotkey**

411 typedef struct **_IMHotkeyClass IMHotkeyClass**

412 typedef struct **_IMHotkeyProfile IMHotkeyProfile**

413 typedef struct **_IMHotkeyProfileClass IMHotkeyProfileClass**

Enumerations

414 enum **IMHotkeyProfileScope** { **IM_HOTKEY_PROFILE_LOCAL** = 0, **IM_HOTKEY_PROFILE_GLOBAL** = 1 }

Indicate the effect scope of a hotkey profile.

Functions

415 **IMHotkey** * **im_hotkey_new** (**IMUInt32** keycode)

Create a new hotkey with specified key code.

416 void **im_hotkey_set_key_code** (**IMHotkey** *hotkey, **IMUInt32** keycode)

Set key code of a hotkey.

417 **IMUInt32 im_hotkey_get_key_code** (const **IMHotkey** *hotkey)

*Get key code of a specified **IMHotkey** object.*

418 **IMSize im_hotkey_get_n_events** (const **IMHotkey** *hotkey)

*Get how many events stored in a specified **IMHotkey** object.*

419 void **im_hotkey_clear** (**IMHotkey** *hotkey)

*Clear all events stored in a specified **IMHotkey** object.*

420 void **im_hotkey_append_event** (**IMHotkey** *hotkey, **IMEvent** *event)

*Append an event to a specified **IMHotkey** object.*

421 **IMEvent** * **im_hotkey_get_event** (const **IMHotkey** *hotkey, **IMSize** index)

*Get an event stored in a **IMHotkey** object at sepcified index.*

422 void **im_hotkey_foreach_event** (const **IMHotkey** *hotkey, **IMFunc** func, **IMPointer** user_data)

*Traverse all events stored in an **IMHotkey** object by calling specified function.*

423 **IMHotkeyProfile** * **im_hotkey_profile_new** (**IMUInt32** uid, **IMHotkeyProfileScope** scope)

*Create a new **IMHotkeyProfile** object.*

424 void **im_hotkey_profile_set_uid** (**IMHotkeyProfile** *profile, **IMUInt32** uid)

Set uid of a specified hotkey profile.

425 **IMUInt32 im_hotkey_profile_get_uid** (const **IMHotkeyProfile** *profile)

Get uid of a specified hotkey profile.

426 void **im_hotkey_profile_set_scope** (**IMHotkeyProfile** *profile, **IMHotkeyProfileScope** scope)

Set scope of a specified hotkey profile.

427 **IMHotkeyProfileScope im_hotkey_profile_get_scope** (const **IMHotkeyProfile** *profile)

Get scope of a specified hotkey profile.

428 void **im_hotkey_profile_clear** (**IMHotkeyProfile** *profile)

Clear all hotkeys stored in a specified hotkey profile.

429 **IMBool im_hotkey_profile_empty** (const **IMHotkeyProfile** *profile)

Check whether a hotkey profile is empty or not.

430 void **im_hotkey_profile_add** (**IMHotkeyProfile** *profile, **IMHotkey** *hotkey)

Add a hotkey into a specified hotkey profile.

431 void **im_hotkey_profile_remove** (**IMHotkeyProfile** *profile, **IMUInt32** keycode)

Remove a hotkey from a specified hotkey profile, by its key code.

432 **IMHotkey** * **im_hotkey_profile_lookup** (const **IMHotkeyProfile** *profile, **IMUInt32** keycode)

Lookup a hotkey in a hotkey profile, by specified key code.

433 void **im_hotkey_profile_foreach** (const **IMHotkeyProfile** *profile, **IMFunc** func, **IMPointer** user_data)

*Traverse all hotkeys stored in an **IMHotkeyProfile** object by calling specified function.*

Define Documentation

#define IM_CONST_HOTKEY(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_HOTKEY,IMHotkey))

#define IM_CONST_HOTKEY_PROFILE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_HOTKEY_PROFILE,IMHotkeyProfile))

#define IM_HOTKEY(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_HOTKEY,IMHotkey))

#define IM_HOTKEY_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_HOTKEY,IMHotkeyClass))

#define IM_HOTKEY_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_HOTKEY,IMHotkeyClass))

#define IM_HOTKEY_PROFILE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_HOTKEY_PROFILE,IMHotkeyProfile))

#define IM_HOTKEY_PROFILE_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_HOTKEY_PROFILE,IMHotkeyProfileClass))

#define IM_HOTKEY_PROFILE_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_HOTKEY_PROFILE,IMHotkeyProfileClass))

#define IM_IS_HOTKEY(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_HOTKEY))

#define IM_IS_HOTKEY_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_HOTKEY))

#define IM_IS_HOTKEY_PROFILE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_HOTKEY_PROFILE))

#define IM_IS_HOTKEY_PROFILE_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_HOTKEY_PROFILE))

Typedef Documentation

typedef struct _IMHotkey IMHotkey

typedef struct _IMHotkeyClass IMHotkeyClass

typedef struct _IMHotkeyProfile IMHotkeyProfile

typedef struct _IMHotkeyProfileClass IMHotkeyProfileClass

Enumeration Type Documentation

enum IMHotkeyProfileScope

Indicate the effect scope of a hotkey profile.

434 IM_HOTKEY_PROFILE_LOCAL Indicates that the hotkey profile takes effect only when the current focused IC is active and associated to corresponding IMEngine/IMLogic. Only IMEngine/IMLogic can register local hotkey profile.

435 IM_HOTKEY_PROFILE_GLOBAL Indicates that the hotkey profile takes effect all the time. Components other than IMEngines/IMLogics can only register global hotkey profiles.

Enumerator:

IM_HOTKEY_PROFILE_LOCAL
IM_HOTKEY_PROFILE_GLOBAL

Function Documentation

void im_hotkey_append_event (IMHotkey * *hotkey*, IMEvent * *event*)

Append an event to a specified **IMHotkey** object.

void im_hotkey_clear (IMHotkey * *hotkey*)

Clear all events stored in a specified **IMHotkey** object.

void im_hotkey_foreach_event (const IMHotkey * *hotkey*, IMFunc *func*, IMPointer *user_data*)

Traverse all events stored in an **IMHotkey** object by calling specified function.

IMEvent* im_hotkey_get_event (const IMHotkey * *hotkey*, IMSize *index*)

Get an event stored in a **IMHotkey** object at sepcified index.

IMUInt32 im_hotkey_get_key_code (const IMHotkey * *hotkey*)

Get key code of a specified **IMHotkey** object.

IMSize im_hotkey_get_n_events (const IMHotkey * *hotkey*)

Get how many events stored in a specified **IMHotkey** object.

IMHotkey* im_hotkey_new (IMUInt32 *keycode*)

Create a new hotkey with specified key code.

void im_hotkey_profile_add (IMHotkeyProfile * *profile*, IMHotkey * *hotkey*)

Add a hotkey into a specified hotkey profile.

If a hotkey with the same key code is already available int the profile, the old hotkey will be replaced by the new one.

void im_hotkey_profile_clear (IMHotkeyProfile * *profile*)

Clear all hotkeys stored in a specified hotkey profile.

IMBool im_hotkey_profile_empty (const IMHotkeyProfile * *profile*)

Check whether a hotkey profile is empty or not.

void im_hotkey_profile_foreach (const IMHotkeyProfile * *profile*, IMFunc *func*, IMPointer *user_data*)

Traverse all hotkeys stored in an **IMHotkeyProfile** object by calling specified function.

IMHotkeyProfileScope im_hotkey_profile_get_scope (const IMHotkeyProfile * *profile*)

Get scope of a specified hotkey profile.

IMUInt32 im_hotkey_profile_get_uid (const IMHotkeyProfile * *profile*)

Get uid of a specified hotkey profile.

IMHotkey* im_hotkey_profile_lookup (const IMHotkeyProfile * *profile*, IMUInt32 *keycode*)

Lookup a hotkey in a hotkey profile, by specified key code.

Returns:

The matched hotkey or 0 if no hotkey is matched.

IMHotkeyProfile* im_hotkey_profile_new (IMUInt32 *uid*, IMHotkeyProfileScope *scope*)

Create a new **IMHotkeyProfile** object.

Parameters:

uid Unique id of the profile to identify the profile among all profiles owned by a IMComponent component.
scope Scope of the hotkey profile.

void im_hotkey_profile_remove (IMHotkeyProfile * *profile*, IMUInt32 *keycode*)

Remove a hotkey from a specified hotkey profile, by its key code.

void im_hotkey_profile_set_scope (IMHotkeyProfile * *profile*, IMHotkeyProfileScope *scope*)

Set scope of a specified hotkey profile.

void im_hotkey_profile_set_uid (IMHotkeyProfile * *profile*, IMUInt32 *uid*)

Set uid of a specified hotkey profile.

void im_hotkey_set_key_code (IMHotkey * *hotkey*, IMUInt32 *keycode*)

Set key code of a hotkey.

IMLookupTable and IMCandidate

Data Structures

436 struct **IMCandidate**

IMCandidate *class is used to hold necessary data of a candidate phrase which can be put into an* ***IMLookupTable*** *object and displayed by GUI component.*

437 struct **IMLookupTable**

IMLookupTable *class is used to manipulate a set of* ***IMCandidate*** *objects.*

Defines

438 #define **IM_LOOKUP_TABLE_MAX_N_COLUMNS** 256

439 #define **IM_LOOKUP_TABLE_MAX_N_ROWS** 256

440 #define **IM_CANDIDATE**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_CANDIDATE,**IMCandidate**))

441 #define **IM_CONST_CANDIDATE**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_CANDIDATE,**IMCandidate**))

442 #define **IM_CANDIDATE_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_CANDIDATE,**IMCandidateClass**))

443 #define **IM_IS_CANDIDATE**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_CANDIDATE))

444 #define **IM_IS_CANDIDATE_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_CANDIDATE))

445 #define **IM_CANDIDATE_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_CANDIDATE,**IMCandidateClass**))

446 #define **IM_LOOKUP_TABLE**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_LOOKUP_TABLE,**IMLookupTable**))

447 #define **IM_CONST_LOOKUP_TABLE**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_LOOKUP_TABLE,**IMLookupTable**))

448 #define **IM_LOOKUP_TABLE_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_LOOKUP_TABLE,**IMLookupTableClass**))

449 #define **IM_IS_LOOKUP_TABLE**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_LOOKUP_TABLE))

450 #define **IM_IS_LOOKUP_TABLE_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_LOOKUP_TABLE))

451 #define **IM_LOOKUP_TABLE_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_LOOKUP_TABLE,**IMLookupTableClass**))

Typedefs

452 typedef struct **_IMCandidate IMCandidate**

453 typedef struct **_IMCandidateClass IMCandidateClass**

454 typedef struct **_IMLookupTable IMLookupTable**

455 typedef struct **_IMLookupTableClass IMLookupTableClass**

Enumerations

456 enum **IMLookupTableOrientation** { **IM_LOOKUP_TABLE_HORIZONTAL** = 0, **IM_LOOKUP_TABLE_VERTICAL** = 1 }

Layout orientation of a lookup table.

457 enum **IMLookupTableFlag** { **IM_LOOKUP_TABLE_SHOW_TITLE** = 0x01, **IM_LOOKUP_TABLE_SHOW_CURSOR** = 0x02, **IM_LOOKUP_TABLE_SHOW_SCROLL_BAR** = 0x04, **IM_LOOKUP_TABLE_SHOW_PAGE_FLIP_BUTTONS** = 0x08, **IM_LOOKUP_TABLE_HORIZONTAL_SHRINKABLE** = 0x10, **IM_LOOKUP_TABLE_VERTICAL_SHRINKABLE** = 0x20 }

Flags to control various behaviour of a lookup table.

Functions

458 **IMCandidate** * **im_candidate_new** (**IMText** *content)

*Creates a new **IMCandidate** with specified content.*

459 **IMCandidate** * **im_candidate_new_full** (**IMText** *content, **IMText** *label, **IMString** *icon, **IMPointer** annotation)

*Creates a new **IMCandidate** with specified content, label, icon and annotation.*

460 void **im_candidate_set_content** (**IMCandidate** *candidate, **IMText** *content)

*Sets the content of an **IMCandidate**.*

461 **IMText** * **im_candidate_get_content** (const **IMCandidate** *candidate)

*Gets the content of an **IMCandidate**.*

462 void **im_candidate_set_label** (**IMCandidate** *candidate, **IMText** *label)

*Sets the label of an **IMCandidate**.*

463 **IMText** * **im_candidate_get_label** (const **IMCandidate** *candidate)

*Gets the label of an **IMCandidate**.*

464 void **im_candidate_set_icon** (**IMCandidate** *candidate, **IMString** *icon)

*Sets the icon file path of an **IMCandidate**.*

465 **IMString** * **im_candidate_get_icon** (const **IMCandidate** *candidate)

*Gets the icon file path of an **IMCandidate**.*

466 void **im_candidate_set_annotation** (**IMCandidate** *candidate, **IMPointer** *annotation)

*Sets the annotation data of an **IMCandidate**.*

467 **IMPointer im_candidate_get_annotation** (const **IMCandidate** *candidate)

*Gets the annotation data of an **IMCandidate**.*

468 **IMLookupTable** * **im_lookup_table_new** (**IMUInt32** uid)

*Creates a new empty **IMLookupTable**.*

469 **IMLookupTable** * **im_lookup_table_new_full** (**IMUInt32** uid, **IMUInt** orientation, **IMUInt** flags, **IMUInt** cols, **IMUInt** rows)

*Creates a new empty **IMLookupTable**.*

470 void **im_lookup_table_clear** (**IMLookupTable** *table)

*Clears the content of an **IMLookupTable**.*

471 void **im_lookup_table_set_unique_id** (**IMLookupTable** *table, **IMUInt32** uid)

*Sets the unique ID of an **IMLookupTable**.*

472 **IMUInt32 im_lookup_table_get_unique_id** (const **IMLookupTable** *table)

*Gets the unique ID of an **IMLookupTable**.*

473 void **im_lookup_table_set_orientation** (**IMLookupTable** *table, **IMUInt** orientation)

*Sets the layout orientation of an **IMLookupTable**.*

474 **IMUInt im_lookup_table_get_orientation** (const **IMLookupTable** *table)

*Gets the layout orientation of an **IMLookupTable**.*

475 void **im_lookup_table_set_page_size** (**IMLookupTable** *table, **IMUInt** cols, **IMUInt** rows)

Sets the IMLookupTable's page size.

476 void **im_lookup_table_get_page_size** (const **IMLookupTable** *table, **IMUInt** *cols, **IMUInt** *rows)

Get lookup table's page size.

477 void **im_lookup_table_set_cursor** (**IMLookupTable** *table, **IMUInt** cur)

Set cursor position (the currently focused candidate) in current page.

478 **IMUInt im_lookup_table_get_curosr** (const **IMLookupTable** *table)

Get cursor position (the currently focused candidate).

479 void **im_lookup_table_enable_flags** (**IMLookupTable** *table, **IMUInt** flags)

Enable various flags for a lookup table to adjust its behaviour.

480 void **im_lookup_table_disable_flags** (**IMLookupTable** *table, **IMUInt** flags)

Disable various flags for a lookup table to adjust its behaviour.

481 **IMBool im_lookup_table_check_flags** (const **IMLookupTable** *table, **IMUInt** flags)

Check whether one or more flags is enabled or not.

482 void **im_lookup_table_set_title** (**IMLookupTable** *table, **IMText** *title)

Set title of a lookup table.

483 **IMText** * **im_lookup_table_get_title** (const **IMLookupTable** *table)

Get title of a lookup table.

484 void **im_lookup_table_set_scroll_info** (**IMLookupTable** *table, **IMUInt** num_pages, **IMUInt** cur_page)

Set scroll bar information of a lookup table.

485 void **im_lookup_table_get_scroll_info** (const **IMLookupTable** *table, **IMUInt** *num_pages, **IMUInt** *cur_page)

Get scroll bar information of a lookup table.

486 void **im_lookup_table_set_candidate** (**IMLookupTable** *table, **IMUInt** index, **IMCandidate** *candidate)

Set the candidate at specified index.

487 **IMCandidate** * **im_lookup_table_get_candidate** (const **IMLookupTable** *table, **IMUInt** index)

Get the candidate at specified index.

Define Documentation

#define IM_CANDIDATE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_CANDIDATE,IMCandidate))

#define IM_CANDIDATE_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_CANDIDATE,IMCandidateClass))

#define IM_CANDIDATE_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_CANDIDATE,IMCandidateClass))

#define IM_CONST_CANDIDATE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_CANDIDATE,IMCandidate))

#define IM_CONST_LOOKUP_TABLE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_LOOKUP_TABLE,IMLookupTable))

#define IM_IS_CANDIDATE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_CANDIDATE))

#define IM_IS_CANDIDATE_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_CANDIDATE))

#define IM_IS_LOOKUP_TABLE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_LOOKUP_TABLE))

#define IM_IS_LOOKUP_TABLE_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_LOOKUP_TABLE))

#define IM_LOOKUP_TABLE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_LOOKUP_TABLE,IMLookupTable))

#define IM_LOOKUP_TABLE_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_LOOKUP_TABLE,IMLookupTableClass))

#define IM_LOOKUP_TABLE_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_LOOKUP_TABLE,IMLookupTableClass))

#define IM_LOOKUP_TABLE_MAX_N_COLUMNS 256

#define IM_LOOKUP_TABLE_MAX_N_ROWS 256

Typedef Documentation

typedef struct _IMCandidate IMCandidate

typedef struct _IMCandidateClass IMCandidateClass

typedef struct _IMLookupTable IMLookupTable

typedef struct _IMLookupTableClass IMLookupTableClass

Enumeration Type Documentation

enum IMLookupTableFlag

Flags to control various behaviour of a lookup table.

488 IM_LOOKUP_TABLE_SHOW_TITLE Whether to show title text or not, default not show.
489 IM_LOOKUP_TABLE_SHOW_CURSOR Whether to show cursor or not, default show.
490 IM_LOOKUP_TABLE_SHOW_SCROLL_BAR Whether to show scroll bar or not, default not show.
491 IM_LOOKUP_TABLE_SHOW_PAGE_FLIP_BUTTONS Whether to show page flip buttons (page up and page down) or not, default show.
492 IM_LOOKUP_TABLE_HORIZONTAL_SHRINKABLE Whether the page size can be shrunk by GUI component in horizontal orientation or not, default not shrinkable. If it's shrinkable and the size is shrunk by GUI component, an event will be sent back to IMEngine to indicate it.
493 IM_LOOKUP_TABLE_VERTICAL_SHRINKABLE Whether the page size can be shrunk by GUI component in vertical orientation or not, default not shrinkable. If it's shrinkable and the size is shrunk by GUI component, an event will be sent back to IMEngine to indicate it.

Enumerator:

IM_LOOKUP_TABLE_SHOW_TITLE
IM_LOOKUP_TABLE_SHOW_CURSOR
IM_LOOKUP_TABLE_SHOW_SCROLL_BAR
IM_LOOKUP_TABLE_SHOW_PAGE_FLIP_BUTTONS
IM_LOOKUP_TABLE_HORIZONTAL_SHRINKABLE
IM_LOOKUP_TABLE_VERTICAL_SHRINKABLE

enum IMLookupTableOrientation

Layout orientation of a lookup table.

Enumerator:

IM_LOOKUP_TABLE_HORIZONTAL
IM_LOOKUP_TABLE_VERTICAL

Function Documentation

IMPointer im_candidate_get_annotation (const IMCandidate * *candidate*)

Gets the annotation data of an **IMCandidate**.

Parameters:

candidate an **IMCandidate**.

Returns:

the candidate's annotation data.

IMText* im_candidate_get_content (const IMCandidate * *candidate*)

Gets the content of an **IMCandidate**.

Parameters:

candidate an **IMCandidate**.

Returns:

the candidate's content.

IMString* im_candidate_get_icon (const IMCandidate * *candidate*)

Gets the icon file path of an **IMCandidate**.

Parameters:

candidate an **IMCandidate**.

Returns:

the candidate's icon file path.

IMText* im_candidate_get_label (const IMCandidate * *candidate*)

Gets the label of an **IMCandidate**.

Parameters:

candidate an **IMCandidate**.

Returns:

the candidate's label.

IMCandidate* im_candidate_new (IMText * *content*)

Creates a new **IMCandidate** with specified content.

Parameters:

content an **IMText** of the candidate's content.

Returns:

a new **IMCandidate**.

IMCandidate* im_candidate_new_full (IMText * *content*, IMText * *label*, IMString * *icon*, IMPointer *annotation*)

Creates a new **IMCandidate** with specified content, label, icon and annotation.

Parameters:

content an **IMText** of the candidate's content.
label an **IMText** of the candidate's label.
icon an **IMText** of the candidate's icon.
annotation an IMPointer of annotation data for the candidate.

Returns:

the new **IMCandidate**.

void im_candidate_set_annotation (IMCandidate * *candidate*, IMPointer * *annotation*)

Sets the annotation data of an **IMCandidate**.

Parameters:

candidate an **IMCandidate**.
annotation the new annotation data for candidate.

void im_candidate_set_content (IMCandidate * *candidate*, IMText * *content*)

Sets the content of an **IMCandidate**.

Parameters:

candidate an **IMCandidate**.

content the new content for candidate.

void im_candidate_set_icon (IMCandidate * *candidate*, IMString * *icon*)

Sets the icon file path of an **IMCandidate**.

Parameters:

candidate an **IMCandidate**.

icon the new icon file path for candidate.

void im_candidate_set_label (IMCandidate * *candidate*, IMText * *label*)

Sets the label of an **IMCandidate**.

Parameters:

candidate an **IMCandidate**.

label the new label for candidate.

IMBool im_lookup_table_check_flags (const IMLookupTable * *table*, IMUInt *flags*)

Check whether one or more flags is enabled or not. This fucntion return true if succeeded, otherwise return false.

void im_lookup_table_clear (IMLookupTable * *table*)

Clears the content of an **IMLookupTable**.

Only title, candidates, labels and annotations will be cleared. Other information won't be changed.

Parameters:

table an **IMLookupTable**.

void im_lookup_table_disable_flags (IMLookupTable * *table*, IMUInt *flags*)

Disable various flags for a lookup table to adjust its behaviour.

void im_lookup_table_enable_flags (IMLookupTable * *table*, IMUInt *flags*)

Enable various flags for a lookup table to adjust its behaviour.

IMCandidate* im_lookup_table_get_candidate (const IMLookupTable * *table*, IMUInt *index*)

Get the candidate at specified index.

Parameters:

table a **IMLookupTable** object

index candidate index to be get. Must be 0 to (page_size - 1). page_size = cols * rows.

Returns:

The **IMCandidate** object. Or 0 if the candidate is empty.

IMUInt im_lookup_table_get_curosr (const IMLookupTable * *table*)

Get cursor position (the currently focused candidate).

IMUInt im_lookup_table_get_orientation (const IMLookupTable * *table*)

Gets the layout orientation of an **IMLookupTable**.

Parameters:

table an **IMLookupTable**.

Returns:

the layout orientation, either IM_LOOKUP_TABLE_HORIZONTAL or IM_LOOKUP_TABLE_VERTICAL.

void im_lookup_table_get_page_size (const IMLookupTable * *table*, IMUInt * *cols*, IMUInt * *rows*)

Get lookup table's page size.

void im_lookup_table_get_scroll_info (const IMLookupTable * *table*, IMUInt * *num_pages*, IMUInt * *cur_page*)

Get scroll bar information of a lookup table.

IMText* im_lookup_table_get_title (const IMLookupTable * *table*)

Get title of a lookup table.

Returns:

The **IMText** object used by the lookup table object as title, 0 if there is no title. It shouldn't be modified or released by caller.

IMUInt32 im_lookup_table_get_unique_id (const IMLookupTable * *table*)

Gets the unique ID of an **IMLookupTable**.

Parameters:

table an **IMLookupTable**.

Returns:

the unique ID.

IMLookupTable* im_lookup_table_new (IMUInt32 *uid*)

Creates a new empty **IMLookupTable**.

Each **IMLookupTable** has an unique ID to identify itself. This allows an IMEngine to use multiple IMLookupTables at the same time. It is useful for multi-level lookup table case: candidates in a lookup table have other lookup table objects as their annotations.

Parameters:

uid a unique ID to identify this **IMLookupTable**.

Returns:

a new **IMLookupTable**.

IMLookupTable* im_lookup_table_new_full (IMUInt32 *uid*, IMUInt *orientation*, IMUInt *flags*, IMUInt *cols*, IMUInt *rows*)

Creates a new empty **IMLookupTable**.

See also:

im_lookup_table_new()

Parameters:

uid a unique ID to identify this **IMLookupTable**.
orientation the orientation of the **IMLookupTable**, should be either IM_LOOKUP_TABLE_HORIZONTAL or IM_LOOKUP_TABLE_VERTICAL.
flags FIXME
cols number of columns in the lookup table.
rows number of rows in the lookup table.

Returns:

a new **IMLookupTable**.

void im_lookup_table_set_candidate (IMLookupTable * *table*, IMUInt *index*, IMCandidate * *candidate*)

Set the candidate at specified index.

Parameters:

table an **IMLookupTable** object
index candidate index to be set. Must be 0 to (page_size - 1). page_size = cols * rows.
cand an **IMCandidate** object.

void im_lookup_table_set_cursor (IMLookupTable * *table*, IMUInt *cur*)

Set cursor position (the currently focused candidate) in current page.

void im_lookup_table_set_orientation (IMLookupTable * *table*, IMUInt *orientation*)

Sets the layout orientation of an **IMLookupTable**.

Parameters:

table an **IMLookupTable**.
orientation can be IM_LOOKUP_TABLE_HORIZONTAL or IM_LOOKUP_TABLE_VERTICAL.

void im_lookup_table_set_page_size (IMLookupTable * *table*, IMUInt *cols*, IMUInt *rows*)

Sets the IMLookupTable's page size.

A lookup table is a two dimension table, which can have multiple columns and rows. The total number of candidates can be hold by a table is number_of_columns * number_of_rows. Candidates can be indexed by linear numbers from * 0 to (ncols * nrows - 1). The layout orientation of candidates is specified by caller.

Parameters:

table an **IMLookupTable**.
cols number of columns (X size), must be greater than zero and less equal than IM_LOOKUP_TABLE_MAX_N_COLUMNS.
rows number of rows (Y size), must be greater than zero and less equal than IM_LOOKUP_TABLE_MAX_N_ROWS.

void im_lookup_table_set_scroll_info (IMLookupTable * *table*, IMUInt *num_pages*, IMUInt *cur_page*)

Set scroll bar information of a lookup table.

A lookup table can have a scroll bar to indicate which page is currently displayed among all pages. The scroll bar is also clickable by user to generate page flip event.

Parameters:

table A **IMLookupTable** object
num_pages Number of pages totally available.
cur_page Current page is being displayed, must between 0 to (num_pages - 1). If it's 0, then the scroll bar cursor will be at beginning position, and the page up button will be inactivated, so that user can't do page up. If it's num_pages - 1, then the scroll bar cursor will be at ending position, and the page down button will be inactivated, so that user can't do page done.

void im_lookup_table_set_title (IMLookupTable * *table*, IMText * *title*)

Set title of a lookup table.

The title is a **IMText** object, which can have text attributes to decorate text.

void im_lookup_table_set_unique_id (IMLookupTable * *table*, IMUInt32 *uid*)

Sets the unique ID of an **IMLookupTable**.

Parameters:

table an **IMLookupTable**.

IMObject

Data Structures

494 struct **_IMObjectClass**

495 struct **_IMObject**

496 struct **IMObjectClass**

The base struct for all classes that derived from IM_TYPE_OBJECT.

497 struct **IMObject**

The base struct for all object instances.

Defines

498 #define **IM_OBJECT**(object) (IM_TYPE_INSTANCE_CHECK_CAST((object), IM_TYPE_OBJECT, **IMObject**))

499 #define **IM_CONST_OBJECT**(object) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((object), IM_TYPE_OBJECT, **IMObject**))

500 #define **IM_OBJECT_CLASS**(klass) (IM_TYPE_CLASS_CHECK_CAST((klass), IM_TYPE_OBJECT, **IMObjectClass**))

501 #define **IM_IS_OBJECT**(object) (IM_TYPE_INSTANCE_CHECK_TYPE((object),IM_TYPE_OBJECT))

502 #define **IM_IS_OBJECT_CLASS**(klass) (IM_TYPE_CLASS_CHECK_TYPE((klass),IM_TYPE_OBJECT))

503 #define **IM_OBJECT_GET_CLASS**(object) (IM_TYPE_INSTANCE_GET_CLASS_CAST((object),IM_TYPE_OBJECT,**IMObjectClass**))

504 #define **IM_TYPE_IS_OBJECT**(type) (im_type_is_a((type),IM_TYPE_OBJECT))

Typedefs

505 typedef struct **_IMObject IMObject**

506 typedef struct **_IMObjectClass IMObjectClass**

507 typedef struct **_IMObjectImpl IMObjectImpl**

Functions

508 const **IMObjectClass** * **im_object_class_get_parent** (**IMConstPointer** klass)

*Get parent class pointer of a class which is derived from **IMObjectClass**.*

509 **IMPointer im_object_new** (**IMType** type)

Create a new object instance for a given type.

510 **IMPointer im_object_ref** (**IMPointer** object)

Increate reference count of an object by one.

511 void **im_object_unref** (**IMPointer** object)

Decrease reference count of an object by one.

512 **IMPointer im_object_copy** (**IMPointer** dest, **IMConstPointer** src)

Copy the content of src object to dest object.

513 **IMPointer im_object_clone** (**IMConstPointer** object)

Clone an object.

514 **IMSize im_object_sizeof** (**IMConstPointer** object)

Return the size of an object.

515 **IMBool im_object_valid** (**IMConstPointer** object)

Check whether an object is valid or not.

516 **IMType im_object_get_type** (**IMConstPointer** object)

Get the type of an object.

517 const **IMObjectClass** * **im_object_get_class** (**IMConstPointer** object)

Get class struct pointer of an object.

518 **IMBool im_object_is_a** (**IMConstPointer** object, **IMType** is_a_type)

*Same as **im_type_instance_is_a()**.*

519 **IMBool im_object_serialize** (**IMConstPointer** object, **IMPointer** stream)

*Save an object into an **IMStream** object.*

520 **IMBool im_object_deserialize** (**IMPointer** object, **IMPointer** stream)

*Load an object from an **IMStream** object.*

521 **IMBool im_object_put_to_stream** (**IMConstPointer** object, **IMPointer** stream)

*Save an object into an **IMStream** object.*

522 **IMBool im_object_get_from_stream** (**IMPointer** object, **IMPointer** stream)

*Load an object from an **IMStream** object.*

523 void **im_object_attach_child** (**IMPointer** object, const **IMChar** *key, **IMPointer** child)

Attach an object to another object with specified string key.

524 **IMPointer im_object_get_child** (**IMConstPointer** object, const **IMChar** *key)

Return the child object with specified key of an object.

525 void **im_object_remove_child** (**IMPointer** object, const **IMChar** *key)

Remove a child object with specified key of an object.

526 void **im_object_attach_quark_child** (**IMPointer** object, **IMQuark** quark, **IMPointer** child)

Attach an object to another object with specified quark key.

527 **IMPointer im_object_get_quark_child** (**IMConstPointer** object, **IMQuark** quark)

Return the child object with specified quark key of an object.

528 void **im_object_remove_quark_child** (**IMPointer** object, **IMQuark** quark)

Remove a child object with specified quark key of an object.

Define Documentation

#define IM_CONST_OBJECT(object) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((object), IM_TYPE_OBJECT, IMObject))

#define IM_IS_OBJECT(object) (IM_TYPE_INSTANCE_CHECK_TYPE((object),IM_TYPE_OBJECT))

#define IM_IS_OBJECT_CLASS(klass) (IM_TYPE_CLASS_CHECK_TYPE((klass),IM_TYPE_OBJECT))

#define IM_OBJECT(object) (IM_TYPE_INSTANCE_CHECK_CAST((object), IM_TYPE_OBJECT, IMObject))

#define IM_OBJECT_CLASS(klass) (IM_TYPE_CLASS_CHECK_CAST((klass), IM_TYPE_OBJECT, IMObjectClass))

#define IM_OBJECT_GET_CLASS(object) (IM_TYPE_INSTANCE_GET_CLASS_CAST((object),IM_TYPE_OBJECT,IMObjectClass))

#define IM_TYPE_IS_OBJECT(type) (im_type_is_a((type),IM_TYPE_OBJECT))

Typedef Documentation

typedef struct _IMObject IMObject

typedef struct _IMObjectClass IMObjectClass

typedef struct _IMObjectImpl IMObjectImpl

Function Documentation

void im_object_attach_child (IMPointer *object*, const IMChar * *key*, IMPointer *child*)

Attach an object to another object with specified string key.

An object can have one or more child objects attached. Each child object is associated with a specified key, so that it can be retrieved by the key later.

Parameters:

object The parent object.
key The string key to be associated to the child object.
child The child object. It'll just be referenced by the parent object, rather than making a copy.

void im_object_attach_quark_child (IMPointer *object*, IMQuark *quark*, IMPointer *child*)

Attach an object to another object with specified quark key.

Just like **im_object_attach_child()**, but use a IMQuark instead of a string key.

const IMObjectClass* im_object_class_get_parent (IMConstPointer *klass*)

Get parent class pointer of a class which is derived from **IMObjectClass**.

IMPointer im_object_clone (IMConstPointer *object*)

Clone an object.

It creates a new object and copy the content from source object by calling im_object_copy ().

Parameters:

object The source object to be cloned.

Returns:

New cloned object.

IMPointer im_object_copy (IMPointer *dest*, IMConstPointer *src*)

Copy the content of src object to dest object.

The copy function is a virtual function. Each class should implement its own copy function with correct behaviour.

This function will only call copy method of dest class. Copy method of its parent classes should be called inside copy method of dest class.

The type of src and dest must be same or at least src is a derived type of dest. Otherwise the copy will fail.

Parameters:

dest Pointer to dest object.
src Pointer to source object.

Returns:

dest on success, 0 on fail.

IMBool im_object_deserialize (IMPointer *object*, IMPointer *stream*)

Load an object from an **IMStream** object.

This function will only call deserialize method of the object's class.

deserialize method of its parent class should be called inside its own deserialize method, so that data member of its parent class can be deserialized correctly.

This function couldn't handle type information of the object written by **im_stream_put_object()** function. Use **im_stream_get_object()** instead to load an object which is written by **im_stream_put_object()**.

IMPointer im_object_get_child (IMConstPointer *object*, const IMChar * *key*)

Return the child object with specified key of an object.

Parameters:

object The parent object.
key Key of the child object to be retrieved.

Returns:

The child object, or 0 if there is no child object for that key.

const IMObjectClass* im_object_get_class (IMConstPointer *object*)

Get class struct pointer of an object.

IMBool im_object_get_from_stream (IMPointer *object*, IMPointer *stream*)

Load an object from an **IMStream** object.

This function calls im_stream_get_object(stream, object) to load an object from a stream.

The type of object must be same as the object type in the stream's current read position.

The old content of given object will be destroyed.

See also:

im_stream_get_object();

Returns:

TRUE if an object is loaded successfully.

IMPointer im_object_get_quark_child (IMConstPointer *object*, IMQuark *quark*)

Return the child object with specified quark key of an object.

Just like **im_object_get_child()**, but use a IMQuark instead of a string key.

IMType im_object_get_type (IMConstPointer *object*)

Get the type of an object.

It just calls **im_type_instance_get_type()**.

IMBool im_object_is_a (IMConstPointer *object*, IMType *is_a_type*)

Same as **im_type_instance_is_a()**.

IMPointer im_object_new (IMType *type*)

Create a new object instance for a given type.

This function will allocate required memory and call constructors of the given class type and its parent class types in correct sequence to initialize the object.

A newly created object doesn't have reference count, it's called floating object. It can be destroyed by calling **im_object_unref()**.

Parameters:

type Type of the class.

Returns:

A pointer to the newly created object.

IMBool im_object_put_to_stream (IMConstPointer *object*, IMPointer *stream*)

Save an object into an **IMStream** object.

This function calls im_stream_put_object(stream, object) to save an object into a stream.

See also:

im_stream_put_object();

IMPointer im_object_ref (IMPointer *object*)

Increate reference count of an object by one.

For a newly created object, the reference count will be set to 1. Means that it now has one owner.

Parameters:

object Pointer to the object.

Returns:

The same pointer if success, 0 if failed.

void im_object_remove_child (IMPointer *object*, const IMChar * *key*)

Remove a child object with specified key of an object.

Parameters:

object The parent object.
key Key of the child object to be removed.

void im_object_remove_quark_child (IMPointer *object*, IMQuark *quark*)

Remove a child object with specified quark key of an object.

Just like **im_object_remove_child()**, but use a IMQuark instead.

IMBool im_object_serialize (IMConstPointer *object*, IMPointer *stream*)

Save an object into an **IMStream** object.

This function will only call serialize method of the object's class.

serialize method of its parent class should be called inside its own serialize method, so that data member of its parent class can be serialized correctly.

This function will not write type information of the object into the stream. Use **im_stream_put_object()** instead to write type information as well as object data into the stream.

IMSize im_object_sizeof (IMConstPointer *object*)

Return the size of an object.

void im_object_unref (IMPointer *object*)

Decrease reference count of an object by one.

For a newly created object, it'll be destroyed directly. For object which has reference count, it'll only be destroyed when reference count is reduced to zero.

Parameters:

object Pointer to the object.

IMBool im_object_valid (IMConstPointer *object*)

Check whether an object is valid or not. This fucntion return true if succeeded, otherwise return false.

IMProperty

Data Structures

1 struct **IMProperty**

An object holds a property of an IMComponent.

2 struct **IMPropList**

An object to hold multiple properties.

3 struct **IMPropIterator**

*Iterator of **IMPropList** to iterate all properties stored in a **IMPropList** object.*

Defines

4 #define **IM_PROPERTY**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_PROPERTY,**IMProperty**))

5 #define **IM_CONST_PROPERTY**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_PROPERTY,**IMProperty**))

6 #define **IM_PROPERTY_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_PROPERTY,**IMPropertyClass**))

7 #define **IM_IS_PROPERTY**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_PROPERTY))

8 #define **IM_IS_PROPERTY_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_PROPERTY))

9 #define **IM_PROPERTY_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_PROPERTY,**IMPropertyClass**))

10 #define **IM_PROP_LIST**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_PROP_LIST,**IMPropList**))

11 #define **IM_CONST_PROP_LIST**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_PROP_LIST,**IMPropList**))

12 #define **IM_PROP_LIST_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_PROP_LIST,**IMPropListClass**))

13 #define **IM_IS_PROP_LIST**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_PROP_LIST))

14 #define **IM_IS_PROP_LIST_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_PROP_LIST))

15 #define **IM_PROP_LIST_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_PROP_LIST,**IMPropListClass**))

Typedefs

16 typedef struct **_IMProperty IMProperty**

17 typedef struct **_IMPropertyClass IMPropertyClass**

18 typedef struct **_IMPropList IMPropList**

19 typedef struct **_IMPropListClass IMPropListClass**

20 typedef struct **_IMPropIterator IMPropIterator**

Enumerations

21 enum **IMPropertyType** { **IM_PROPERTY_NORMAL** = 0, **IM_PROPERTY_SWITCH**, **IM_PROPERTY_RADIO**, **IM_PROPERTY_SEPARATOR** }

22 enum **IMPropertyFlag** { **IM_PROPERTY_SENSITIVE** = 0x01, **IM_PROPERTY_VISIBLE** = 0x02, **IM_PROPERTY_CHECKED** = 0x04, **IM_PROPERTY_UNCHECKED** = 0x08, **IM_PROPERTY_INCONSISTENT** = IM_PROPERTY_CHECKED| IM_PROPERTY_UNCHECKED }

Functions

23 **IMProperty** * **im_property_new** (**IMString** *key)

Create a new property with a specified key string.

24 **IMProperty** * **im_property_new_full** (**IMPropertyType** type, **IMUInt** flags, **IMString** *key, **IMString** *icon, **IMText** *label, **IMText** *tip)

Create a new property with specified parameters.

25 void **im_property_set_type** (**IMProperty** *prop, **IMPropertyType** type)

Set the type of specified property.

26 **IMPropertyType im_property_get_type** (const **IMProperty** *prop)

Get the type of specified property.

27 void **im_property_enable_flags** (**IMProperty** *prop, **IMUInt** flags)

Enable various flags for a property to adjust its behaviour.

28 void **im_property_disable_flags** (**IMProperty** *prop, **IMUInt** flags)

Disable various flags for a property to adjust its behaviour.

29 **IMBool im_property_check_flags** (const **IMProperty** *prop, **IMUInt** flags)

Check whether one or more flags is enabled or not.

30 void **im_property_set_key** (**IMProperty** *prop, **IMString** *key)

Set key string of a property.

31 **IMString** * **im_property_get_key** (const **IMProperty** *prop)

*Get key string of a specified **IMProperty** object.*

32 void **im_property_set_icon** (**IMProperty** *prop, **IMString** *icon)

Set icon file of a property.

33 **IMString** * **im_property_get_icon** (const **IMProperty** *prop)

*Get icon file path of a specified **IMProperty** object.*

34 void **im_property_set_label** (**IMProperty** *prop, **IMText** *label)

*Set the label of a specofied **IMProperty** object.*

35 **IMText** * **im_property_get_label** (const **IMProperty** *prop)

*Get tje label of a specified **IMProperty** object.*

36 void **im_property_set_tip** (**IMProperty** *prop, **IMText** *tip)

*Set the tip of a specofied **IMProperty** object.*

37 **IMText** * **im_property_get_tip** (const **IMProperty** *prop)

*Get the tip of a specified **IMProperty** object.*

38 void **im_property_set_sub_props** (**IMProperty** *prop, **IMPropList** *subprops)

Attach or remove a list of sub properties to a specified property.

39 **IMPropList** * **im_property_get_sub_props** (const **IMProperty** *prop)

Get sub properties list attached to a specified property.

40 **IMPropList** * **im_prop_list_new** ()

*Create a new **IMPropList** object.*

41 void **im_prop_list_clear** (**IMPropList** *list)

*Clear an **IMPropList** object. All **IMProperty** objects held by it will be released.*

42 **IMBool im_prop_list_empty** (const **IMPropList** *list)

*Check whether a specified **IMPropList** object is empty or not.*

43 void **im_prop_list_append** (**IMPropList** *list, **IMProperty** *prop)

*Append an **IMProperty** object into a specified **IMPropList** object.*

44 void **im_prop_list_remove** (**IMPropList** *list, const **IMChar** *key)

*Remove a **IMProperty** object with specified key from a specified **IMPropList** object.*

45 **IMProperty** * **im_prop_list_get** (const **IMPropList** *list, const **IMChar** *key)

*Get a **IMProperty** object with specified key stored in a specified **IMPropList** object.*

46 void **im_prop_list_foreach** (const **IMPropList** *list, **IMFunc** func, **IMPointer** user_data)

*Call a specified function against each **IMProperty** objects stored in a specified **IMPropList** object.*

47 **IMPropIterator** * **im_prop_list_get_iterator** (**IMPropList** *list)

*Create an iterator for a specified **IMPropList** object.*

48 **IMPropIterator** * **im_prop_iterator_clone** (const **IMPropIterator** *iter)

*Clone a **IMPropIterator** structure.*

49 void **im_prop_iterator_destroy** (**IMPropIterator** *iter)

*Destroy a **IMPropIterator** instance.*

50 **IMBool im_prop_iterator_next** (**IMPropIterator** *iter)

*Advance a **IMPropIterator** instance to point to the next **IMProperty** object inside its associated **IMPropList** object.*

51 **IMProperty** * **im_prop_iterator_get** (const **IMPropIterator** *iter)

*Get the **IMProperty** object pointed by a specified **IMPropIterator** instance.*

52 **IMBool im_prop_iterator_valid** (const **IMPropIterator** *iter)

*Check whether a **IMPropIterator** instance is pointed to a valid **IMProperty** object.*

53 **IMBool im_prop_iterator_equal** (const **IMPropIterator** *iter1, const **IMPropIterator** *iter2)

*Check whether two **IMPropIterator** instances point to the same **IMProperty** object or not.*

54 void **im_prop_iterator_remove** (**IMPropIterator** *iter)

*Remove the **IMProperty** object pointed by specified **IMPropIterator** instance from its associated **IMPropList** object,.*

Define Documentation

#define IM_CONST_PROP_LIST(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_PROP_LIST,IMPropList))

#define IM_CONST_PROPERTY(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_PROPERTY,IMProperty))

#define IM_IS_PROP_LIST(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_PROP_LIST))

#define IM_IS_PROP_LIST_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_PROP_LIST))

#define IM_IS_PROPERTY(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_PROPERTY))

#define IM_IS_PROPERTY_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_PROPERTY))

#define IM_PROP_LIST(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_PROP_LIST,IMPropList))

#define IM_PROP_LIST_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_PROP_LIST,IMPropListClass))

#define IM_PROP_LIST_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_PROP_LIST,IMPropListClass))

#define IM_PROPERTY(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_PROPERTY,IMProperty))

#define IM_PROPERTY_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_PROPERTY,IMPropertyClass))

#define IM_PROPERTY_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_PROPERTY,IMPropertyClass))

Typedef Documentation

typedef struct _IMProperty IMProperty

typedef struct _IMPropertyClass IMPropertyClass

typedef struct _IMPropIterator IMPropIterator

typedef struct _IMPropList IMPropList

typedef struct _IMPropListClass IMPropListClass

Enumeration Type Documentation

enum IMPropertyFlag

Enumerator:

IM_PROPERTY_SENSITIVE
IM_PROPERTY_VISIBLE
IM_PROPERTY_CHECKED
IM_PROPERTY_UNCHECKED
IM_PROPERTY_INCONSISTENT

enum IMPropertyType

Enumerator:

IM_PROPERTY_NORMAL
IM_PROPERTY_SWITCH
IM_PROPERTY_RADIO
IM_PROPERTY_SEPARATOR

Function Documentation

IMPropIterator* im_prop_iterator_clone (const IMPropIterator * *iter*)

Clone a **IMPropIterator** structure.

void im_prop_iterator_destroy (IMPropIterator * *iter*)

Destroy a **IMPropIterator** instance.

IMBool im_prop_iterator_equal (const IMPropIterator * *iter1*, const IMPropIterator * *iter2*)

Check whether two **IMPropIterator** instances point to the same **IMProperty** object or not. This fucntion return true if succeeded, otherwise return false.

IMProperty* im_prop_iterator_get (const IMPropIterator * *iter*)

Get the **IMProperty** object pointed by a specified **IMPropIterator** instance.

IMBool im_prop_iterator_next (IMPropIterator * *iter*)

Advance a **IMPropIterator** instance to point to the next **IMProperty** object inside its associated **IMPropList** object. This fucntion return true if succeeded, otherwise return false.

void im_prop_iterator_remove (IMPropIterator * *iter*)

Remove the **IMProperty** object pointed by specified **IMPropIterator** instance from its associated **IMPropList** object,.

IMBool im_prop_iterator_valid (const IMPropIterator * *iter*)

Check whether a **IMPropIterator** instance is pointed to a valid **IMProperty** object. This fucntion return true if succeeded, otherwise return false.

void im_prop_list_append (IMPropList * *list*, IMProperty * *prop*)

Append an **IMProperty** object into a specified **IMPropList** object.

void im_prop_list_clear (IMPropList * *list*)

Clear an **IMPropList** object. All **IMProperty** objects held by it will be released.

IMBool im_prop_list_empty (const IMPropList * *list*)

Check whether a specified **IMPropList** object is empty or not. This fucntion return true if succeeded, otherwise return false.

void im_prop_list_foreach (const IMPropList * *list*, IMFunc *func*, IMPointer *user_data*)

Call a specified function against each **IMProperty** objects stored in a specified **IMPropList** object.

IMProperty* im_prop_list_get (const IMPropList * *list*, const IMChar * *key*)

Get a **IMProperty** object with specified key stored in a specified **IMPropList** object.

IMPropIterator* im_prop_list_get_iterator (IMPropList * *list*)

Create an iterator for a specified **IMPropList** object.

IMPropList* im_prop_list_new ()

Create a new **IMPropList** object.

void im_prop_list_remove (IMPropList * *list*, const IMChar * *key*)

Remove a **IMProperty** object with specified key from a specified **IMPropList** object.

IMBool im_property_check_flags (const IMProperty * *prop*, IMUInt *flags*)

Check whether one or more flags is enabled or not. This fucntion return true if succeeded, otherwise return false.

void im_property_disable_flags (IMProperty * *prop*, IMUInt *flags*)

Disable various flags for a property to adjust its behaviour.

void im_property_enable_flags (IMProperty * *prop*, IMUInt *flags*)

Enable various flags for a property to adjust its behaviour.

IMString* im_property_get_icon (const IMProperty * *prop*)

Get icon file path of a specified **IMProperty** object.

IMString* im_property_get_key (const IMProperty * *prop*)

Get key string of a specified **IMProperty** object.

IMText* im_property_get_label (const IMProperty * *prop*)

Get tje label of a specified **IMProperty** object.

IMPropList* im_property_get_sub_props (const IMProperty * *prop*)

Get sub properties list attached to a specified property.

Returns:

The **IMPropList** object which contains sub properties, or 0 if there is no sub properties. Caller shouldn't modify or release the returned object.

IMText* im_property_get_tip (const IMProperty * *prop*)

Get the tip of a specified **IMProperty** object.

IMPropertyType im_property_get_type (const IMProperty * *prop*)

Get the type of specified property.

IMProperty* im_property_new (IMString * *key*)

Create a new property with a specified key string.

The default type is IM_PROPERTY_NORMAL, default flag is IM_PROPERTY_SENSITIVE| IM_PROPERTY_VISIBLE

IMProperty* im_property_new_full (IMPropertyType *type*, IMUInt *flags*, IMString * *key*, IMString * *icon*, IMText * *label*, IMText * *tip*)

Create a new property with specified parameters.

void im_property_set_icon (IMProperty * *prop*, IMString * *icon*)

Set icon file of a property.

void im_property_set_key (IMProperty * *prop*, IMString * *key*)

Set key string of a property.

void im_property_set_label (IMProperty * *prop*, IMText * *label*)

Set the label of a specofied **IMProperty** object.

The label **IMText** object will only be referenced rather than cloning.

void im_property_set_sub_props (IMProperty * *prop*, IMPropList * *subprops*)

Attach or remove a list of sub properties to a specified property.

Parameters:

prop A specified property.
subprops A **IMPropList** object contains one or more sub properties, or 0 to remove the old list.

void im_property_set_tip (IMProperty * *prop*, IMText * *tip*)

Set the tip of a specofied **IMProperty** object.

The tip **IMText** object will only be referenced rather than cloning.

void im_property_set_type (IMProperty * *prop*, IMPropertyType *type*)

Set the type of specified property.

If the type is set to IM_PROPERTY_SEPARATOR, all original information, except the string key, will be lost.

IMString

Data Structures

55 struct **IMString**

***IMString** class is used to manipulate a piece of ascii or UTF-8 encoded string.*

Defines

56 #define **IM_STRING**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_STRING,**IMString**))

57 #define **IM_CONST_STRING**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_STRING,**IMString**))

58 #define **IM_STRING_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_STRING,**IMStringClass**))

59 #define **IM_IS_STRING**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_STRING))

60 #define **IM_IS_STRING_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_STRING))

61 #define **IM_STRING_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_STRING,**IMStringClass**))

62 #define **IM_STRING_NPOS** (**IMSize**)(-1)

Typedefs

63 typedef struct **_IMString IMString**

64 typedef struct **_IMStringClass IMStringClass**

Functions

65 **IMString** * **im_string_new** (const **IMChar** *cstr)

*Create a new **IMString** object with specified string as initial content.*

66 **IMString** * **im_string_new_length** (const **IMChar** *cstr, **IMSize** length)

*Create a new **IMString** object with specified string and length as initial content.*

67 **IMString** * **im_string_new_static** (const **IMChar** *cstr)

*Create a new **IMString** object with specified static string as initial content.*

68 **IMString** * **im_string_new_reserved** (**IMSize** capacity)

*Create a new **IMString** object with enough memory for specified number of characters to prevent reallocation when inserting many characters.*

69 **IMString** * **im_string_new_from_printf** (const **IMChar** *format,...) IM_GNUC_PRINTF(1

*Create a new **IMString** object from arbitrary printf result.*

70 **IMString** ***IMString** * **im_string_new_from_text_utf8** (**IMConstPointer** text)

*Create a new **IMString** object from a specified **IMText** object.*

71 void **im_string_clear** (**IMString** *string)

*Erases the content of a **IMString** object, making it empty.*

72 void **im_string_reserve** (**IMString** *string, **IMSize** size)

Preallocate enough memory for specified number of characters to prevent reallocation when inserting many characters.

73 **IMSize im_string_capacity** (const **IMString** *string)

*Returns the total number of characters that the **IMString** can hold before needing to allocate more memory.*

74 **IMBool im_string_empty** (const **IMString** *string)

*Returns true if the **IMString** object is empty.*

75 void **im_string_resize** (**IMString** *string, **IMSize** new_size, **IMChar** fill_ch)

*Resizes the **IMString** object to the specified number of characters.*

76 **IMSize im_string_length** (const **IMString** *string)

*Return length of an **IMString** object, in number of characters.*

77 **IMUInt im_string_hash** (const **IMString** *string)

*Hash function of **IMString**. Used with **IMHashTable**.*

78 const **IMChar** * **im_string_c_str** (const **IMString** *string)

*Return the real content of a **IMString** object.*

79 **IMString** * **im_string_assign** (**IMString** *string, const **IMChar** *cstr, **IMSize** len)

*Copy the content of specified C string to an **IMString** object.*

80 **IMString** * **im_string_assign_static** (**IMString** *string, const **IMChar** *cstr)

*Set a static C string as the content of specified **IMString** object.*

81 **IMString** * **im_string_assign_text_utf8** (**IMString** *string, **IMConstPointer** text)

*Copy the content of specified **IMText** object to an **IMString** object.*

82 **IMChar im_string_get_char** (const **IMString** *string, **IMSize** idx)

*Get the character at specified index in an **IMString** object.*

83 void **im_string_set_char** (**IMString** *string, **IMSize** idx, **IMChar** ch)

*Set the character at specified index in an **IMString** object to a given character.*

84 **IMString** * **im_string_get_sub_string** (const **IMString** *string, **IMSize** pos, **IMSize** len)

*Create a new **IMString** object containing a sub portion of a given **IMString** object.*

85 **IMString** * **im_string_prepend** (**IMString** *string, const **IMString** *other)

*Prepend an **IMString** object into the beginning of a **IMString** object.*

86 **IMString** * **im_string_prepend_str** (**IMString** *string, const **IMChar** *other, **IMSize** len)

*Prepend a C string into the beginning of a **IMString** object.*

87 **IMString** * **im_string_prepend_char** (**IMString** *string, **IMChar** ch)

*Prepend an unicode character into the beginning of a **IMString** object.*

88 **IMString** * **im_string_append** (**IMString** *string, const **IMString** *other)

*Append an **IMString** object into the end of a **IMString** object.*

89 **IMString** * **im_string_append_str** (**IMString** *string, const **IMChar** *other, **IMSize** len)

*Append a C string into the beginning of a **IMString** object.*

90 **IMString** * **im_string_append_char** (**IMString** *string, **IMChar** ch)

*Append an unicode character into the end of a **IMString** object.*

91 **IMString** * **im_string_insert** (**IMString** *string, **IMSize** pos, const **IMString** *other)

*Insert an **IMString** object into an **IMString** object at the specified position.*

92 **IMString** * **im_string_insert_str** (**IMString** *string, **IMSize** pos, const **IMChar** *other, **IMSize** len)

*Insert a C string into the beginning of a **IMString** object.*

93 **IMString** * **im_string_insert_char** (**IMString** *string, **IMSize** pos, **IMChar** ch)

*Insert an unicode character into an **IMString** object at the specified position.*

94 **IMString** * **im_string_erase** (**IMString** *string, **IMSize** pos, **IMSize** len)

*Erase a specified portion of an **IMString** object.*

95 **IMInt im_string_compare** (const **IMString** *string1, const **IMString** *string2)

*Compare two **IMString** objects.*

96 **IMInt im_string_compare_str** (const **IMString** *string1, const **IMChar** *string2)

*Compare an **IMString** object with a C string.*

97 **IMInt im_string_n_compare** (const **IMString** *string1, const **IMString** *string2, **IMSize** len)

*Compare two **IMString** objects for a specified length.*

98 **IMInt im_string_n_compare_str** (const **IMString** *string1, const **IMChar** *string2, **IMSize** len)

*Compare an **IMString** object with a C string for a specified length.*

99 **IMInt im_string_ascii_case_compare** (const **IMString** *string1, const **IMString** *string2)

*Compare two **IMString** objects ignoring the case of ASCII characters.*

100 **IMInt im_string_ascii_case_compare_str** (const **IMString** *string1, const **IMChar** *string2)

*Compare an **IMString** object with a C string ignoring the case of ASCII characters.*

101 **IMInt im_string_ascii_n_case_compare** (const **IMString** *string1, const **IMString** *string2, **IMSize** len)

*Compare two **IMString** objects for a specified length ignoring the case of ASCII characters.*

102 **IMInt im_string_ascii_n_case_compare_str** (const **IMString** *string1, const **IMChar** *string2, **IMSize** len)

*Compare an **IMString** object with a C string for a specified length ignoring the case of ASCII characters.*

103 **IMString** * **im_string_prepend_printf** (**IMString** *string, const **IMChar** *format,...) IM_GNUC_PRINTF(2

Prepend arbitrary UTF-8 encoded printf result to a specified string.

104 **IMString** ***IMString** * **im_string_append_printf** (**IMString** *string, const **IMChar** *format,...) IM_GNUC_PRINTF(2

Append arbitrary UTF-8 encoded printf result to a specified string.

105 **IMString** ***IMString** ***IMString** * **im_string_insert_printf** (**IMString** *string, **IMSize** pos, const **IMChar** *format,...) IM_GNUC_PRINTF(3

Insert arbitrary UTF-8 encoded printf result to a specified string at specified position.

106 **IMString** ***IMString** ***IMString** ***IMString** * **im_string_printf** (**IMString** *string, const **IMChar** *format,...) IM_GNUC_PRINTF(2

*Print arbitrary string into an **IMString** object.*

107 **IMString** ***IMString** ***IMString** ***IMString** ***IMSize im_string_find** (const **IMString** *string, const **IMString** *other, **IMSize** pos)

Find position of a string.

108 **IMSize im_string_find_str** (const **IMString** *string, const **IMChar** *other, **IMSize** pos, **IMSize** len)

Find position of a string.

109 **IMSize im_string_find_char** (const **IMString** *string, **IMChar** ch, **IMSize** pos)

Find position of a string.

110 **IMSize im_string_rfind** (const **IMString** *string, const **IMString** *other, **IMSize** pos)

Find last position of a string.

111 **IMSize im_string_rfind_str** (const **IMString** *string, const **IMChar** *other, **IMSize** pos, **IMSize** len)

Find last position of a string.

112 **IMSize im_string_rfind_char** (const **IMString** *string, **IMChar** ch, **IMSize** pos)

Find last position of a string.

113 **IMSize im_string_find_first_of** (const **IMString** *string, const **IMString** *other, **IMSize** pos)

Find position of a character of string.

114 **IMSize im_string_find_first_of_str** (const **IMString** *string, const **IMChar** *other, **IMSize** pos, **IMSize** len)

Find position of a character of string.

115 **IMSize im_string_find_last_of** (const **IMString** *string, const **IMString** *other, **IMSize** pos)

Find last position of a character of string.

116 **IMSize im_string_find_last_of_str** (const **IMString** *string, const **IMChar** *other, **IMSize** pos, **IMSize** len)

Find last position of a character of string.

117 void **im_string_ascii_tolower** (**IMString** *string)

Convert all alphabet characters in a specified string to lowercase.

118 void **im_string_ascii_toupper** (**IMString** *string)

Convert all alphabet characters in a specified string to uppercase.

119 void **im_string_ascii_tolower_range** (**IMString** *string, **IMSize** pos, **IMSize** len)

Convert all alphabet characters in a specified range of a string to lowercase.

120 void **im_string_ascii_toupper_range** (**IMString** *string, **IMSize** pos, **IMSize** len)

Convert all alphabet characters in a specified range of a string to uppercase.

121 void **im_string_trim_blanks** (**IMString** *string)

Remove blank characters at the begin and end of a string.

122 **IMObjectArray** * **im_string_split** (const **IMString** *string, const **IMChar** *delim_chars)

Split the specified string into sub strings by one of given delimiter chars.

123 **IMString** * **im_string_get_left_portion** (const **IMString** *string, const **IMChar** *delim_chars)

Get the left portion of a specified string, which is splitted by one of given delimiters. If there are more than one delimiters, then the string will be splitted by the left most delimiter. White space around the sub string will be trimmed.

124 **IMString** * **im_string_get_right_portion** (const **IMString** *string, const **IMChar** *delim_chars)

Get the right portion of a specified string, which is splitted by one of given delimiters. If there are more than one delimiters, then the string will be splitted by the left most delimiter. White space around the sub string will be trimmed.

Define Documentation

#define IM_CONST_STRING(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_STRING,IMString))

#define IM_IS_STRING(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_STRING))

#define IM_IS_STRING_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_STRING))

#define IM_STRING(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_STRING,IMString))

#define IM_STRING_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_STRING,IMStringClass))

#define IM_STRING_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_STRING,IMStringClass))

#define IM_STRING_NPOS (IMSize)(-1)

Typedef Documentation

typedef struct _IMString IMString

typedef struct _IMStringClass IMStringClass

Function Documentation

IMString* im_string_append (IMString * *string*, const IMString * *other*)

Append an **IMString** object into the end of a **IMString** object.

Parameters:

string an **IMString** object.
other an **IMString** object which will be appended into string.

Returns:

string.

IMString* im_string_append_char (IMString * *string*, IMChar *ch*)

Append an unicode character into the end of a **IMString** object.

Parameters:

string an **IMString** object.
ch a character which will be appended into string.

Returns:

string.

IMString* IMString* im_string_append_printf (IMString * *string*, const IMChar * *format*, ...)

Append arbitrary UTF-8 encoded printf result to a specified string.

Parameters:

string a **IMString** object.
format format string.
... arbitrary variables to be printed according to format string.

Returns:

string

IMString* im_string_append_str (IMString * *string*, const IMChar * *other*, IMSize *len*)

Append a C string into the beginning of a **IMString** object.

Parameters:

string an **IMString** object.
other a string which will be appended into string.
len length of other, -1 means until end of the string.

Returns:

string.

IMInt im_string_ascii_case_compare (const IMString * *string1*, const IMString * *string2*)

Compare two **IMString** objects ignoring the case of ASCII characters.

Returns:

-1 means string1 < string2, 0 means they are equal, 1 means string1 > string2.

IMInt im_string_ascii_case_compare_str (const IMString * *string1*, const IMChar * *string2*)

Compare an **IMString** object with a C string ignoring the case of ASCII characters.

Returns:

-1 means string1 < string2, 0 means they are equal, 1 means string1 > string2.

IMInt im_string_ascii_n_case_compare (const IMString * *string1*, const IMString * *string2*, IMSize *len*)

Compare two **IMString** objects for a specified length ignoring the case of ASCII characters.

Only a specified length of string content will be compared.

IMInt im_string_ascii_n_case_compare_str (const IMString * *string1*, const IMChar * *string2*, IMSize *len*)

Compare an **IMString** object with a C string for a specified length ignoring the case of ASCII characters.

Only a specified length of string content will be compared.

void im_string_ascii_tolower (IMString * *string*)

Convert all alphabet characters in a specified string to lowercase.

void im_string_ascii_tolower_range (IMString * *string*, IMSize *pos*, IMSize *len*)

Convert all alphabet characters in a specified range of a string to lowercase.

void im_string_ascii_toupper (IMString * *string*)

Convert all alphabet characters in a specified string to uppercase.

void im_string_ascii_toupper_range (IMString * *string*, IMSize *pos*, IMSize *len*)

Convert all alphabet characters in a specified range of a string to uppercase.

IMString* im_string_assign (IMString * *string*, const IMChar * *cstr*, IMSize *len*)

Copy the content of specified C string to an **IMString** object.

Old content of the **IMString** object will be replaced with the new string.

Parameters:

string an **IMString** object
cstr a C string.
len length of the C string, -1 means until end of the string..

Returns:

Pointer pointed to the original string object.

IMString* im_string_assign_static (IMString * *string*, const IMChar * *cstr*)

Set a static C string as the content of specified **IMString** object.

Old content of the **IMString** object will be replaced with the new string.

Parameters:

string an **IMString** object
cstr a static C string.

Returns:

Pointer pointed to the original string object.

IMString* im_string_assign_text_utf8 (IMString * *string*, IMConstPointer *text*)

Copy the content of specified **IMText** object to an **IMString** object.

The content of the specified **IMText** object will be converted into UTF-8 encoded string and assigned to specified **IMString** object.

Parameters:

text Pointer to an **IMText** object.

const IMChar* im_string_c_str (const IMString * *string*)

Return the real content of a **IMString** object.

IMSize im_string_capacity (const IMString * *string*)

Returns the total number of characters that the **IMString** can hold before needing to allocate more memory.

void im_string_clear (IMString * *string*)

Erases the content of a **IMString** object, making it empty.

All content will be removed.

IMInt im_string_compare (const IMString * *string1*, const IMString * *string2*)

Compare two **IMString** objects.

Returns:

-1 means string1 < string2, 0 means they are equal, 1 means string1 > string2.

IMInt im_string_compare_str (const IMString * *string1*, const IMChar * *string2*)

Compare an **IMString** object with a C string.

Returns:

-1 means string1 < string2, 0 means they are equal, 1 means string1 > string2.

IMBool im_string_empty (const IMString * *string*)

Returns true if the **IMString** object is empty.

IMString* im_string_erase (IMString * *string*, IMSize *pos*, IMSize *len*)

Erase a specified portion of an **IMString** object.

Parameters:

string an **IMString** object.
pos position of the portion to be erased.
len length of the portion.

Returns:

string

IMString* IMString* IMString* IMString* IMSize im_string_find (const IMString * *string*, const IMString * *other*, IMSize *pos*)

Find position of a string.

Starting from pos, searches forward for value of other within this string. If found, returns the index where it begins. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
other an **IMString** object to locate inside string.
pos Index of character to search from

Returns:

Index of start of first occurrence, or IM_STRING_NPOS if not found.

IMSize im_string_find_char (const IMString * *string*, IMChar *ch*, IMSize *pos*)

Find position of a string.

Starting from pos, searches forward for specified character within this string. If found, returns the index where it begins. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
ch a character to locate inside string.
pos Index of character to search from

Returns:

Index of start of first occurrence, or IM_STRING_NPOS if not found.

IMSize im_string_find_first_of (const IMString * *string*, const IMString * *other*, IMSize *pos*)

Find position of a character of string.

Starting from pos, searches forward for one of the characters of other within this string. If found, returns the index where it was found. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
other String containing characters to locate.
pos Index of character to search from.

Returns:

Index of first occurrence, or IM_STRING_NPOS if not found.

IMSize im_string_find_first_of_str (const IMString * *string*, const IMChar * *other*, IMSize *pos*, IMSize *len*)

Find position of a character of string.

Starting from pos, searches forward for one of the characters of other within this string. If found, returns the index where it was found. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
other String containing characters to locate.
pos Index of character to search from.
len Number of characters from other to search for

Returns:

Index of first occurrence, or IM_STRING_NPOS if not found.

IMSize im_string_find_last_of (const IMString * *string*, const IMString * *other*, IMSize *pos*)

Find last position of a character of string.

Starting from pos, searches backward for one of the characters of other within this string. If found, returns the index where it was found. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
other String containing characters to locate.
pos Index of character to search from, IM_STRING_NPOS means starting from end of string.

Returns:

Index of first occurrence, or IM_STRING_NPOS if not found.

IMSize im_string_find_last_of_str (const IMString * *string*, const IMChar * *other*, IMSize *pos*, IMSize *len*)

Find last position of a character of string.

Starting from pos, searches backward for one of the characters of other within this string. If found, returns the index where it was found. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
other String containing characters to locate.
pos Index of character to search from. IM_STRING_NPOS means starting from end of string.
len Number of characters from other to search for

Returns:

Index of first occurrence, or IM_STRING_NPOS if not found.

IMSize im_string_find_str (const IMString * *string*, const IMChar * *other*, IMSize *pos*, IMSize *len*)

Find position of a string.

Starting from pos, searches forward for other within this string. If found, returns the index where it begins. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
other a C string to locate inside string.
pos Index of character to search from
len Number of characters from other to search for.

Returns:

Index of start of first occurrence, or IM_STRING_NPOS if not found.

IMChar im_string_get_char (const IMString * *string*, IMSize *idx*)

Get the character at specified index in an **IMString** object.

Parameters:

string an **IMString** object
idx the character index, must less than string length.

Returns:

the unicode character at the index, or 0 if the index is invalid.

IMString* im_string_get_left_portion (const IMString * *string*, const IMChar * *delim_chars*)

Get the left portion of a specified string, which is splitted by one of given delimiters. If there are more than one delimiters, then the string will be splitted by the left most delimiter. White space around the sub string will be trimmed.

For example, im_string_get_left_portion("abc = foo", "=") will return "abc".

The returned string should be freed afterwards by calling **im_object_unref()**;

IMString* im_string_get_right_portion (const IMString * *string*, const IMChar * *delim_chars*)

Get the right portion of a specified string, which is splitted by one of given delimiters. If there are more than one delimiters, then the string will be splitted by the left most delimiter. White space around the sub string will be trimmed.

For example, im_string_get_left_portion("abc = foo", "=") will return "foo".

The returned string should be freed afterwards by calling **im_object_unref()**;

IMString* im_string_get_sub_string (const IMString * *string*, IMSize *pos*, IMSize *len*)

Create a new **IMString** object containing a sub portion of a given **IMString** object.

Parameters:

string an **IMString** object.
pos start position of the sub portion.
len length of the sub portion.

Returns:

a newly created **IMString** object containing the sub string. It must be released by **im_object_unref()** afterwards.

IMUInt im_string_hash (const IMString * *string*)

Hash function of **IMString**. Used with **IMHashTable**.

IMString* im_string_insert (IMString * *string*, IMSize *pos*, const IMString * *other*)

Insert an **IMString** object into an **IMString** object at the specified position.

Parameters:

string an **IMString** object.
pos position in string.
other an **IMString** object to be inserted into string.

Returns:

string.

IMString* im_string_insert_char (IMString * *string*, IMSize *pos*, IMChar *ch*)

Insert an unicode character into an **IMString** object at the specified position.

Parameters:

string an **IMString** object.
pos position in string.
ch an character to be inserted into string.

Returns:

string.

IMString* IMString* IMString* im_string_insert_printf (IMString * *string*, IMSize *pos*, const IMChar * *format*, ...)

Insert arbitrary UTF-8 encoded printf result to a specified string at specified position.

Parameters:

string a **IMString** object.
format format string.
... arbitrary variables to be printed according to format string.

Returns:

string

IMString* im_string_insert_str (IMString * *string*, IMSize *pos*, const IMChar * *other*, IMSize *len*)

Insert a C string into the beginning of a **IMString** object.

Parameters:

string an **IMString** object.
pos position in string.
other a string which will be inserted into string.
len length of other, -1 means until end of the string.

Returns:

string.

IMSize im_string_length (const IMString * *string*)

Return length of an **IMString** object, in number of characters.

IMInt im_string_n_compare (const IMString * *string1*, const IMString * *string2*, IMSize *len*)

Compare two **IMString** objects for a specified length.

Only a specified length of string content will be compared.

IMInt im_string_n_compare_str (const IMString * *string1*, const IMChar * *string2*, IMSize *len*)

Compare an **IMString** object with a C string for a specified length.

Only a specified length of string content will be compared.

IMString* im_string_new (const IMChar * *cstr*)

Create a new **IMString** object with specified string as initial content.

IMString* im_string_new_from_printf (const IMChar * *format*, ...)

Create a new **IMString** object from arbitrary printf result.

Parameters:

format format string.
... arbitrary variables to be printed according to format string.

Returns:

the newly created **IMString** object.

IMString* IMString* im_string_new_from_text_utf8 (IMConstPointer *text*)

Create a new **IMString** object from a specified **IMText** object.

The content of the specified **IMText** object will be converted into UTF-8 encoded string and assigned to the newly created **IMString** object.

Parameters:

text Pointer to an **IMText** object.

IMString* im_string_new_length (const IMChar * *cstr*, IMSize *length*)

Create a new **IMString** object with specified string and length as initial content.

IMString* im_string_new_reserved (IMSize *capacity*)

Create a new **IMString** object with enough memory for specified number of characters to prevent reallocation when inserting many characters.

IMString* im_string_new_static (const IMChar * *cstr*)

Create a new **IMString** object with specified static string as initial content.

IMString* im_string_prepend (IMString * *string*, const IMString * *other*)

Prepend an **IMString** object into the beginning of a **IMString** object.

Parameters:

string an **IMString** object.
other an **IMString** object which will be prepended into string.

Returns:

string.

IMString* im_string_prepend_char (IMString * *string*, IMChar *ch*)

Prepend an unicode character into the beginning of a **IMString** object.

Parameters:

string an **IMString** object.
ch a character which will be prepended into string.

Returns:

string.

IMString* im_string_prepend_printf (IMString * *string*, const IMChar * *format*, ...)

Prepend arbitrary UTF-8 encoded printf result to a specified string.

Parameters:

string a **IMString** object.
format format string.
... arbitrary variables to be printed according to format string.

Returns:

string

IMString* im_string_prepend_str (IMString * *string*, const IMChar * *other*, IMSize *len*)

Prepend a C string into the beginning of a **IMString** object.

Parameters:

string an **IMString** object.
other a string which will be prepended into string.
len length of other, -1 means until end of the string.

Returns:

string.

IMString* IMString* IMString* IMString* im_string_printf (IMString * *string*, const IMChar * *format*, ...)

Print arbitrary string into an **IMString** object.

Old content of target **IMString** object will be erased.

Parameters:

string a **IMString** object.
format format string.
... arbitrary variables to be printed according to format string.

Returns:

string

void im_string_reserve (IMString * *string*, IMSize *size*)

Preallocate enough memory for specified number of characters to prevent reallocation when inserting many characters.

Parameters:

string an **IMString** object.
size number of characters required.

void im_string_resize (IMString * *string*, IMSize *new_size*, IMChar *fill_ch*)

Resizes the **IMString** object to the specified number of characters.

This function will resize the **IMString** object to the specified number of characters. If the number is smaller than the IMString's current size the **IMString** is truncated, otherwise the **IMString** is extended and new elements are set to fill_ch.

Parameters:

string **IMString** object to be resized.
new_size number of characters the **IMString** should contain.
fill_ch character to fill any new elements.

IMSize im_string_rfind (const IMString * *string*, const IMString * *other*, IMSize *pos*)

Find last position of a string.

Starting from pos, searches backward for value of other within this string. If found, returns the index where it begins. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
other an **IMString** object to locate inside string.
pos Index of character to search from, IM_STRING_NPOS means starting from end of string.

Returns:

Index of start of first occurrence, or IM_STRING_NPOS if not found.

IMSize im_string_rfind_char (const IMString * *string*, IMChar *ch*, IMSize *pos*)

Find last position of a string.

Starting from pos, searches backward for specified character within this string. If found, returns the index where it begins. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
ch a character to locate inside string.
pos Index of character to search from, IM_STRING_NPOS means starting from end of string.

Returns:

Index of start of first occurrence, or IM_STRING_NPOS if not found.

IMSize im_string_rfind_str (const IMString * *string*, const IMChar * *other*, IMSize *pos*, IMSize *len*)

Find last position of a string.

Starting from pos, searches backward for other within this string. If found, returns the index where it begins. If not found, returns IM_STRING_NPOS.

Parameters:

string an **IMString** object.
other a C string to locate inside string.
pos Index of character to search from, IM_STRING_NPOS means starting from end of string.
len Number of characters from other to search for.

Returns:

Index of start of first occurrence, or IM_STRING_NPOS if not found.

void im_string_set_char (IMString * *string*, IMSize *idx*, IMChar *ch*)

Set the character at specified index in an **IMString** object to a given character.

Parameters:

string an **IMString** object
idx the character index, must less than the length of this **IMString** object.
ch The character code to be set.

IMObjectArray* im_string_split (const IMString * *string*, const IMChar * *delim_chars*)

Split the specified string into sub strings by one of given delimiter chars.

Returns:

An object array which holds all sub strings. It must be released by caller.

void im_string_trim_blanks (IMString * *string*)

Remove blank characters at the begin and end of a string.

IMText

Data Structures

125 struct **IMText**

***IMText** class is used to manipulate a piece of multilingual text along with its text attributes.*

Defines

126 #define **IM_TEXT**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_TEXT,**IMText**))

127 #define **IM_CONST_TEXT**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_TEXT,**IMText**))

128 #define **IM_TEXT_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_TEXT,**IMTextClass**))

129 #define **IM_IS_TEXT**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_TEXT))

130 #define **IM_IS_TEXT_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_TEXT))

131 #define **IM_TEXT_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_TEXT,**IMTextClass**))

Typedefs

132 typedef struct **_IMText IMText**

133 typedef struct **_IMTextClass IMTextClass**

Functions

134 **IMText** * **im_text_new** ()

*Create a new empty **IMText** object.*

135 **IMText** * **im_text_new_from_utf8_string** (const **IMChar** *str, **IMInt** len)

*Create a new **IMText** object from an UTF-8 string.*

136 **IMText** * **im_text_new_from_utf8_printf** (const **IMChar** *format,...) IM_GNUC_PRINTF(1

*Create a new **IMText** object from arbitrary printf result.*

137 **IMText** ***IMText** * **im_text_new_from_utf16_string** (const **IMUInt16** *str, **IMInt** len)

*Create a new **IMText** object from an UTF-16 string.*

138 **IMText** * **im_text_new_from_ucs4_string** (const **IMUniChar** *str, **IMInt** len)

*Create a new **IMText** object from an UCS-4 string.*

139 void **im_text_clear** (**IMText** *text)

*Erases the content of a **IMText** object, making it empty.*

140 void **im_text_reserve** (**IMText** *text, **IMSize** size)

Preallocate enough memory for specified number of characters to prevent reallocation when inserting many characters.

141 **IMSize im_text_capacity** (const **IMText** *text)

*Returns the total number of characters that the **IMText** can hold before needing to allocate more memory.*

142 **IMBool im_text_empty** (const **IMText** *text)

*Returns true if the **IMText** object is empty.*

143 void **im_text_resize** (**IMText** *text, **IMSize** new_size, **IMUniChar** fill_ch)

*Resizes the **IMText** object to the specified number of characters.*

144 **IMText** * **im_text_from_utf8_string** (**IMText** *text, const **IMChar** *str, **IMInt** len)

*Load **IMText** content from an UTF-8 string.*

145 **IMText** * **im_text_from_utf16_string** (**IMText** *text, const **IMUInt16** *str, **IMInt** len)

*Load **IMText** content from an UTF-16 string.*

146 **IMText** * **im_text_from_ucs4_string** (**IMText** *text, const **IMUniChar** *str, **IMInt** len)

*Load **IMText** content from an UCS-4 string.*

147 **IMSize im_text_to_utf8_string** (const **IMText** *text, **IMChar** *dest, **IMSize** dest_size)

*Output the content of an **IMText** object to a buffer with UTF-8 encoding.*

148 **IMSize im_text_to_utf16_string** (const **IMText** *text, **IMUInt16** *dest, **IMSize** dest_size)

*Output the content of an **IMText** object to a buffer with UTF-16 encoding.*

149 **IMSize im_text_to_ucs4_string** (const **IMText** *text, **IMUniChar** *dest, **IMSize** dest_size)

*Output the content of an **IMText** object to a buffer with UCS-4 encoding.*

150 **IMSize im_text_length** (const **IMText** *text)

*Return length of an **IMText** object, in number of characters.*

151 **IMSize im_text_utf8_length** (const **IMText** *text)

*Return how many bytes space required when converting an **IMText** object to an utf-8 string. Not including the ending zero.*

152 **IMSize im_text_utf16_length** (const **IMText** *text)

*Return how many uint16 space required when converting an **IMText** object to an utf-16 string. Not including the ending zero.*

153 **IMUniChar im_text_get_char** (const **IMText** *text, **IMSize** idx)

*Get the character at specified index in an **IMText** object.*

154 void **im_text_set_char** (**IMText** *text, **IMSize** idx, **IMUniChar** ch)

*Set the character at specified index in an **IMText** object to a given character.*

155 **IMText** * **im_text_get_sub_text** (const **IMText** *text, **IMSize** pos, **IMSize** len)

*Create a new **IMText** object containing a sub portion of a given **IMText** object.*

156 **IMText** * **im_text_prepend** (**IMText** *text, const **IMText** *other)

*Prepend an **IMText** object into the beginning of a **IMText** object.*

157 **IMText** * **im_text_prepend_utf8_string** (**IMText** *text, const **IMChar** *str, **IMInt** len)

158 **IMText** * **im_text_prepend_utf16_string** (**IMText** *text, const **IMUInt16** *str, **IMInt** len)

159 **IMText** * **im_text_prepend_ucs4_string** (**IMText** *text, const **IMUniChar** *str, **IMInt** len)

160 **IMText** * **im_text_prepend_unichar** (**IMText** *text, **IMUniChar** ch)

*Prepend an unicode character into the beginning of a **IMText** object.*

161 **IMText** * **im_text_append** (**IMText** *text, const **IMText** *other)

*Append an **IMText** object into the end of a **IMText** object.*

162 **IMText** * **im_text_append_utf8_string** (**IMText** *text, const **IMChar** *str, **IMInt** len)

163 **IMText** * **im_text_append_utf16_string** (**IMText** *text, const **IMUInt16** *str, **IMInt** len)

164 **IMText** * **im_text_append_ucs4_string** (**IMText** *text, const **IMUniChar** *str, **IMInt** len)

165 **IMText** * **im_text_append_unichar** (**IMText** *text, **IMUniChar** ch)

*Append an unicode character into the end of a **IMText** object.*

166 **IMText** * **im_text_insert** (**IMText** *text, **IMSize** pos, const **IMText** *other)

*Insert an **IMText** object into an **IMText** object at the specified position.*

167 **IMText** * **im_text_insert_utf8_string** (**IMText** *text, **IMSize** pos, const **IMChar** *str, **IMInt** len)

168 **IMText** * **im_text_insert_utf16_string** (**IMText** *text, **IMSize** pos, const **IMUInt16** *str, **IMInt** len)

169 **IMText** * **im_text_insert_ucs4_string** (**IMText** *text, **IMSize** pos, const **IMUniChar** *str, **IMInt** len)

170 **IMText** * **im_text_insert_unichar** (**IMText** *text, **IMSize** pos, **IMUniChar** ch)

*Insert an unicode character into an **IMText** object at the specified position.*

171 **IMText** * **im_text_erase** (**IMText** *text, **IMSize** pos, **IMSize** len)

*Erase a specified portion of an **IMText** object.*

172 void **im_text_add_attribute** (**IMText** *text, **IMSize** pos, **IMSize** len, **IMAttribute** *attr)

*Add an attribute to a specified portion of an **IMText** object.*

173 void **im_text_remove_attribute** (**IMText** *text, **IMSize** pos, **IMSize** len, **IMType** attrtype)

*Remove attributes that match a given type in a specified portion of an **IMText** object.*

174 void **im_text_clear_attributes** (**IMText** *text, **IMSize** pos, **IMSize** len)

*Remove all attributes in a specified portion of an **IMText** object.*

175 **IMAttrList** * **im_text_get_attributes** (const **IMText** *text, **IMSize** pos, **IMSize** len)

*Get all attributes in a specified portion of an **IMText** object.*

176 void **im_text_set_attr_list** (**IMText** *text, **IMAttrList** *attrs)

*Set an **IMAttrList** object to an **IMText** object.*

177 **IMAttrList** * **im_text_get_attr_list** (const **IMText** *text)

*Get the **IMAttrList** object used by an **IMText** object.*

178 **IMAttrList** * **im_text_dup_attr_list** (const **IMText** *text)

*Duplicate the **IMAttrList** object used by an **IMText** object.*

179 **IMInt im_text_compare** (const **IMText** *text1, const **IMText** *text2)

*Compare two **IMText** objects.*

180 **IMInt im_text_n_compare** (const **IMText** *text1, const **IMText** *text2, **IMSize** len)

*Compare two **IMText** objects for a specified length.*

181 **IMText** * **im_text_prepend_utf8_printf** (**IMText** *text, const **IMChar** *format,...) IM_GNUC_PRINTF(2

Prepend arbitrary UTF-8 encoded printf result to a specified text.

182 **IMText** ***IMText** * **im_text_append_utf8_printf** (**IMText** *text, const **IMChar** *format,...) IM_GNUC_PRINTF(2

Append arbitrary UTF-8 encoded printf result to a specified text.

183 **IMText** ***IMText** ***IMText** * **im_text_insert_utf8_printf** (**IMText** *text, **IMSize** pos, const **IMChar** *format,...) IM_GNUC_PRINTF(3

Insert arbitrary UTF-8 encoded printf result to a specified text at specified position.

Define Documentation

#define IM_CONST_TEXT(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_TEXT,IMText))

#define IM_IS_TEXT(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_TEXT))

#define IM_IS_TEXT_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_TEXT))

#define IM_TEXT(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_TEXT,IMText))

#define IM_TEXT_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_TEXT,IMTextClass))

#define IM_TEXT_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_TEXT,IMTextClass))

Typedef Documentation

typedef struct _IMText IMText

typedef struct _IMTextClass IMTextClass

Function Documentation

void im_text_add_attribute (IMText * *text*, IMSize *pos*, IMSize *len*, IMAttribute * *attr*)

Add an attribute to a specified portion of an **IMText** object.

Ownership of the attribute will be assumed by the **IMText** object.

Parameters:

text an **IMText** object.
pos position of the portion.
len length of the portion.
attr attribute to be added to the portion.

IMText* im_text_append (IMText * *text*, const IMText * *other*)

Append an **IMText** object into the end of a **IMText** object.

Parameters:

text an **IMText** object.
other an **IMText** object which will be appended into text.

Returns:

text.

IMText* im_text_append_ucs4_string (IMText * *text*, const IMUniChar * *str*, IMInt *len*)

IMText* im_text_append_unichar (IMText * *text*, IMUniChar *ch*)

Append an unicode character into the end of a **IMText** object.

Parameters:

text an **IMText** object.
ch a character which will be appended into text.

Returns:

text.

IMText* im_text_append_utf16_string (IMText * *text*, const IMUInt16 * *str*, IMInt *len*)

IMText* IMText* im_text_append_utf8_printf (IMText * *text*, const IMChar * *format*, ...)

Append arbitrary UTF-8 encoded printf result to a specified text.

Parameters:

text a **IMText** object.
format format string.
... arbitrary variables to be printed according to format string.

Returns:

text

IMText* im_text_append_utf8_string (IMText * *text*, const IMChar * *str*, IMInt *len*)

IMSize im_text_capacity (const IMText * *text*)

Returns the total number of characters that the **IMText** can hold before needing to allocate more memory.

void im_text_clear (IMText * *text*)

Erases the content of a **IMText** object, making it empty.

All content will be removed.

void im_text_clear_attributes (IMText * *text*, IMSize *pos*, IMSize *len*)

Remove all attributes in a specified portion of an **IMText** object.

Parameters:

text an **IMText** object.
pos position of the portion.
len length of the portion.

IMInt im_text_compare (const IMText * *text1*, const IMText * *text2*)

Compare two **IMText** objects.

Only text content will be compared.

IMAttrList* im_text_dup_attr_list (const IMText * *text*)

Duplicate the **IMAttrList** object used by an **IMText** object.

Returns:

a copy of the **IMAttrList** object currently used by text. i It should be released by caller.

IMBool im_text_empty (const IMText * *text*)

Returns true if the **IMText** object is empty.

IMText* im_text_erase (IMText * *text*, IMSize *pos*, IMSize *len*)

Erase a specified portion of an **IMText** object.

Parameters:

text an **IMText** object.
pos position of the portion to be erased.
len length of the portion.

Returns:

text

IMText* im_text_from_ucs4_string (IMText * *text*, const IMUniChar * *str*, IMInt *len*)

Load **IMText** content from an UCS-4 string.

Byte order of the UCS-4 string must be same as the machine endian. And this function won't recognize BOM. Old content and attributes will be lost.

Parameters:

text a **IMText** object
str an UCS-4 string.
len Length of str, -1 means until 0.

Returns:

the original **IMText** object on success, 0 if failed.

IMText* im_text_from_utf16_string (IMText * *text*, const IMUInt16 * *str*, IMInt *len*)

Load **IMText** content from an UTF-16 string.

Byte order of the UTF-16 string must be same as the machine endian. And this function won't recognize BOM. Old content and attributes will be lost.

Parameters:

text a **IMText** object
str an UTF-16 string.
len Length of str, -1 means until 0.

Returns:

the original **IMText** object on success, 0 if failed.

IMText* im_text_from_utf8_string (IMText * *text*, const IMChar * *str*, IMInt *len*)

Load **IMText** content from an UTF-8 string.

Old content and attributes will be lost.

Parameters:

text a **IMText** object
str an UTF-8 string.
len Length of str, -1 means until 0.

Returns:

the original **IMText** object on success, 0 if failed.

IMAttrList* im_text_get_attr_list (const IMText * *text*)

Get the **IMAttrList** object used by an **IMText** object.

Returns:

the **IMAttrList** object currently used by text. It shouldn't be changed or released by caller.

IMAttrList* im_text_get_attributes (const IMText * *text*, IMSize *pos*, IMSize *len*)

Get all attributes in a specified portion of an **IMText** object.

The range of returned attributes are relative to pos.

Parameters:

text an **IMText** object.
pos position of the portion.
len length of the portion.

Returns:

a newly created **IMAttrList** object containing the attributes. It must be released by **im_object_unref()** afterwards.

IMUniChar im_text_get_char (const IMText * *text*, IMSize *idx*)

Get the character at specified index in an **IMText** object.

Parameters:

text an **IMText** object
idx the character index, must less than text length.

Returns:

the unicode character at the index, or IM_UNICHAR_INVALID if the index is invalid.

IMText* im_text_get_sub_text (const IMText * *text*, IMSize *pos*, IMSize *len*)

Create a new **IMText** object containing a sub portion of a given **IMText** object.

Parameters:

text an **IMText** object.
pos start position of the sub portion.
len length of the sub portion.

Returns:

a newly created **IMText** object containing the sub text. It must be released by **im_object_unref()** afterwards.

IMText* im_text_insert (IMText * *text*, IMSize *pos*, const IMText * *other*)

Insert an **IMText** object into an **IMText** object at the specified position.

Parameters:

text an **IMText** object.
pos position in text.
other an **IMText** object to be inserted into text.

Returns:

text.

IMText* im_text_insert_ucs4_string (IMText * *text*, IMSize *pos*, const IMUniChar * *str*, IMInt *len*)

IMText* im_text_insert_unichar (IMText * *text*, IMSize *pos*, IMUniChar *ch*)

Insert an unicode character into an **IMText** object at the specified position.

Parameters:

text an **IMText** object.
pos position in text.
ch an character to be inserted into text.

Returns:

text.

IMText* im_text_insert_utf16_string (IMText * *text*, IMSize *pos*, const IMUInt16 * *str*, IMInt *len*)

IMText* IMText* IMText* im_text_insert_utf8_printf (IMText * *text*, IMSize *pos*, const IMChar * *format*, ...)

Insert arbitrary UTF-8 encoded printf result to a specified text at specified position.

Parameters:

text a **IMText** object.
format format string.
... arbitrary variables to be printed according to format string.

Returns:

text

IMText* im_text_insert_utf8_string (IMText * *text*, IMSize *pos*, const IMChar * *str*, IMInt *len*)

IMSize im_text_length (const IMText * *text*)

Return length of an **IMText** object, in number of characters.

IMInt im_text_n_compare (const IMText * *text1*, const IMText * *text2*, IMSize *len*)

Compare two **IMText** objects for a specified length.

Only a specified length of text content will be compared.

IMText* im_text_new ()

Create a new empty **IMText** object.

IMText* im_text_new_from_ucs4_string (const IMUniChar * *str*, IMInt *len*)

Create a new **IMText** object from an UCS-4 string.

Byte order of the UCS-4 string must be same as the machine endian. And this function won't recognize BOM.

Parameters:

str an UCS-4 string.
len Length of str, -1 means until 0.

Returns:

the newly created **IMText** object.

IMText* IMText* im_text_new_from_utf16_string (const IMUInt16 * *str*, IMInt *len*)

Create a new **IMText** object from an UTF-16 string.

Byte order of the UTF-16 string must be same as the machine endian. And this function won't recognize BOM.

Parameters:

str an UTF-16 string.
len Length of str, -1 means until 0.

Returns:

the newly created **IMText** object.

IMText* im_text_new_from_utf8_printf (const IMChar * *format*, ...)

Create a new **IMText** object from arbitrary printf result.

The result string must be UTF-8 encoded.

Parameters:

format format string.
... arbitrary variables to be printed according to format string.

Returns:

the newly created **IMText** object.

IMText* im_text_new_from_utf8_string (const IMChar * *str*, IMInt *len*)

Create a new **IMText** object from an UTF-8 string.

Parameters:

str an UTF-8 string.
len Length of str, -1 means until 0.

Returns:

the newly created **IMText** object.

IMText* im_text_prepend (IMText * *text*, const IMText * *other*)

Prepend an **IMText** object into the beginning of a **IMText** object.

Parameters:

text an **IMText** object.
other an **IMText** object which will be prepended into text.

Returns:

text.

IMText* im_text_prepend_ucs4_string (IMText * *text*, const IMUniChar * *str*, IMInt *len*)

IMText* im_text_prepend_unichar (IMText * *text*, IMUniChar *ch*)

Prepend an unicode character into the beginning of a **IMText** object.

Parameters:

text an **IMText** object.
ch a character which will be prepended into text.

Returns:

text.

IMText* im_text_prepend_utf16_string (IMText * *text*, const IMUInt16 * *str*, IMInt *len*)

IMText* im_text_prepend_utf8_printf (IMText * *text*, const IMChar * *format*, ...)

Prepend arbitrary UTF-8 encoded printf result to a specified text.

Parameters:

text a **IMText** object.
format format string.
... arbitrary variables to be printed according to format string.

Returns:

text

IMText* im_text_prepend_utf8_string (IMText * *text*, const IMChar * *str*, IMInt *len*)

void im_text_remove_attribute (IMText * *text*, IMSize *pos*, IMSize *len*, IMType *attrtype*)

Remove attributes that match a given type in a specified portion of an **IMText** object.

Parameters:

text an **IMText** object.
pos position of the portion.
len length of the portion.
attrtype attribute type to be removed.

void im_text_reserve (IMText * *text*, IMSize *size*)

Preallocate enough memory for specified number of characters to prevent reallocation when inserting many characters.

Parameters:

text an **IMText** object.
size number of characters required.

void im_text_resize (IMText * *text*, IMSize *new_size*, IMUniChar *fill_ch*)

Resizes the **IMText** object to the specified number of characters.

This function will resize the **IMText** object to the specified number of characters. If the number is smaller than the IMText's current size the **IMText** is truncated, otherwise the **IMText** is extended and new elements are set to fill_ch.

Parameters:

text **IMText** object to be resized.
new_size number of characters the **IMText** should contain.
fill_ch character to fill any new elements.

void im_text_set_attr_list (IMText * *text*, IMAttrList * *attrs*)

Set an **IMAttrList** object to an **IMText** object.

Old list will be released. Ownership of of the new list will be assumed by the **IMText** object.

Parameters:

text the **IMText** object.
attrs the **IMAttrList** object.

void im_text_set_char (IMText * *text*, IMSize *idx*, IMUniChar *ch*)

Set the character at specified index in an **IMText** object to a given character.

Parameters:

text an **IMText** object
idx the character index, must less than the length of this **IMText** object.
ch The character code to be set, if it's 0 then the text will be truncated.

IMSize im_text_to_ucs4_string (const IMText * *text*, IMUniChar * *dest*, IMSize *dest_size*)

Output the content of an **IMText** object to a buffer with UCS-4 encoding.

The result UCS-4 string has the same byte order than current machine endian.

Parameters:

text an **IMText** object
dest a buffer to hold result ucs4 string.
dest_size size of dest buffer.

Returns:

How many IMUniChar elements actually written into dest buffer.

IMSize im_text_to_utf16_string (const IMText * *text*, IMUInt16 * *dest*, IMSize *dest_size*)

Output the content of an **IMText** object to a buffer with UTF-16 encoding.

The result UTF-16 string has the same byte order than current machine endian.

Parameters:

text an **IMText** object
dest a buffer to hold result utf-16 string.
dest_size size of dest buffer.

Returns:

How many uint16 elements actually written into dest buffer.

IMSize im_text_to_utf8_string (const IMText * *text*, IMChar * *dest*, IMSize *dest_size*)

Output the content of an **IMText** object to a buffer with UTF-8 encoding.

Parameters:

text an **IMText** object
dest a buffer to hold result utf-8 string.
dest_size size of dest buffer.

Returns:

How many bytes actually written into dest buffer.

IMSize im_text_utf16_length (const IMText * *text*)

Return how many uint16 space required when converting an **IMText** object to an utf-16 string. Not including the ending zero.

This function can be used to determine the minimum required buffer size when using **im_text_to_utf16_string()** to convert an **IMText** object to an utf-16 string.

IMSize im_text_utf8_length (const IMText * *text*)

Return how many bytes space required when converting an **IMText** object to an utf-8 string. Not including the ending zero.

This function can be used to determine the minimum required buffer size when using **im_text_to_utf8_string()** to convert an **IMText** object to an utf-8 string.

IMTypeClass

184 struct **_IMTypeClass**

185 struct **_IMTypeInstance**

186 struct **_IMTypeInfo**

187 struct **IMTypeClass**

The base struct of all type class structs.

188 struct **IMTypeInstance**

The base struct of all class type instances.

Defines

189 #define **IM_TYPE_INSTANCE_CHECK_CAST**(instance, type, cast_type) ((cast_type*)im_type_instance_check_cast(instance, type))

190 #define **IM_TYPE_INSTANCE_CHECK_CAST_CONST**(instance, type, cast_type) ((const cast_type*)im_type_instance_check_cast_const(instance, type))

191 #define **IM_TYPE_CLASS_CHECK_CAST**(klass, type, cast_type) ((const cast_type*)im_type_class_check_cast(klass, type))

192 #define **IM_TYPE_INSTANCE_CHECK_TYPE**(instance, type) (im_type_instance_is_a(instance,type))

193 #define **IM_TYPE_CLASS_CHECK_TYPE**(klass, type) (im_type_class_is_a(klass,type))

194 #define **IM_TYPE_INSTANCE_GET_CLASS_CAST**(instance, type, cast_type) (IM_TYPE_CLASS_CHECK_CAST(im_type_instance_get_class(instance),type,cast_type))

Typedefs

195 typedef void(* **IMInstanceInitFunc**)(**IMPointer** instance)

196 typedef void(* **IMInstanceFinalizeFunc**)(**IMPointer** instance)

197 typedef void(* **IMClassInitFunc**)(**IMPointer** klass)

198 typedef void(* **IMClassFinalizeFunc**)(**IMPointer** klass)

199 typedef struct **_IMTypeClass IMTypeClass**

200 typedef struct **_IMTypeInstance IMTypeInstance**

201 typedef struct **_IMTypeInfo IMTypeInfo**

Enumerations

202 enum { **IM_TYPE_INVALID** = 0, **IM_TYPE_VOID**, **IM_TYPE_CHAR**, **IM_TYPE_UCHAR**, **IM_TYPE_BOOL**, **IM_TYPE_INT16**, **IM_TYPE_UINT16**, **IM_TYPE_INT32**, **IM_TYPE_UINT32**, **IM_TYPE_INT64**, **IM_TYPE_UINT64**, **IM_TYPE_DOUBLE**, **IM_TYPE_C_STRING**, **IM_TYPE_POINTER**, **IM_TYPE_BYTE_ARRAY**, **IM_TYPE_BOOL_ARRAY**, **IM_TYPE_INT16_ARRAY**, **IM_TYPE_UINT16_ARRAY**, **IM_TYPE_INT32_ARRAY**, **IM_TYPE_UINT32_ARRAY**, **IM_TYPE_INT64_ARRAY**, **IM_TYPE_UINT64_ARRAY**, **IM_TYPE_DOUBLE_ARRAY**, **IM_TYPE_LAST_FUNDAMENTAL** = IM_TYPE_DOUBLE_ARRAY, **IM_TYPE_OBJECT** = 32, **IM_TYPE_VALUE**, **IM_TYPE_VALUE_ARRAY**, **IM_TYPE_OBJECT_ARRAY**, **IM_TYPE_OBJECT_QUEUE**, **IM_TYPE_MEMORY_CHUNK**, **IM_TYPE_STREAM**, **IM_TYPE_BYTE_STREAM**, **IM_TYPE_TEXT_STREAM**, **IM_TYPE_ATTRIBUTE**, **IM_TYPE_ATTR_LANGUAGE**, **IM_TYPE_ATTR_FONT_FAMILY**, **IM_TYPE_ATTR_FONT_STYLE**, **IM_TYPE_ATTR_FONT_WEIGHT**, **IM_TYPE_ATTR_FONT_SIZE**, **IM_TYPE_ATTR_FONT_SCALE**, **IM_TYPE_ATTR_FOREGROUND**, **IM_TYPE_ATTR_BACKGROUND**, **IM_TYPE_ATTR_UNDERLINE**, **IM_TYPE_ATTR_UNDERLINE_COLOR**, **IM_TYPE_ATTR_STRIKETHROUGH**, **IM_TYPE_ATTR_STRIKETHROUGH_COLOR**, **IM_TYPE_ATTR_HIGHLIGHT**, **IM_TYPE_ATTR_REVERSE**, **IM_TYPE_ATTR_STRING**, **IM_TYPE_ATTR_OBJECT**, **IM_TYPE_ATTR_LIST**, **IM_TYPE_STRING**, **IM_TYPE_TEXT**, **IM_TYPE_CANDIDATE**, **IM_TYPE_LOOKUP_TABLE**, **IM_TYPE_PROPERTY**, **IM_TYPE_PROP_LIST**, **IM_TYPE_EVENT**, **IM_TYPE_EVENT_ROLES**, **IM_TYPE_HOTKEY**,

IM_TYPE_HOTKEY_PROFILE, **IM_TYPE_IO_CHANNEL**, **IM_TYPE_IO_CHANNEL_UNIX**, **IM_TYPE_CONNECTION**, **IM_TYPE_MAIN_LOOP**, **IM_TYPE_MAIN_LOOP_UNIX**, **IM_TYPE_SERVER**, **IM_TYPE_SERVER_UNIX**, **IM_TYPE_ADDRESS**, **IM_TYPE_ADDRESS_UNIX**, **IM_TYPE_MODULE**, **IM_TYPE_LAST_BUILTIN** = IM_TYPE_MODULE, **IM_TYPE_FIRST_USER_DEFINED** = IM_TYPE_LAST_BUILTIN + 1 }

Predefined built-in types, including all fundamental types, as well as various class types.

Functions

203 void **im_type_init** ()

Init type system.

204 **IMType im_type_register_class** (**IMType** parent, const **IMChar** *class_name, const **IMTypeInfo** *type_info, **IMBool** instantiatable)

Register a new class.

205 **IMBool im_type_deregister_class** (**IMType** type)

Deregister a class type.

206 const **IMChar** * **im_type_get_name** (**IMType** type)

Get the name of a type.

207 **IMType im_type_from_name** (const **IMChar** *name)

Get a type by its name.

208 **IMType im_type_get_parent** (**IMType** type)

Get the parent type of a type.

209 **IMBool im_type_is_fundamental** (**IMType** type)

Check whether a type is a fundamental type.

210 **IMBool im_type_is_built_in** (**IMType** type)

Check whether a type is a built-in type or not.

211 **IMBool im_type_is_instantiatable** (**IMType** type)

Check whether a type is instantiatable or not.

212 **IMSize im_type_get_instance_size** (**IMType** type)

Get instance size of a type.

213 **IMBool im_type_is_a** (**IMType** type, **IMType** is_a_type)

Check whether a type is derived from another type or not.

214 **IMBool im_type_is_referenced** (**IMType** type)

Check whether a specified type is referenced.

215 **IMBool im_type_is_valid** (**IMType** type)

Check whether a specified type is a valid type.

216 **IMBool im_type_class_valid** (**IMConstPointer** klass)

Check whether a pointer is pointed to a valid class struct instance or not.

217 const **IMChar** * **im_type_class_get_name** (**IMConstPointer** klass)

Get the name of an type class struct.

218 **IMType im_type_class_get_type** (**IMConstPointer** klass)

Get the type of an type class struct.

219 **IMConstPointer im_type_class_get_parent** (**IMConstPointer** klass)

Get the parent of an type class struct.

220 **IMConstPointer im_type_class_from_name** (const **IMChar** *name)

Get a type class struct instance by its name.

221 **IMConstPointer im_type_class_from_type** (**IMType** type)

Get a type class struct instance by its type.

222 **IMBool im_type_class_is_a** (**IMConstPointer** klass, **IMType** is_a_type)

Check whether a type class is derived from a parent class type or not.

223 **IMConstPointer im_type_class_check_cast** (**IMConstPointer** klass, **IMType** is_a_type)

Cast a type class pointer to a parent type.

224 **IMSize im_type_class_sizeof** (**IMConstPointer** klass)

Return the size of a type class.

225 **IMPointer im_type_instance_new** (**IMType** type)

Create a new type instance for a given type.

226 void **im_type_instance_delete** (**IMPointer** instance)

Delete a type instance.

227 **IMSize im_type_instance_sizeof** (**IMConstPointer** instance)

Get the size of a type instance.

228 **IMBool im_type_instance_valid** (**IMConstPointer** instance)

Check whether a type instance is valid or not.

229 **IMType im_type_instance_get_type** (**IMConstPointer** instance)

Get the class type of a type instance.

230 **IMConstPointer im_type_instance_get_class** (**IMConstPointer** instance)

Get class struct pointer of a type instance.

231 **IMBool im_type_instance_is_a** (**IMConstPointer** instance, **IMType** is_a_type)

Check whether a type instance inherits a parent type.

232 **IMPointer im_type_instance_check_cast** (**IMPointer** instance, **IMType** is_a_type)

Cast a type instance to a parent type.

233 **IMConstPointer im_type_instance_check_cast_const** (**IMConstPointer** instance, **IMType** is_a_type)

Cast a const type instance to a parent type.

234 void **im_type_print_all_type_informations** ()

Print information of all types.

Define Documentation

#define IM_TYPE_CLASS_CHECK_CAST(klass, type, cast_type) ((const cast_type*)im_type_class_check_cast(klass, type))

#define IM_TYPE_CLASS_CHECK_TYPE(klass, type) (im_type_class_is_a(klass,type))

#define IM_TYPE_INSTANCE_CHECK_CAST(instance, type, cast_type) ((cast_type*)im_type_instance_check_cast(instance, type))

#define IM_TYPE_INSTANCE_CHECK_CAST_CONST(instance, type, cast_type) ((const cast_type*)im_type_instance_check_cast_const(instance, type))

#define IM_TYPE_INSTANCE_CHECK_TYPE(instance, type) (im_type_instance_is_a(instance,type))

#define IM_TYPE_INSTANCE_GET_CLASS_CAST(instance, type, cast_type) (IM_TYPE_CLASS_CHECK_CAST(im_type_instance_get_class(instance),type,cast_type))

Typedef Documentation

typedef void(* IMClassFinalizeFunc)(IMPointer klass)

typedef void(* IMClassInitFunc)(IMPointer klass)

typedef void(* IMInstanceFinalizeFunc)(IMPointer instance)

typedef void(* IMInstanceInitFunc)(IMPointer instance)

typedef struct _IMTypeClass IMTypeClass

typedef struct _IMTypeInfo IMTypeInfo

typedef struct _IMTypeInstance IMTypeInstance

Enumeration Type Documentation

anonymous enum

Predefined built-in types, including all fundamental types, as well as various class types.

Enumerator:

IM_TYPE_INVALID
IM_TYPE_VOID A placeholder type >
IM_TYPE_CHAR
IM_TYPE_UCHAR
IM_TYPE_BOOL
IM_TYPE_INT16
IM_TYPE_UINT16
IM_TYPE_INT32
IM_TYPE_UINT32
IM_TYPE_INT64
IM_TYPE_UINT64
IM_TYPE_DOUBLE
IM_TYPE_C_STRING Plain old C string, not IM_TYPE_STRING object >
IM_TYPE_POINTER
IM_TYPE_BYTE_ARRAY
IM_TYPE_BOOL_ARRAY
IM_TYPE_INT16_ARRAY
IM_TYPE_UINT16_ARRAY
IM_TYPE_INT32_ARRAY
IM_TYPE_UINT32_ARRAY
IM_TYPE_INT64_ARRAY
IM_TYPE_UINT64_ARRAY
IM_TYPE_DOUBLE_ARRAY
IM_TYPE_LAST_FUNDAMENTAL
IM_TYPE_OBJECT
IM_TYPE_VALUE
IM_TYPE_VALUE_ARRAY
IM_TYPE_OBJECT_ARRAY
IM_TYPE_OBJECT_QUEUE
IM_TYPE_MEMORY_CHUNK
IM_TYPE_STREAM
IM_TYPE_BYTE_STREAM
IM_TYPE_TEXT_STREAM
IM_TYPE_ATTRIBUTE
IM_TYPE_ATTR_LANGUAGE

IM_TYPE_ATTR_FONT_FAMILY
IM_TYPE_ATTR_FONT_STYLE
IM_TYPE_ATTR_FONT_WEIGHT
IM_TYPE_ATTR_FONT_SIZE
IM_TYPE_ATTR_FONT_SCALE
IM_TYPE_ATTR_FOREGROUND
IM_TYPE_ATTR_BACKGROUND
IM_TYPE_ATTR_UNDERLINE
IM_TYPE_ATTR_UNDERLINE_COLOR
IM_TYPE_ATTR_STRIKETHROUGH
IM_TYPE_ATTR_STRIKETHROUGH_COLOR
IM_TYPE_ATTR_HIGHLIGHT
IM_TYPE_ATTR_REVERSE
IM_TYPE_ATTR_STRING
IM_TYPE_ATTR_OBJECT
IM_TYPE_ATTR_LIST
IM_TYPE_STRING
IM_TYPE_TEXT
IM_TYPE_CANDIDATE
IM_TYPE_LOOKUP_TABLE
IM_TYPE_PROPERTY
IM_TYPE_PROP_LIST
IM_TYPE_EVENT
IM_TYPE_EVENT_ROLES
IM_TYPE_HOTKEY
IM_TYPE_HOTKEY_PROFILE
IM_TYPE_IO_CHANNEL
IM_TYPE_IO_CHANNEL_UNIX
IM_TYPE_CONNECTION
IM_TYPE_MAIN_LOOP
IM_TYPE_MAIN_LOOP_UNIX
IM_TYPE_SERVER
IM_TYPE_SERVER_UNIX
IM_TYPE_ADDRESS
IM_TYPE_ADDRESS_UNIX
IM_TYPE_MODULE
IM_TYPE_LAST_BUILTIN
IM_TYPE_FIRST_USER_DEFINED

Function Documentation

IMConstPointer im_type_class_check_cast (IMConstPointer *klass*, IMType *is_a_type*)

Cast a type class pointer to a parent type.

Parameters:

klass The type class pointer to be casted.
is_a_type The parent type.

Returns:

If cast can be done then klass will be returned, otherwise return 0.

IMConstPointer im_type_class_from_name (const IMChar * *name*)

Get a type class struct instance by its name.

IMConstPointer im_type_class_from_type (IMType *type*)

Get a type class struct instance by its type.

const IMChar* im_type_class_get_name (IMConstPointer *klass*)

Get the name of an type class struct.

Parameters:

klass Pointer to the type class.

IMConstPointer im_type_class_get_parent (IMConstPointer *klass*)

Get the parent of an type class struct.

Parameters:

klass Pointer to the type class.

IMType im_type_class_get_type (IMConstPointer *klass*)

Get the type of an type class struct.

Parameters:

klass Pointer to the type class.

IMBool im_type_class_is_a (IMConstPointer *klass*, IMType *is_a_type*)

Check whether a type class is derived from a parent class type or not.

Parameters:

klass A pointer to the type class struct to be checked.
is_a_type The parent type.

Returns:

True if klass is derived from is_a_type.

IMSize im_type_class_sizeof (IMConstPointer *klass*)

Return the size of a type class.

Parameters:

klass The type class pointer.

IMBool im_type_class_valid (IMConstPointer *klass*)

Check whether a pointer is pointed to a valid class struct instance or not. This fucntion return true if succeeded, otherwise return false.

IMBool im_type_deregister_class (IMType *type*)

Deregister a class type.

A class type can only be deregistered if there is no instance instantiated from it and no class derived from it.

This fucntion return true if succeeded, otherwise return false.

IMType im_type_from_name (const IMChar * *name*)

Get a type by its name.

IMSize im_type_get_instance_size (IMType *type*)

Get instance size of a type.

const IMChar* im_type_get_name (IMType *type*)

Get the name of a type.

IMType im_type_get_parent (IMType *type*)

Get the parent type of a type.

void im_type_init ()

Init type system.

This function must be called at beginning of the program.

IMPointer im_type_instance_check_cast (IMPointer *instance*, IMType *is_a_type*)

Cast a type instance to a parent type.

Returns:

the instance itself if cast can be done, otherwise 0.

IMConstPointer im_type_instance_check_cast_const (IMConstPointer *instance*, IMType *is_a_type*)

Cast a const type instance to a parent type.

Returns:

the type instance itself if cast can be done, otherwise 0.

void im_type_instance_delete (IMPointer *instance*)

Delete a type instance.

Parameters:

instance Pointer to the instance.

IMConstPointer im_type_instance_get_class (IMConstPointer *instance*)

Get class struct pointer of a type instance.

IMType im_type_instance_get_type (IMConstPointer *instance*)

Get the class type of a type instance.

IMBool im_type_instance_is_a (IMConstPointer *instance*, IMType *is_a_type*)

Check whether a type instance inherits a parent type. This fucntion return true if succeeded, otherwise return false.

IMPointer im_type_instance_new (IMType *type*)

Create a new type instance for a given type.

This function will allocate required memory and call constructors of the given class type and its parent class types in correct sequence to initialize the instance.

Parameters:

type Type of the class.

Returns:

A pointer to the newly created instance.

IMSize im_type_instance_sizeof (IMConstPointer *instance*)

Get the size of a type instance.

IMBool im_type_instance_valid (IMConstPointer *instance*)

Check whether a type instance is valid or not. This fucntion return true if succeeded, otherwise return false.

IMBool im_type_is_a (IMType *type*, IMType *is_a_type*)

Check whether a type is derived from another type or not.

Parameters:

type The type to be checked.
is_a_type A parent type.

Returns:

True if type is derived from is_a_type.

IMBool im_type_is_built_in (IMType *type*)

Check whether a type is a built-in type or not.

A built-in type is a type definedby IMFramework, which has constant type id. This fucntion return true if succeeded, otherwise return false.

IMBool im_type_is_fundamental (IMType *type*)

Check whether a type is a fundamental type.

types such as void, char, bool, int32, int64, double, string, byte array, etc. are fundamental. This fucntion return true if succeeded, otherwise return false.

IMBool im_type_is_instantiatable (IMType *type*)

Check whether a type is instantiatable or not.

A type is only instantiatable if it's a fundamental type or non-pure virtual class. This fucntion return true if succeeded, otherwise return false.

IMBool im_type_is_referenced (IMType *type*)

Check whether a specified type is referenced.

A type is referenced, if an instance of this type is created, or this type is inherited by another type. This fucntion return true if succeeded, otherwise return false.

IMBool im_type_is_valid (IMType *type*)

Check whether a specified type is a valid type. This fucntion return true if succeeded, otherwise return false.

void im_type_print_all_type_informations ()

Print information of all types.

It's for debug purpose.

IMType im_type_register_class (IMType *parent*, const IMChar * *class_name*, const IMTypeInfo * *type_info*, IMBool *instantiatable*)

Register a new class.

Register a new class into type system, so that it can be handled by type system.

Parameters:

parent Type of parent class.
class_name Name of the new class, must be a static string.
type_info Information of the type.
instantiatable Whether it's a instantiatable class. A class with pure virtual methods can't be instantiated.

Returns:

Type of the new class.

IMValue

235 struct **IMValue**

A container object which can hold one datum, The datum can be a bool, a char/uchar, an int32/uint32, an int64/uint64, a double, a pointer, a string or an object.

Defines

236 #define **IM_VALUE**(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_VALUE,**IMValue**))

237 #define **IM_CONST_VALUE**(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_VALUE,**IMValue**))

238 #define **IM_VALUE_CLASS**(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_VALUE,**IMValueClass**))

239 #define **IM_IS_VALUE**(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_VALUE))

240 #define **IM_IS_VALUE_CLASS**(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_VALUE))

241 #define **IM_VALUE_GET_CLASS**(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_VALUE,**IMValueClass**))

242 #define **IM_VALUE_TYPE**(p) (im_value_get_value_type (p))

243 #define **IM_VALUE_TYPE_NAME**(p) (im_type_get_name(IM_VALUE_TYPE(p)))

Typedefs

244 typedef struct **_IMValue IMValue**

245 typedef struct **_IMValueClass IMValueClass**

Functions

246 **IMValue** * **im_value_new** ()

*Create a new zero-filled **IMValue** object.*

247 **IMValue** * **im_value_new_char** (**IMChar** v_char)

*Create a new **IMValue** object that holds a char value.*

248 **IMValue** * **im_value_new_uchar** (**IMUChar** v_uchar)

*Create a new **IMValue** object that holds a unsigned char value.*

249 **IMValue** * **im_value_new_bool** (**IMBool** v_bool)

*Create a new **IMValue** object that holds a bool value.*

250 **IMValue** * **im_value_new_int16** (**IMInt16** v_int16)

*Create a new **IMValue** object that holds an int16 value.*

251 **IMValue** * **im_value_new_uint16** (**IMUInt16** v_uint16)

*Create a new **IMValue** object that holds a unsigned int16 value.*

252 **IMValue** * **im_value_new_int32** (**IMInt32** v_int32)

*Create a new **IMValue** object that holds an int32 value.*

253 **IMValue** * **im_value_new_uint32** (**IMUInt32** v_uint32)

*Create a new **IMValue** object that holds a unsigned int32 value.*

254 **IMValue** * **im_value_new_int64** (**IMInt64** v_int64)

*Create a new **IMValue** object that holds an int64 value.*

255 **IMValue** * **im_value_new_uint64** (**IMUInt64** v_uint64)

*Create a new **IMValue** object that holds a unsigned int64 value.*

256 **IMValue** * **im_value_new_double** (**IMDouble** v_double)

*Create a new **IMValue** object that holds a double value.*

257 **IMValue** * **im_value_new_pointer** (**IMPointer** v_pointer)

*Create a new **IMValue** object that holds a pointer.*

258 **IMValue** * **im_value_new_c_string** (const **IMChar** *v_string)

*Create a new **IMValue** object that holds a string.*

259 **IMValue** * **im_value_new_static_c_string** (const **IMChar** *v_string)

*Create a new **IMValue** object that holds a static string.*

260 **IMValue** * **im_value_new_take_c_string** (**IMChar** *v_string)

*Create a new **IMValue** object that takes an ownership of a newly allocated string.*

261 **IMValue** * **im_value_new_object** (**IMPointer** v_object)

*Create a new **IMValue** object that holds an object.*

262 **IMType im_value_get_value_type** (const **IMValue** *value)

*Get the value type of a value holded in a **IMValue** object.*

263 void **im_value_clear** (**IMValue** *value)

*Clear the value of a **IMValue** object.*

264 void **im_value_set_char** (**IMValue** *value, **IMChar** v_char)

*Store a char value into a **IMValue** object.*

265 **IMChar im_value_get_char** (const **IMValue** *value)

*Get the char value holded in a **IMValue** object.*

266 void **im_value_set_uchar** (**IMValue** *value, **IMUChar** v_uchar)

*Store a unsigned char value into a **IMValue** object.*

267 **IMUChar im_value_get_uchar** (const **IMValue** *value)

*Get the unsigned char value holded in a **IMValue** object.*

268 void **im_value_set_bool** (**IMValue** *value, **IMBool** v_bool)

*Store a bool value into a **IMValue** object.*

269 **IMBool im_value_get_bool** (const **IMValue** *value)

*Get the bool value holded in a **IMValue** object.*

270 void **im_value_set_int16** (**IMValue** *value, **IMInt16** v_int32)

*Store an int16 value into a **IMValue** object.*

271 **IMInt16 im_value_get_int16** (const **IMValue** *value)

*Get the int16 value holded in a **IMValue** object.*

272 void **im_value_set_uint16** (**IMValue** *value, **IMUInt16** v_uint32)

*Store a unsigned int16 value into a **IMValue** object.*

273 **IMUInt16 im_value_get_uint16** (const **IMValue** *value)

*Get the unsigned int16 value holded in a **IMValue** object.*

274 void **im_value_set_int32** (**IMValue** *value, **IMInt32** v_int32)

*Store an int32 value into a **IMValue** object.*

275 **IMInt32 im_value_get_int32** (const **IMValue** *value)

*Get the int32 value holded in a **IMValue** object.*

276 void **im_value_set_uint32** (**IMValue** *value, **IMUInt32** v_uint32)

*Store a unsigned int32 value into a **IMValue** object.*

277 **IMUInt32 im_value_get_uint32** (const **IMValue** *value)

*Get the unsigned int32 value holded in a **IMValue** object.*

278 void **im_value_set_int64** (**IMValue** *value, **IMInt64** v_int64)

*Store an int64 value into a **IMValue** object.*

279 **IMInt64 im_value_get_int64** (const **IMValue** *value)

*Get the int64 value holded in a **IMValue** object.*

280 void **im_value_set_uint64** (**IMValue** *value, **IMUInt64** v_uint64)

*Store a unsigned int64 value into a **IMValue** object.*

281 **IMUInt64 im_value_get_uint64** (const **IMValue** *value)

*Get the unsigned int64 value holded in a **IMValue** object.*

282 void **im_value_set_double** (**IMValue** *value, **IMDouble** v_double)

*Store a double value into a **IMValue** object.*

283 **IMDouble im_value_get_double** (const **IMValue** *value)

*Get the double value holded in a **IMValue** object.*

284 void **im_value_set_pointer** (**IMValue** *value, **IMPointer** v_pointer)

*Store a pointer into a **IMValue** object.*

285 **IMPointer im_value_get_pointer** (const **IMValue** *value)

*Get the pointer value that is held in a **IMValue** object.*

286 void **im_value_set_c_string** (**IMValue** *value, const **IMChar** *v_string)

*Store a string into a **IMValue** object.*

287 void **im_value_set_static_c_string** (**IMValue** *value, const **IMChar** *v_string)

*Store a static string into a **IMValue** object.*

288 void **im_value_take_c_string** (**IMValue** *value, **IMChar** *v_string)

Take an ownership holded in a newly allocated string.

289 const **IMChar** * **im_value_get_c_string** (const **IMValue** *value)

*Get the string held by an **IMValue** object.*

290 **IMChar** * **im_value_dup_c_string** (const **IMValue** *value)

*im_value_dup_c_string - duplicate a string in an **IMValue** object*

291 void **im_value_set_object** (**IMValue** *value, **IMPointer** v_object)

*Store an object into a **IMValue** object.*

292 **IMPointer im_value_get_object** (const **IMValue** *value)

*Get the object holded b a **IMValue** object.*

293 **IMPointer im_value_dup_object** (const **IMValue** *value)

Duplicate the object holded by a value.

Define Documentation

#define IM_CONST_VALUE(p) (IM_TYPE_INSTANCE_CHECK_CAST_CONST((p),IM_TYPE_VALUE,IMValue))

#define IM_IS_VALUE(p) (IM_TYPE_INSTANCE_CHECK_TYPE((p),IM_TYPE_VALUE))

#define IM_IS_VALUE_CLASS(c) (IM_TYPE_CLASS_CHECK_TYPE((c),IM_TYPE_VALUE))

#define IM_VALUE(p) (IM_TYPE_INSTANCE_CHECK_CAST((p),IM_TYPE_VALUE,IMValue))

#define IM_VALUE_CLASS(c) (IM_TYPE_CLASS_CHECK_CAST((c),IM_TYPE_VALUE,IMValueClass))

#define IM_VALUE_GET_CLASS(p) (IM_TYPE_INSTANCE_GET_CLASS_CAST((p),IM_TYPE_VALUE,IMValueClass))

#define IM_VALUE_TYPE(p) (im_value_get_value_type (p))

#define IM_VALUE_TYPE_NAME(p) (im_type_get_name(IM_VALUE_TYPE(p)))

Typedef Documentation

typedef struct _IMValue IMValue

typedef struct _IMValueClass IMValueClass

Function Documentation

void im_value_clear (IMValue * *value*)

Clear the value of a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

void.

IMChar* im_value_dup_c_string (const IMValue * *value*)

im_value_dup_c_string - duplicate a string in an **IMValue** object

The **im_value_dup_c_string()** function shall return a pointer to a new string, which is a duplicate of the string contained in the **IMValue** object pointed by value. The returned pointer can be passed to **im_free()**. A null pointer is returned if the new string cannot be created or the datum contained in the **IMValue** object is not a string.

Parameters:

value a pointer to an **IMValue** object.

Returns:

The **im_value_dup_c_string()** function shall return a pointer to a new string in type of IMChar* on success. Otherwise it shall return a null pointer.

IMPointer im_value_dup_object (const IMValue * *value*)

Duplicate the object holded by a value.

Parameters:

value* a **IMValue object.

Returns:

IMPointer a pointer to a cloned object. Must be released by **im_object_unref()**;

IMBool im_value_get_bool (const IMValue * *value*)

Get the bool value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMBool a bool value.

const IMChar* im_value_get_c_string (const IMValue * *value*)

Get the string held by an **IMValue** object.

The string pointer holded by the **IMValue** object will be returned, it must not be freed or modified outside the **IMValue** object.

Parameters:

value* an **IMValue object.

Returns:

IMChar* The string pointer held by the **IMValue** object. If the **IMValue** object specified by the value parameter is not a type of string, the return value is NULL.

IMChar im_value_get_char (const IMValue * *value*)

Get the char value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMChar a char value.

IMDouble im_value_get_double (const IMValue * *value*)

Get the double value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMDouble a double value.

IMInt16 im_value_get_int16 (const IMValue * *value*)

Get the int16 value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMUInt16 a unsigned int32 value.

IMInt32 im_value_get_int32 (const IMValue * *value*)

Get the int32 value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMInt32 an int32 value.

IMInt64 im_value_get_int64 (const IMValue * *value*)

Get the int64 value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMInt64 an int64 value.

IMPointer im_value_get_object (const IMValue * *value*)

Get the object holded b a **IMValue** object.

Pointer to the object will be returned. If this object will continue to be used afer destroying the **IMValue** object, **im_object_ref()** should be called against it to increase its reference count.

Parameters:

value* a **IMValue object.

Returns:

IMPointer a Pointer to object.

IMPointer im_value_get_pointer (const IMValue * *value*)

Get the pointer value that is held in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMPointer a Pointer.

IMUChar im_value_get_uchar (const IMValue * *value*)

Get the unsigned char value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMUChar an unsigned char value.

IMUInt16 im_value_get_uint16 (const IMValue * *value*)

Get the unsigned int16 value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMUInt16 a unsigned int32 value.

IMUInt32 im_value_get_uint32 (const IMValue * *value*)

Get the unsigned int32 value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMUInt32 a unsigned int32 value.

IMUInt64 im_value_get_uint64 (const IMValue * *value*)

Get the unsigned int64 value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMUInt64 a unsigned int64 value.

IMType im_value_get_value_type (const IMValue * *value*)

Get the value type of a value holded in a **IMValue** object.

Parameters:

value* a **IMValue object.

Returns:

IMType a type name of the variant that has been filled into the **IMValue** object.

IMValue* im_value_new ()

Create a new zero-filled **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_bool (IMBool *v_bool*)

Create a new **IMValue** object that holds a bool value.

Parameters:

v_bool a bool value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_c_string (const IMChar * *v_string*)

Create a new **IMValue** object that holds a string.

Parameters:

v_string* a string to be filled into the new **IMValue object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_char (IMChar *v_char*)

Create a new **IMValue** object that holds a char value.

Parameters:

v_char a char value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_double (IMDouble *v_double*)

Create a new **IMValue** object that holds a double value.

Parameters:

v_double a double value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_int16 (IMInt16 *v_int16*)

Create a new **IMValue** object that holds an int16 value.

Parameters:

v_int16 an int16 value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_int32 (IMInt32 *v_int32*)

Create a new **IMValue** object that holds an int32 value.

Parameters:

v_int32 an int32 value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_int64 (IMInt64 *v_int64*)

Create a new **IMValue** object that holds an int64 value.

Parameters:

v_int64 an int64 value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_object (IMPointer *v_object*)

Create a new **IMValue** object that holds an object.

Parameters:

v_pointer a pointer to an object to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_pointer (IMPointer *v_pointer*)

Create a new **IMValue** object that holds a pointer.

Parameters:

v_pointer a pointer to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_static_c_string (const IMChar * *v_string*)

Create a new **IMValue** object that holds a static string.

Parameters:

v_string* a string to be filled into the new **IMValue object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_take_c_string (IMChar * *v_string*)

Create a new **IMValue** object that takes an ownership of a newly allocated string.

Parameters:

v_string* a string to be filled into the new **IMValue object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_uchar (IMUChar *v_uchar*)

Create a new **IMValue** object that holds a unsigned char value.

Parameters:

v_uchar an unsigned char value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_uint16 (IMUInt16 *v_uint16*)

Create a new **IMValue** object that holds a unsigned int16 value.

Parameters:

v_uint16 a unsigned int16 value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_uint32 (IMUInt32 *v_uint32*)

Create a new **IMValue** object that holds a unsigned int32 value.

Parameters:

v_uint32 a unsinged int32 value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

IMValue* im_value_new_uint64 (IMUInt64 *v_uint64*)

Create a new **IMValue** object that holds a unsigned int64 value.

Parameters:

v_uint64 a unsinged int64 value to be filled into the new **IMValue** object.

Returns:

IMValue* a new **IMValue** object.

void im_value_set_bool (IMValue * *value*, IMBool *v_bool*)

Store a bool value into a **IMValue** object.

Parameters:

value* a **IMValue object.

v_bool a bool value to be stored into the **IMValue** object.

Returns:

void.

void im_value_set_c_string (IMValue * *value*, const IMChar * *v_string*)

Store a string into a **IMValue** object.

The **IMValue** object will hold a copy of the string. So the original string can be freed or modified afterwards.

Parameters:

value* a **IMValue object.

v_string* a string to be stored into the **IMValue object.

Returns:

void

void im_value_set_char (IMValue * *value*, IMChar *v_char*)

Store a char value into a **IMValue** object.

Parameters:

value* a **IMValue object.

v_char a char value to be stored into the **IMValue** object.

Returns:
void.

void im_value_set_double (IMValue * *value*, IMDouble *v_double*)

Store a double value into a **IMValue** object.

Parameters:
value* a **IMValue object.
v_double a double value to be stored into the **IMValue** object.

Returns:
void.

void im_value_set_int16 (IMValue * *value*, IMInt16 *v_int32*)

Store an int16 value into a **IMValue** object.

Parameters:
value* a **IMValue object.
v_int32 an int16 value to be stored into the **IMValue** object.

Returns:
void.

void im_value_set_int32 (IMValue * *value*, IMInt32 *v_int32*)

Store an int32 value into a **IMValue** object.

Parameters:
value* a **IMValue object.
v_int32 an int32 value to be stored into the **IMValue** object.

Returns:
void.

void im_value_set_int64 (IMValue * *value*, IMInt64 *v_int64*)

Store an int64 value into a **IMValue** object.

Parameters:
value* a **IMValue object.
v_int64 an int64 value to be stored into the **IMValue** object.

Returns:
void.

void im_value_set_object (IMValue * *value*, IMPointer *v_object*)

Store an object into a **IMValue** object.

The reference count of the object will be increased, so that it won't be destroyed before destroying the **IMValue** object.

Parameters:
value* a **IMValue object.
v_object an object to stored.

Returns:
void.

void im_value_set_pointer (IMValue * *value*, IMPointer *v_pointer*)

Store a pointer into a **IMValue** object.

Parameters:

value* a **IMValue object.
v_pointer a pointer to be stored into the **IMValue** object.

Returns:

void.

void im_value_set_static_c_string (IMValue * *value*, const IMChar * *v_string*)

Store a static string into a **IMValue** object.

The **IMValue** object will just hold the original pointer, and won't free it when destroying.

So the string must not be freed or modified before destroying the value object.

Parameters:

value* a **IMValue object.
v_string* a string to be stored into the **IMValue object.

Returns:

void

void im_value_set_uchar (IMValue * *value*, IMUChar *v_uchar*)

Store a unsigned char value into a **IMValue** object.

Parameters:

value* a **IMValue object.
v_uchar a unsigned char value to be stored into the **IMValue** object.

Returns:

void.

void im_value_set_uint16 (IMValue * *value*, IMUInt16 *v_uint32*)

Store a unsigned int16 value into a **IMValue** object.

Parameters:

value* a **IMValue object.
v_uint32 a unsigned int16 value to be stored into the **IMValue** object.

Returns:

void.

void im_value_set_uint32 (IMValue * *value*, IMUInt32 *v_uint32*)

Store a unsigned int32 value into a **IMValue** object.

Parameters:

value* a **IMValue object.
v_uint32 a unsigned int32 value to be stored into the **IMValue** object.

Returns:

void.

void im_value_set_uint64 (IMValue * *value*, IMUInt64 *v_uint64*)

Store a unsigned int64 value into a **IMValue** object.

Parameters:

value* a **IMValue object.
v_uint64 a unsigned int64 value to be stored into the **IMValue** object.

Returns:

void.

void im_value_take_c_string (IMValue * *value*, IMChar * *v_string*)

Take an ownership holded in a newly allocated string.

The string will be freed when destroying the **IMValue** object.

Parameters:

value* a **IMValue object.
v_string* a string to be stored into the **IMValue object.

Returns:

void

6.8. Resource Management Service

6.8.1 Setting IMConfigurationStorage Attribute

im_configuration_storage_set_attr - setting IMConfigurationStorage attribute

The im_configuration_storage_set_attr() function set IMConfigurationStorage's attribute associated with IMConfigurationStorage attributes' identifier
described in this clause.

The im_configuration_storage_set_attr() function shall return an boolean value which holds a result whether the attribute is set
successfully or not.

The *key* argument specifies the name of the IMConfigurationStorage attributes described in this clause.
The *value* argument specifies the value of the IMConfigurationStorage attribute associated with the key.

When IMConfigurationStorage already has the attribute specified by the *key* argument,
the im_configuration_storage_set_attr() shall override the previous assigned one with the value argument.

SYNOPSYS

IMBool im_configuration_storage_set_attr(IMConfiurationStorage *storage, const IMChar *key, IMValue *value)

ARGUMENTS

storage:
a pointer to an IMConfigurationStorage instance that the IMConfigurationStorage attribute is set to.
key:
a pointer to a string in type of IMChar* that indicates the name of IMConfigurationStorage attributes.
value:
a pointer to a IMValue instance which contains the corresponding data type specified by the key.

RETURNS

TRUE on success. FALSE otherwise

6.8.2 Getting IMConfigurationStorage Attribute

im_configuration_storage_get_attr - gettine IMConfigurationStorage attribute

The im_configuration_storage_get_attr() function gets the value of IMConfigurationStorage attribute associated with the name of IMConfigurationStorage attributes.

The *key* specifies the name of the IMConfigurationStorage attributes described in this clause.
The *value* specifies an IMValue objects holds a result value associated with *key*.

When IMConfigurationStorage has not the value associated with *key, value* may be cleared.
On success, *value* IMValue instance holds Value of the attribute.

SYNOPSIS

IMBool im_configuration_storage_get_attr(IMConfigurationStorage *storage, const IMChar *key, IMValue *value)

Arguments

storage:
a pointer to an IMConfigurationStorage instance that the item is stored to.
key:
a poiter to a string in type of IMChar* that indicates the name of IMConfigurationStorage attributes.
value:
a pointer to a IMValue instance which is used to store the return value of IMConfigurationStorage attributes specified by the key.

Returns

TRUE on success, FALSE otherwise.

7. Calling convention of IM engine SPI

See SWG1_N075.

Annex A. List of keycodes(normative)

A keycode shall be a 32 bit integer which represents the status of modifier keys and character code for a key.

The format of keycode is :
Bit 31 : represents key press or release mask, this shall be 1 when key pressed
Bit 21 - 30: represents the status of modifier keys, ie, shift, caps lock, control, alt etc.
Bit 0 - 20 : represents character code as UCS-4 code.

The character code has same value with UCS-4 if the key represents unicode character code between 0 and 0x110000. For other keys which represent modifiers, functions keys, and other control keys, the character code has a value between 0x110000 and 0x11ffff.

A.1 Mask Values

IM_KEY_SHIFT_MASK 0x00200000
IM_KEY_CAPS_LOCK_MASK 0x00400000
IM_KEY_CTRL_MASK 0x00800000
IM_KEY_ALT_MASK 0x01000000
IM_KEY_META_MASK 0x02000000
IM_KEY_SUPER_MASK 0x04000000
IM_KEY_HYPER_MASK 0x08000000
IM_KEY_NUM_LOCK_MASK 0x10000000

A.2 Control keys

IM_KEY_topleftsummation 0x1108b1
IM_KEY_botleftsummation 0x1108b2
IM_KEY_topvertsummationconnector 0x1108b3
IM_KEY_botvertsummationconnector 0x1108b4
IM_KEY_toprightsummation 0x1108b5
IM_KEY_botrightsummation 0x1108b6
IM_KEY_rightmiddlesummation 0x1108b7
IM_KEY_marker 0x110abf
IM_KEY_trademarkincircle 0x110acb
IM_KEY_hexagram 0x110ada
IM_KEY_cursor 0x110aff
IM_KEY_Hangul_KkogjiDalrinIeung 0x110ef3
IM_KEY_Hangul_J_KkogjiDalrinIeung 0x110ef9
IM_KEY_3270_Duplicate 0x11fd01
IM_KEY_3270_FieldMark 0x11fd02
IM_KEY_3270_Right2 0x11fd03
IM_KEY_3270_Left2 0x11fd04
IM_KEY_3270_BackTab 0x11fd05
IM_KEY_3270_EraseEOF 0x11fd06
IM_KEY_3270_EraseInput 0x11fd07
IM_KEY_3270_Reset 0x11fd08

IM_KEY_3270_Quit 0x11fd09
IM_KEY_3270_PA1 0x11fd0a
IM_KEY_3270_PA2 0x11fd0b
IM_KEY_3270_PA3 0x11fd0c
IM_KEY_3270_Test 0x11fd0d
IM_KEY_3270_Attn 0x11fd0e
IM_KEY_3270_CursorBlink 0x11fd0f
IM_KEY_3270_AltCursor 0x11fd10
IM_KEY_3270_KeyClick 0x11fd11
IM_KEY_3270_Jump 0x11fd12
IM_KEY_3270_Ident 0x11fd13
IM_KEY_3270_Rule 0x11fd14
IM_KEY_3270_Copy 0x11fd15
IM_KEY_3270_Play 0x11fd16
IM_KEY_3270_Setup 0x11fd17
IM_KEY_3270_Record 0x11fd18
IM_KEY_3270_ChangeScreen 0x11fd19
IM_KEY_3270_DeleteWord 0x11fd1a
IM_KEY_3270_ExSelect 0x11fd1b
IM_KEY_3270_CursorSelect 0x11fd1c
IM_KEY_3270_PrintScreen 0x11fd1d
IM_KEY_3270_Enter 0x11fd1e
IM_KEY_ISO_Lock 0x11fe01
IM_KEY_ISO_Level2_Latch 0x11fe02
IM_KEY_ISO_Level3_Shift 0x11fe03
IM_KEY_ISO_Level3_Latch 0x11fe04
IM_KEY_ISO_Level3_Lock 0x11fe05
IM_KEY_ISO_Group_Latch 0x11fe06
IM_KEY_ISO_Group_Lock 0x11fe07
IM_KEY_ISO_Next_Group 0x11fe08
IM_KEY_ISO_Next_Group_Lock 0x11fe09
IM_KEY_ISO_Prev_Group 0x11fe0a
IM_KEY_ISO_Prev_Group_Lock 0x11fe0b
IM_KEY_ISO_First_Group 0x11fe0c
IM_KEY_ISO_First_Group_Lock 0x11fe0d
IM_KEY_ISO_Last_Group 0x11fe0e
IM_KEY_ISO_Last_Group_Lock 0x11fe0f
IM_KEY_ISO_Left_Tab 0x11fe20
IM_KEY_ISO_Move_Line_Up 0x11fe21
IM_KEY_ISO_Move_Line_Down 0x11fe22
IM_KEY_ISO_Partial_Line_Up 0x11fe23
IM_KEY_ISO_Partial_Line_Down 0x11fe24
IM_KEY_ISO_Partial_Space_Left 0x11fe25
IM_KEY_ISO_Partial_Space_Right 0x11fe26
IM_KEY_ISO_Set_Margin_Left 0x11fe27

IM_KEY_ISO_Set_Margin_Right 0x11fe28
IM_KEY_ISO_Release_Margin_Left 0x11fe29
IM_KEY_ISO_Release_Margin_Right 0x11fe2a
IM_KEY_ISO_Release_Both_Margins 0x11fe2b
IM_KEY_ISO_Fast_Cursor_Left 0x11fe2c
IM_KEY_ISO_Fast_Cursor_Right 0x11fe2d
IM_KEY_ISO_Fast_Cursor_Up 0x11fe2e
IM_KEY_ISO_Fast_Cursor_Down 0x11fe2f
IM_KEY_ISO_Continuous_Underline 0x11fe30
IM_KEY_ISO_Discontinuous_Underline 0x11fe31
IM_KEY_ISO_Emphasize 0x11fe32
IM_KEY_ISO_Center_Object 0x11fe33
IM_KEY_ISO_Enter 0x11fe34
IM_KEY_dead_grave 0x11fe50
IM_KEY_dead_acute 0x11fe51
IM_KEY_dead_circumflex 0x11fe52
IM_KEY_dead_tilde 0x11fe53
IM_KEY_dead_macron 0x11fe54
IM_KEY_dead_breve 0x11fe55
IM_KEY_dead_abovedot 0x11fe56
IM_KEY_dead_diaeresis 0x11fe57
IM_KEY_dead_abovering 0x11fe58
IM_KEY_dead_doubleacute 0x11fe59
IM_KEY_dead_caron 0x11fe5a
IM_KEY_dead_cedilla 0x11fe5b
IM_KEY_dead_ogonek 0x11fe5c
IM_KEY_dead_iota 0x11fe5d
IM_KEY_dead_voiced_sound 0x11fe5e
IM_KEY_dead_semivoiced_sound 0x11fe5f
IM_KEY_dead_belowdot 0x11fe60
IM_KEY_dead_hook 0x11fe61
IM_KEY_dead_horn 0x11fe62
IM_KEY_AccessX_Enable 0x11fe70
IM_KEY_AccessX_Feedback_Enable 0x11fe71
IM_KEY_RepeatKeys_Enable 0x11fe72
IM_KEY_SlowKeys_Enable 0x11fe73
IM_KEY_BounceKeys_Enable 0x11fe74
IM_KEY_StickyKeys_Enable 0x11fe75
IM_KEY_MouseKeys_Enable 0x11fe76
IM_KEY_MouseKeys_Accel_Enable 0x11fe77
IM_KEY_Overlay1_Enable 0x11fe78
IM_KEY_Overlay2_Enable 0x11fe79
IM_KEY_AudibleBell_Enable 0x11fe7a
IM_KEY_First_Virtual_Screen 0x11fed0
IM_KEY_Prev_Virtual_Screen 0x11fed1
IM_KEY_Next_Virtual_Screen 0x11fed2

IM_KEY_Last_Virtual_Screen 0x11fed4
IM_KEY_Terminate_Server 0x11fed5
IM_KEY_Pointer_Left 0x11fee0
IM_KEY_Pointer_Right 0x11fee1
IM_KEY_Pointer_Up 0x11fee2
IM_KEY_Pointer_Down 0x11fee3
IM_KEY_Pointer_UpLeft 0x11fee4
IM_KEY_Pointer_UpRight 0x11fee5
IM_KEY_Pointer_DownLeft 0x11fee6
IM_KEY_Pointer_DownRight 0x11fee7
IM_KEY_Pointer_Button_Dflt 0x11fee8
IM_KEY_Pointer_Button1 0x11fee9
IM_KEY_Pointer_Button2 0x11feea
IM_KEY_Pointer_Button3 0x11feeb
IM_KEY_Pointer_Button4 0x11feec
IM_KEY_Pointer_Button5 0x11feed
IM_KEY_Pointer_DblClick_Dflt 0x11feee
IM_KEY_Pointer_DblClick1 0x11feef
IM_KEY_Pointer_DblClick2 0x11fef0
IM_KEY_Pointer_DblClick3 0x11fef1
IM_KEY_Pointer_DblClick4 0x11fef2
IM_KEY_Pointer_DblClick5 0x11fef3
IM_KEY_Pointer_Drag_Dflt 0x11fef4
IM_KEY_Pointer_Drag1 0x11fef5
IM_KEY_Pointer_Drag2 0x11fef6
IM_KEY_Pointer_Drag3 0x11fef7
IM_KEY_Pointer_Drag4 0x11fef8
IM_KEY_Pointer_EnableKeys 0x11fef9
IM_KEY_Pointer_Accelerate 0x11fefa
IM_KEY_Pointer_DfltBtnNext 0x11fefb
IM_KEY_Pointer_DfltBtnPrev 0x11fefc
IM_KEY_Pointer_Drag5 0x11fefd
IM_KEY_Clear 0x11ff0b
IM_KEY_Pause 0x11ff13
IM_KEY_Scroll_Lock 0x11ff14
IM_KEY_Sys_Req 0x11ff15
IM_KEY_Escape 0x11ff1b
IM_KEY_Multi_key 0x11ff20
IM_KEY_Kanji 0x11ff21
IM_KEY_Muhenkan 0x11ff22
IM_KEY_Henkan_Mode 0x11ff23
IM_KEY_Romaji 0x11ff24
IM_KEY_Hiragana 0x11ff25
IM_KEY_Katakana 0x11ff26
IM_KEY_Hiragana_Katakana 0x11ff27

IM_KEY_Zenkaku 0x11ff28
IM_KEY_Hankaku 0x11ff29
IM_KEY_Zenkaku_Hankaku 0x11ff2a
IM_KEY_Touroku 0x11ff2b
IM_KEY_Massyo 0x11ff2c
IM_KEY_Kana_Lock 0x11ff2d
IM_KEY_Kana_Shift 0x11ff2e
IM_KEY_Eisu_Shift 0x11ff2f
IM_KEY_Eisu_toggle 0x11ff30
IM_KEY_Hangul 0x11ff31
IM_KEY_Hangul_Start 0x11ff32
IM_KEY_Hangul_End 0x11ff33
IM_KEY_Hangul_Hanja 0x11ff34
IM_KEY_Hangul_Jamo 0x11ff35
IM_KEY_Hangul_Romaja 0x11ff36
IM_KEY_Codeinput 0x11ff37
IM_KEY_Hangul_Jeonja 0x11ff38
IM_KEY_Hangul_Banja 0x11ff39
IM_KEY_Hangul_PreHanja 0x11ff3a
IM_KEY_Hangul_PostHanja 0x11ff3b
IM_KEY_SingleCandidate 0x11ff3c
IM_KEY_MultipleCandidate 0x11ff3d
IM_KEY_PreviousCandidate 0x11ff3e
IM_KEY_Hangul_Special 0x11ff3f
IM_KEY_Home 0x11ff50
IM_KEY_Left 0x11ff51
IM_KEY_Up 0x11ff52
IM_KEY_Right 0x11ff53
IM_KEY_Down 0x11ff54
IM_KEY_Page_Up 0x11ff55
IM_KEY_Page_Down 0x11ff56
IM_KEY_End 0x11ff57
IM_KEY_Begin 0x11ff58
IM_KEY_Select 0x11ff60
IM_KEY_Print 0x11ff61
IM_KEY_Execute 0x11ff62
IM_KEY_Insert 0x11ff63
IM_KEY_Undo 0x11ff65
IM_KEY_Redo 0x11ff66
IM_KEY_Menu 0x11ff67
IM_KEY_Find 0x11ff68
IM_KEY_Cancel 0x11ff69
IM_KEY_Help 0x11ff6a
IM_KEY_Break 0x11ff6b
IM_KEY_Mode_switch 0x11ff7e
IM_KEY_Num_Lock 0x11ff7f

IM_KEY_KP_Space 0x11ff80
IM_KEY_KP_Tab 0x11ff89
IM_KEY_KP_Enter 0x11ff8d
IM_KEY_KP_F1 0x11ff91
IM_KEY_KP_F2 0x11ff92
IM_KEY_KP_F3 0x11ff93
IM_KEY_KP_F4 0x11ff94
IM_KEY_KP_Home 0x11ff95
IM_KEY_KP_Left 0x11ff96
IM_KEY_KP_Up 0x11ff97
IM_KEY_KP_Right 0x11ff98
IM_KEY_KP_Down 0x11ff99
IM_KEY_KP_Page_Up 0x11ff9a
IM_KEY_KP_Page_Down 0x11ff9b
IM_KEY_KP_End 0x11ff9c
IM_KEY_KP_Begin 0x11ff9d
IM_KEY_KP_Insert 0x11ff9e
IM_KEY_KP_Delete 0x11ff9f
IM_KEY_KP_Multiply 0x11ffaa
IM_KEY_KP_Add 0x11ffab
IM_KEY_KP_Separator 0x11ffac
IM_KEY_KP_Subtract 0x11ffad
IM_KEY_KP_Decimal 0x11ffae
IM_KEY_KP_Divide 0x11ffaf
IM_KEY_KP_0 0x11ffb0
IM_KEY_KP_1 0x11ffb1
IM_KEY_KP_2 0x11ffb2
IM_KEY_KP_3 0x11ffb3
IM_KEY_KP_4 0x11ffb4
IM_KEY_KP_5 0x11ffb5
IM_KEY_KP_6 0x11ffb6
IM_KEY_KP_7 0x11ffb7
IM_KEY_KP_8 0x11ffb8
IM_KEY_KP_9 0x11ffb9
IM_KEY_KP_Equal 0x11ffbd
IM_KEY_F1 0x11ffbe
IM_KEY_F2 0x11ffbf
IM_KEY_F3 0x11ffc0
IM_KEY_F4 0x11ffc1
IM_KEY_F5 0x11ffc2
IM_KEY_F6 0x11ffc3
IM_KEY_F7 0x11ffc4
IM_KEY_F8 0x11ffc5
IM_KEY_F9 0x11ffc6
IM_KEY_F10 0x11ffc7

IM_KEY_F11	0x11ffc8
IM_KEY_F12	0x11ffc9
IM_KEY_F13	0x11ffca
IM_KEY_F14	0x11ffcb
IM_KEY_F15	0x11ffcc
IM_KEY_F16	0x11ffcd
IM_KEY_F17	0x11ffce
IM_KEY_F18	0x11ffcf
IM_KEY_F19	0x11ffd0
IM_KEY_F20	0x11ffd1
IM_KEY_F21	0x11ffd2
IM_KEY_F22	0x11ffd3
IM_KEY_F23	0x11ffd4
IM_KEY_F24	0x11ffd5
IM_KEY_F25	0x11ffd6
IM_KEY_F26	0x11ffd7
IM_KEY_F27	0x11ffd8
IM_KEY_F28	0x11ffd9
IM_KEY_F29	0x11ffda
IM_KEY_F30	0x11ffdb
IM_KEY_F31	0x11ffdc
IM_KEY_F32	0x11ffdd
IM_KEY_F33	0x11ffde
IM_KEY_F34	0x11ffdf
IM_KEY_F35	0x11ffe0
IM_KEY_Shift_L	0x11ffe1
IM_KEY_Shift_R	0x11ffe2
IM_KEY_Control_L	0x11ffe3
IM_KEY_Control_R	0x11ffe4
IM_KEY_Caps_Lock	0x11ffe5
IM_KEY_Shift_Lock	0x11ffe6
IM_KEY_Meta_L	0x11ffe7
IM_KEY_Meta_R	0x11ffe8
IM_KEY_Alt_L	0x11ffe9
IM_KEY_Alt_R	0x11ffea
IM_KEY_Super_L	0x11ffeb
IM_KEY_Super_R	0x11ffec
IM_KEY_Hyper_L	0x11ffed
IM_KEY_Hyper_R	0x11ffee
IM_KEY_Delete	0x11ffff

A.3 Character keys

IM_KEY_BackSpace	0x0008
IM_KEY_Tab	0x0009
IM_KEY_Linefeed	0x000a

IM_KEY_Return 0x000d
IM_KEY_space 0x0020
IM_KEY_exclam 0x0021 !
IM_KEY_quotedbl 0x0022 "
IM_KEY_numbersign 0x0023 #
IM_KEY_dollar 0x0024 $
IM_KEY_percent 0x0025 %
IM_KEY_ampersand 0x0026 &
IM_KEY_apostrophe 0x0027 '
IM_KEY_parenleft 0x0028 (
IM_KEY_parenright 0x0029)
IM_KEY_asterisk 0x002a *
IM_KEY_plus 0x002b +
IM_KEY_comma 0x002c ,
IM_KEY_minus 0x002d -
IM_KEY_period 0x002e .
IM_KEY_decimalpoint 0x002e .
IM_KEY_slash 0x002f /
IM_KEY_0 0x0030 0
IM_KEY_1 0x0031 1
IM_KEY_2 0x0032 2
IM_KEY_3 0x0033 3
IM_KEY_4 0x0034 4
IM_KEY_5 0x0035 5
IM_KEY_6 0x0036 6
IM_KEY_7 0x0037 7
IM_KEY_8 0x0038 8
IM_KEY_9 0x0039 9
IM_KEY_colon 0x003a :
IM_KEY_semicolon 0x003b ;
IM_KEY_leftcaret 0x003c <
IM_KEY_less 0x003c <
IM_KEY_equal 0x003d =
IM_KEY_rightcaret 0x003e >
IM_KEY_greater 0x003e >
IM_KEY_question 0x003f ?
IM_KEY_at 0x0040 @
IM_KEY_A 0x0041 A
IM_KEY_B 0x0042 B
IM_KEY_C 0x0043 C
IM_KEY_D 0x0044 D
IM_KEY_E 0x0045 E
IM_KEY_F 0x0046 F
IM_KEY_G 0x0047 G
IM_KEY_H 0x0048 H

IM_KEY_I	0x0049	I
IM_KEY_J	0x004a	J
IM_KEY_K	0x004b	K
IM_KEY_L	0x004c	L
IM_KEY_M	0x004d	M
IM_KEY_N	0x004e	N
IM_KEY_O	0x004f	O
IM_KEY_P	0x0050	P
IM_KEY_Q	0x0051	Q
IM_KEY_R	0x0052	R
IM_KEY_S	0x0053	S
IM_KEY_T	0x0054	T
IM_KEY_U	0x0055	U
IM_KEY_V	0x0056	V
IM_KEY_W	0x0057	W
IM_KEY_X	0x0058	X
IM_KEY_Y	0x0059	Y
IM_KEY_Z	0x005a	Z
IM_KEY_bracketleft	0x005b	[
IM_KEY_backslash	0x005c	\
IM_KEY_bracketright	0x005d	]
IM_KEY_asciicircum	0x005e	^
IM_KEY_underbar	0x005f	_
IM_KEY_underscore	0x005f	_
IM_KEY_grave	0x0060	`
IM_KEY_a	0x0061	a
IM_KEY_b	0x0062	b
IM_KEY_c	0x0063	c
IM_KEY_d	0x0064	d
IM_KEY_e	0x0065	e
IM_KEY_f	0x0066	f
IM_KEY_g	0x0067	g
IM_KEY_h	0x0068	h
IM_KEY_i	0x0069	i
IM_KEY_j	0x006a	j
IM_KEY_k	0x006b	k
IM_KEY_l	0x006c	l
IM_KEY_m	0x006d	m
IM_KEY_n	0x006e	n
IM_KEY_o	0x006f	o
IM_KEY_p	0x0070	p
IM_KEY_q	0x0071	q
IM_KEY_r	0x0072	r
IM_KEY_s	0x0073	s
IM_KEY_t	0x0074	t
IM_KEY_u	0x0075	u

IM_KEY_v 0x0076 v
IM_KEY_w 0x0077 w
IM_KEY_x 0x0078 x
IM_KEY_y 0x0079 y
IM_KEY_z 0x007a z
IM_KEY_braceleft 0x007b {
IM_KEY_bar 0x007c |
IM_KEY_braceright 0x007d }
IM_KEY_asciitilde 0x007e ~
IM_KEY_nobreakspace 0x00a0
IM_KEY_exclamdown 0x00a1 。
IM_KEY_cent 0x00a2 「
IM_KEY_sterling 0x00a3 」
IM_KEY_currency 0x00a4 、
IM_KEY_yen 0x00a5 ・
IM_KEY_brokenbar 0x00a6 ヲ
IM_KEY_section 0x00a7 ァ
IM_KEY_diaeresis 0x00a8 ィ
IM_KEY_copyright 0x00a9 ゥ
IM_KEY_ordfeminine 0x00aa ェ
IM_KEY_guillemotleft 0x00ab ォ
IM_KEY_notsign 0x00ac ャ
IM_KEY_hyphen 0x00ad
IM_KEY_registered 0x00ae ョ
IM_KEY_macron 0x00af ッ
IM_KEY_overbar 0x00af ッ
IM_KEY_degree 0x00b0 ー
IM_KEY_plusminus 0x00b1 ア
IM_KEY_twosuperior 0x00b2 イ
IM_KEY_threesuperior 0x00b3 ウ
IM_KEY_acute 0x00b4 エ
IM_KEY_mu 0x00b5 オ
IM_KEY_paragraph 0x00b6 カ
IM_KEY_periodcentered 0x00b7 キ
IM_KEY_cedilla 0x00b8 ク
IM_KEY_onesuperior 0x00b9 ケ
IM_KEY_masculine 0x00ba コ
IM_KEY_guillemotright 0x00bb サ
IM_KEY_onequarter 0x00bc シ
IM_KEY_onehalf 0x00bd ス
IM_KEY_threequarters 0x00be セ
IM_KEY_questiondown 0x00bf ソ
IM_KEY_Agrave 0x00c0 タ
IM_KEY_Aacute 0x00c1 チ
IM_KEY_Acircumflex 0x00c2 ツ

IM_KEY_Atilde	0x00c3	テ
IM_KEY_Adiaeresis	0x00c4	ト
IM_KEY_Aring	0x00c5	ナ
IM_KEY_AE	0x00c6	ニ
IM_KEY_Ccedilla	0x00c7	ヌ
IM_KEY_Egrave	0x00c8	ネ
IM_KEY_Eacute	0x00c9	ノ
IM_KEY_Ecircumflex	0x00ca	ハ
IM_KEY_Ediaeresis	0x00cb	ヒ
IM_KEY_Igrave	0x00cc	フ
IM_KEY_Iacute	0x00cd	ヘ
IM_KEY_Icircumflex	0x00ce	ホ
IM_KEY_Idiaeresis	0x00cf	マ
IM_KEY_ETH	0x00d0	ミ
IM_KEY_Ntilde	0x00d1	ム
IM_KEY_Ograve	0x00d2	メ
IM_KEY_Oacute	0x00d3	モ
IM_KEY_Ocircumflex	0x00d4	ヤ
IM_KEY_Otilde	0x00d5	ユ
IM_KEY_Odiaeresis	0x00d6	ヨ
IM_KEY_multiply	0x00d7	ラ
IM_KEY_Oslash	0x00d8	リ
IM_KEY_Ugrave	0x00d9	ル
IM_KEY_Uacute	0x00da	レ
IM_KEY_Ucircumflex	0x00db	ロ
IM_KEY_Udiaeresis	0x00dc	ワ
IM_KEY_Yacute	0x00dd	ン
IM_KEY_THORN	0x00de	゛
IM_KEY_ssharp	0x00df	゜
IM_KEY_agrave	0x00e0	à
IM_KEY_aacute	0x00e1	á
IM_KEY_acircumflex	0x00e2	â
IM_KEY_atilde	0x00e3	ã
IM_KEY_adiaeresis	0x00e4	ä
IM_KEY_aring	0x00e5	å
IM_KEY_ae	0x00e6	
IM_KEY_ccedilla	0x00e7	
IM_KEY_egrave	0x00e8	
IM_KEY_eacute	0x00e9	
IM_KEY_ecircumflex	0x00ea	
IM_KEY_ediaeresis	0x00eb	
IM_KEY_igrave	0x00ec	
IM_KEY_iacute	0x00ed	
IM_KEY_icircumflex	0x00ee	
IM_KEY_idiaeresis	0x00ef	
IM_KEY_eth	0x00f0	

IM_KEY_ntilde	0x00f1	ñ
IM_KEY_ograve	0x00f2	ò
IM_KEY_oacute	0x00f3	ó
IM_KEY_ocircumflex	0x00f4	ô
IM_KEY_otilde	0x00f5	õ
IM_KEY_odiaeresis	0x00f6	ö
IM_KEY_division	0x00f7	÷
IM_KEY_oslash	0x00f8	ø
IM_KEY_ugrave	0x00f9	ù
IM_KEY_uacute	0x00fa	ú
IM_KEY_ucircumflex	0x00fb	û
IM_KEY_udiaeresis	0x00fc	ü
IM_KEY_yacute	0x00fd	©
IM_KEY_thorn	0x00fe	™
IM_KEY_ydiaeresis	0x00ff	…
IM_KEY_Amacron	0x0100	Ā
IM_KEY_amacron	0x0101	ā
IM_KEY_Abreve	0x0102	Ă
IM_KEY_abreve	0x0103	ă
IM_KEY_Aogonek	0x0104	Ą
IM_KEY_aogonek	0x0105	ą
IM_KEY_Cacute	0x0106	Ć
IM_KEY_cacute	0x0107	ć
IM_KEY_Ccircumflex	0x0108	Ĉ
IM_KEY_ccircumflex	0x0109	ĉ
IM_KEY_Cabovedot	0x010a	Ċ
IM_KEY_cabovedot	0x010b	ċ
IM_KEY_Ccaron	0x010c	Č
IM_KEY_ccaron	0x010d	č
IM_KEY_Dcaron	0x010e	Ď
IM_KEY_dcaron	0x010f	ď
IM_KEY_Dstroke	0x0110	Đ
IM_KEY_dstroke	0x0111	đ
IM_KEY_Emacron	0x0112	Ē
IM_KEY_emacron	0x0113	ē
IM_KEY_Eabovedot	0x0116	Ė
IM_KEY_eabovedot	0x0117	ė
IM_KEY_Eogonek	0x0118	Ę
IM_KEY_eogonek	0x0119	ę
IM_KEY_Ecaron	0x011a	Ě
IM_KEY_ecaron	0x011b	ě
IM_KEY_Gcircumflex	0x011c	Ĝ
IM_KEY_gcircumflex	0x011d	ĝ
IM_KEY_Gbreve	0x011e	Ğ
IM_KEY_gbreve	0x011f	ğ

IM_KEY_Gabovedot 0x0120 Ġ
IM_KEY_gabovedot 0x0121 ġ
IM_KEY_Gcedilla 0x0122 Ģ
IM_KEY_gcedilla 0x0123 ģ
IM_KEY_Hcircumflex 0x0124 Ĥ
IM_KEY_hcircumflex 0x0125 ĥ
IM_KEY_Hstroke 0x0126 Ħ
IM_KEY_hstroke 0x0127 ħ
IM_KEY_Itilde 0x0128 Ĩ
IM_KEY_itilde 0x0129 ĩ
IM_KEY_Imacron 0x012a Ī
IM_KEY_imacron 0x012b ī
IM_KEY_Ibreve 0x012c Ĭ
IM_KEY_ibreve 0x012d ĭ
IM_KEY_Iogonek 0x012e Į
IM_KEY_iogonek 0x012f į
IM_KEY_Iabovedot 0x0130 İ
IM_KEY_idotless 0x0131 ı
IM_KEY_Jcircumflex 0x0134 Ĵ
IM_KEY_jcircumflex 0x0135 ĵ
IM_KEY_Kcedilla 0x0136 Ķ
IM_KEY_kcedilla 0x0137 ķ
IM_KEY_kra 0x0138 ĸ
IM_KEY_Lacute 0x0139 Ĺ
IM_KEY_lacute 0x013a ĺ
IM_KEY_Lcedilla 0x013b Ļ
IM_KEY_lcedilla 0x013c ļ
IM_KEY_Lcaron 0x013d Ľ
IM_KEY_lcaron 0x013e ľ
IM_KEY_Lstroke 0x0141 Ł
IM_KEY_lstroke 0x0142 ł
IM_KEY_Nacute 0x0143 Ń
IM_KEY_nacute 0x0144 ń
IM_KEY_Ncedilla 0x0145 Ņ
IM_KEY_ncedilla 0x0146 ņ
IM_KEY_Ncaron 0x0147 Ň
IM_KEY_ncaron 0x0148 ň
IM_KEY_ENG 0x014a Ŋ
IM_KEY_eng 0x014b ŋ
IM_KEY_Omacron 0x014c Ō
IM_KEY_omacron 0x014d ō
IM_KEY_Odoubleacute 0x0150 Ő
IM_KEY_odoubleacute 0x0151 ő
IM_KEY_OE 0x0152
IM_KEY_oe 0x0153
IM_KEY_Racute 0x0154 Ŕ

IM_KEY_racute 0x0155 ŕ
IM_KEY_Rcedilla 0x0156 Ŗ
IM_KEY_rcedilla 0x0157 ŗ
IM_KEY_Rcaron 0x0158 Ř
IM_KEY_rcaron 0x0159 ř
IM_KEY_Sacute 0x015a Ś
IM_KEY_sacute 0x015b ś
IM_KEY_Scircumflex 0x015c Ŝ
IM_KEY_scircumflex 0x015d ŝ
IM_KEY_Scedilla 0x015e Ş
IM_KEY_scedilla 0x015f ş
IM_KEY_Scaron 0x0160
IM_KEY_scaron 0x0161
IM_KEY_Tcedilla 0x0162 Ţ
IM_KEY_tcedilla 0x0163 ţ
IM_KEY_Tcaron 0x0164 Ť
IM_KEY_tcaron 0x0165 ť
IM_KEY_Tslash 0x0166 Ŧ
IM_KEY_tslash 0x0167 ŧ
IM_KEY_Utilde 0x0168 Ũ
IM_KEY_utilde 0x0169 ũ
IM_KEY_Umacron 0x016a Ū
IM_KEY_umacron 0x016b ū
IM_KEY_Ubreve 0x016c Ŭ
IM_KEY_ubreve 0x016d ŭ
IM_KEY_Uring 0x016e Ů
IM_KEY_uring 0x016f ů
IM_KEY_Udoubleacute 0x0170 Ű
IM_KEY_udoubleacute 0x0171 ű
IM_KEY_Uogonek 0x0172 Ų
IM_KEY_uogonek 0x0173 ų
IM_KEY_Wcircumflex 0x0174 Ŵ
IM_KEY_wcircumflex 0x0175 ŵ
IM_KEY_Ycircumflex 0x0176 Ŷ
IM_KEY_ycircumflex 0x0177 ŷ
IM_KEY_Ydiaeresis 0x0178
IM_KEY_Zacute 0x0179 Ź
IM_KEY_zacute 0x017a ź
IM_KEY_Zabovedot 0x017b Ż
IM_KEY_zabovedot 0x017c ż
IM_KEY_Zcaron 0x017d
IM_KEY_zcaron 0x017e
IM_KEY_SCHWA 0x018f Ə
IM_KEY_function 0x0192
IM_KEY_Obarred 0x019f Ɵ

IM_KEY_Ohorn 0x01a0 Ơ
IM_KEY_ohorn 0x01a1 ơ
IM_KEY_Uhorn 0x01af Ư
IM_KEY_uhorn 0x01b0 ư
IM_KEY_Zstroke 0x01b5 Ƶ
IM_KEY_zstroke 0x01b6 ƶ
IM_KEY_Ocaron 0x01d2 ǒ
IM_KEY_ocaron 0x01d2 ǒ
IM_KEY_Gcaron 0x01e6 Ǧ
IM_KEY_gcaron 0x01e7 ǧ
IM_KEY_schwa 0x0259 ə
IM_KEY_obarred 0x0275 ɵ
IM_KEY_caron 0x02c7 ˇ
IM_KEY_breve 0x02d8 ˘
IM_KEY_abovedot 0x02d9 ˙
IM_KEY_ogonek 0x02db ˛
IM_KEY_doubleacute 0x02dd ˝
IM_KEY_Greek_accentdieresis 0x0385 ΅
IM_KEY_Greek_ALPHAaccent 0x0386 Ά
IM_KEY_Greek_EPSILONaccent 0x0388 Έ
IM_KEY_Greek_ETAaccent 0x0389 Ή
IM_KEY_Greek_IOTAaccent 0x038a Ί
IM_KEY_Greek_OMICRONaccent 0x038c Ό
IM_KEY_Greek_UPSILONaccent 0x038e ΄Y
IM_KEY_Greek_OMEGAaccent 0x038f Ώ
IM_KEY_Greek_iotaaccentdieresis 0x0390 ΐ
IM_KEY_Greek_ALPHA 0x0391 A
IM_KEY_Greek_BETA 0x0392 B
IM_KEY_Greek_GAMMA 0x0393 Γ
IM_KEY_Greek_DELTA 0x0394 Δ
IM_KEY_Greek_EPSILON 0x0395 E
IM_KEY_Greek_ZETA 0x0396 Z
IM_KEY_Greek_ETA 0x0397 H
IM_KEY_Greek_THETA 0x0398 Θ
IM_KEY_Greek_IOTA 0x0399 I
IM_KEY_Greek_KAPPA 0x039a K
IM_KEY_Greek_LAMDA 0x039b Λ
IM_KEY_Greek_MU 0x039c M
IM_KEY_Greek_NU 0x039d N
IM_KEY_Greek_XI 0x039e Ξ
IM_KEY_Greek_OMICRON 0x039f O
IM_KEY_Greek_PI 0x03a0 Π
IM_KEY_Greek_RHO 0x03a1 P
IM_KEY_Greek_SIGMA 0x03a3 Σ
IM_KEY_Greek_TAU 0x03a4 T
IM_KEY_Greek_UPSILON 0x03a5 Y

IM_KEY_Greek_PHI 0x03a6 Φ
IM_KEY_Greek_CHI 0x03a7 Χ
IM_KEY_Greek_PSI 0x03a8 Ψ
IM_KEY_Greek_OMEGA 0x03a9 Ω
IM_KEY_Greek_IOTAdieresis 0x03aa Ϊ
IM_KEY_Greek_UPSILONdieresis 0x03ab Ϋ
IM_KEY_Greek_alphaaccent 0x03ac ά
IM_KEY_Greek_epsilonaccent 0x03ad έ
IM_KEY_Greek_etaaccent 0x03ae ή
IM_KEY_Greek_iotaaccent 0x03af ί
IM_KEY_Greek_upsilonaccentdieresis 0x03b0 ΰ
IM_KEY_Greek_alpha 0x03b1 α
IM_KEY_Greek_beta 0x03b2 β
IM_KEY_Greek_gamma 0x03b3 γ
IM_KEY_Greek_delta 0x03b4 δ
IM_KEY_Greek_epsilon 0x03b5 ε
IM_KEY_Greek_zeta 0x03b6 ζ
IM_KEY_Greek_eta 0x03b7 η
IM_KEY_Greek_theta 0x03b8 θ
IM_KEY_Greek_iota 0x03b9 ι
IM_KEY_Greek_kappa 0x03ba κ
IM_KEY_Greek_lamda 0x03bb λ
IM_KEY_Greek_mu 0x03bc μ
IM_KEY_Greek_nu 0x03bd ν
IM_KEY_Greek_xi 0x03be ξ
IM_KEY_Greek_omicron 0x03bf ο
IM_KEY_Greek_pi 0x03c0 π
IM_KEY_Greek_rho 0x03c1 ρ
IM_KEY_Greek_finalsmallsigma 0x03c2 ς
IM_KEY_Greek_sigma 0x03c3 σ
IM_KEY_Greek_tau 0x03c4 τ
IM_KEY_Greek_upsilon 0x03c5 υ
IM_KEY_Greek_phi 0x03c6 φ
IM_KEY_Greek_chi 0x03c7 χ
IM_KEY_Greek_psi 0x03c8 ψ
IM_KEY_Greek_omega 0x03c9 ω
IM_KEY_Greek_iotadieresis 0x03ca ϊ
IM_KEY_Greek_upsilondieresis 0x03cb ϋ
IM_KEY_Greek_omicronaccent 0x03cc ό
IM_KEY_Greek_upsilonaccent 0x03cd ύ
IM_KEY_Greek_omegaaccent 0x03ce ώ
IM_KEY_Cyrillic_IO 0x0401 Ё
IM_KEY_Serbian_DJE 0x0402 Ђ
IM_KEY_Macedonia_GJE 0x0403 Ѓ
IM_KEY_Ukrainian_IE 0x0404 Є

IM_KEY_Macedonia_DSE 0x0405 S
IM_KEY_Ukrainian_I 0x0406 І
IM_KEY_Ukrainian_YI 0x0407 Ї
IM_KEY_Cyrillic_JE 0x0408 Ј
IM_KEY_Cyrillic_LJE 0x0409 Љ
IM_KEY_Cyrillic_NJE 0x040a Њ
IM_KEY_Serbian_TSHE 0x040b Ћ
IM_KEY_Macedonia_KJE 0x040c Ќ
IM_KEY_Byelorussian_SHORTU 0x040e Ў
IM_KEY_Cyrillic_DZHE 0x040f Џ
IM_KEY_Cyrillic_A 0x0410 А
IM_KEY_Cyrillic_BE 0x0411 Б
IM_KEY_Cyrillic_VE 0x0412 В
IM_KEY_Cyrillic_GHE 0x0413 Г
IM_KEY_Cyrillic_DE 0x0414 Д
IM_KEY_Cyrillic_IE 0x0415 Е
IM_KEY_Cyrillic_ZHE 0x0416 Ж
IM_KEY_Cyrillic_ZE 0x0417 З
IM_KEY_Cyrillic_I 0x0418 И
IM_KEY_Cyrillic_SHORTI 0x0419 Й
IM_KEY_Cyrillic_KA 0x041a К
IM_KEY_Cyrillic_EL 0x041b Л
IM_KEY_Cyrillic_EM 0x041c М
IM_KEY_Cyrillic_EN 0x041d Н
IM_KEY_Cyrillic_O 0x041e О
IM_KEY_Cyrillic_PE 0x041f П
IM_KEY_Cyrillic_ER 0x0420 Р
IM_KEY_Cyrillic_ES 0x0421 С
IM_KEY_Cyrillic_TE 0x0422 Т
IM_KEY_Cyrillic_U 0x0423 У
IM_KEY_Cyrillic_EF 0x0424 Ф
IM_KEY_Cyrillic_HA 0x0425 Х
IM_KEY_Cyrillic_TSE 0x0426 Ц
IM_KEY_Cyrillic_CHE 0x0427 Ч
IM_KEY_Cyrillic_SHA 0x0428 Ш
IM_KEY_Cyrillic_SHCHA 0x0429 Щ
IM_KEY_Cyrillic_HARDSIGN 0x042a Ъ
IM_KEY_Cyrillic_YERU 0x042b Ы
IM_KEY_Cyrillic_SOFTSIGN 0x042c Ь
IM_KEY_Cyrillic_E 0x042d Э
IM_KEY_Cyrillic_YU 0x042e Ю
IM_KEY_Cyrillic_YA 0x042f Я
IM_KEY_Cyrillic_a 0x0430 а
IM_KEY_Cyrillic_be 0x0431 б
IM_KEY_Cyrillic_ve 0x0432 в
IM_KEY_Cyrillic_ghe 0x0433 г

IM_KEY_Cyrillic_de 0x0434 д
IM_KEY_Cyrillic_ie 0x0435 е
IM_KEY_Cyrillic_zhe 0x0436 ж
IM_KEY_Cyrillic_ze 0x0437 з
IM_KEY_Cyrillic_i 0x0438 и
IM_KEY_Cyrillic_shorti 0x0439 й
IM_KEY_Cyrillic_ka 0x043a к
IM_KEY_Cyrillic_el 0x043b л
IM_KEY_Cyrillic_em 0x043c м
IM_KEY_Cyrillic_en 0x043d н
IM_KEY_Cyrillic_o 0x043e о
IM_KEY_Cyrillic_pe 0x043f п
IM_KEY_Cyrillic_er 0x0440 р
IM_KEY_Cyrillic_es 0x0441 с
IM_KEY_Cyrillic_te 0x0442 т
IM_KEY_Cyrillic_u 0x0443 у
IM_KEY_Cyrillic_ef 0x0444 ф
IM_KEY_Cyrillic_ha 0x0445 х
IM_KEY_Cyrillic_tse 0x0446 ц
IM_KEY_Cyrillic_che 0x0447 ч
IM_KEY_Cyrillic_sha 0x0448 ш
IM_KEY_Cyrillic_shcha 0x0449 щ
IM_KEY_Cyrillic_hardsign 0x044a ъ
IM_KEY_Cyrillic_yeru 0x044b ы
IM_KEY_Cyrillic_softsign 0x044c ь
IM_KEY_Cyrillic_e 0x044d э
IM_KEY_Cyrillic_yu 0x044e ю
IM_KEY_Cyrillic_ya 0x044f я
IM_KEY_Cyrillic_io 0x0451 ё
IM_KEY_Serbian_dje 0x0452 ђ
IM_KEY_Macedonia_gje 0x0453 ѓ
IM_KEY_Ukrainian_ie 0x0454 є
IM_KEY_Macedonia_dse 0x0455 s
IM_KEY_Ukrainian_i 0x0456 і
IM_KEY_Ukrainian_yi 0x0457 ї
IM_KEY_Cyrillic_je 0x0458 j
IM_KEY_Cyrillic_lje 0x0459 љ
IM_KEY_Cyrillic_nje 0x045a њ
IM_KEY_Serbian_tshe 0x045b ћ
IM_KEY_Macedonia_kje 0x045c ќ
IM_KEY_Byelorussian_shortu 0x045e ў
IM_KEY_Cyrillic_dzhe 0x045f џ
IM_KEY_Ukrainian_GHE_WITH_UPTURN 0x0490 Ґ
IM_KEY_Ukrainian_ghe_with_upturn 0x0491 ґ
IM_KEY_Cyrillic_GHE_bar 0x0492 Ғ

IM_KEY_Cyrillic_ghe_bar 0x0493 ғ
IM_KEY_Cyrillic_ZHE_descender 0x0496 Җ
IM_KEY_Cyrillic_zhe_descender 0x0497 җ
IM_KEY_Cyrillic_KA_descender 0x049a Қ
IM_KEY_Cyrillic_ka_descender 0x049b қ
IM_KEY_Cyrillic_KA_vertstroke 0x049c Ҝ
IM_KEY_Cyrillic_ka_vertstroke 0x049d ҝ
IM_KEY_Cyrillic_EN_descender 0x04a2 Ң
IM_KEY_Cyrillic_en_descender 0x04a3 ң
IM_KEY_Cyrillic_U_straight 0x04ae Ү
IM_KEY_Cyrillic_u_straight 0x04af ү
IM_KEY_Cyrillic_U_straight_bar 0x04b0 Ұ
IM_KEY_Cyrillic_u_straight_bar 0x04b1 ұ
IM_KEY_Cyrillic_HA_descender 0x04b2 Ҳ
IM_KEY_Cyrillic_ha_descender 0x04b3 ҳ
IM_KEY_Cyrillic_CHE_descender 0x04b6 Ҷ
IM_KEY_Cyrillic_che_descender 0x04b7 ҷ
IM_KEY_Cyrillic_CHE_vertstroke 0x04b8 Ҹ
IM_KEY_Cyrillic_che_vertstroke 0x04b9 ҹ
IM_KEY_Cyrillic_SHHA 0x04ba Һ
IM_KEY_Cyrillic_shha 0x04bb һ
IM_KEY_Cyrillic_SCHWA 0x04d8 Ә
IM_KEY_Cyrillic_schwa 0x04d9 ә
IM_KEY_Cyrillic_I_macron 0x04e2 Ӣ
IM_KEY_Cyrillic_i_macron 0x04e3 ӣ
IM_KEY_Cyrillic_O_bar 0x04e8 Ө
IM_KEY_Cyrillic_o_bar 0x04e9 ө
IM_KEY_Cyrillic_U_macron 0x04ee Ӯ
IM_KEY_Cyrillic_u_macron 0x04ef ӯ
IM_KEY_Armenian_AYB 0x0531 Ա
IM_KEY_Armenian_BEN 0x0532 Բ
IM_KEY_Armenian_GIM 0x0533 Գ
IM_KEY_Armenian_DA 0x0534 Դ
IM_KEY_Armenian_YECH 0x0535 Ե
IM_KEY_Armenian_ZA 0x0536 Զ
IM_KEY_Armenian_E 0x0537 Է
IM_KEY_Armenian_AT 0x0538 Ը
IM_KEY_Armenian_TO 0x0539 Թ
IM_KEY_Armenian_ZHE 0x053a Ժ
IM_KEY_Armenian_INI 0x053b Ի
IM_KEY_Armenian_LYUN 0x053c Լ
IM_KEY_Armenian_KHE 0x053d Խ
IM_KEY_Armenian_TSA 0x053e Ծ
IM_KEY_Armenian_KEN 0x053f Կ
IM_KEY_Armenian_HO 0x0540 Հ
IM_KEY_Armenian_DZA 0x0541 Ձ

IM_KEY_Armenian_GHAT 0x0542 Ղ
IM_KEY_Armenian_TCHE 0x0543 Ճ
IM_KEY_Armenian_MEN 0x0544 Մ
IM_KEY_Armenian_HI 0x0545 Յ
IM_KEY_Armenian_NU 0x0546 Ն
IM_KEY_Armenian_SHA 0x0547 Շ
IM_KEY_Armenian_VO 0x0548 Ո
IM_KEY_Armenian_CHA 0x0549 Չ
IM_KEY_Armenian_PE 0x054a Պ
IM_KEY_Armenian_JE 0x054b Ջ
IM_KEY_Armenian_RA 0x054c Ռ
IM_KEY_Armenian_SE 0x054d Ս
IM_KEY_Armenian_VEV 0x054e Վ
IM_KEY_Armenian_TYUN 0x054f Տ
IM_KEY_Armenian_RE 0x0550 Ր
IM_KEY_Armenian_TSO 0x0551 Ց
IM_KEY_Armenian_VYUN 0x0552 Ւ
IM_KEY_Armenian_PYUR 0x0553 Փ
IM_KEY_Armenian_KE 0x0554 Ք
IM_KEY_Armenian_O 0x0555 Օ
IM_KEY_Armenian_FE 0x0556 Ֆ
IM_KEY_Armenian_apostrophe 0x055a ՚
IM_KEY_Armenian_accent 0x055b ՛
IM_KEY_Armenian_exclam 0x055c ՜
IM_KEY_Armenian_separation_mark 0x055d ՝
IM_KEY_Armenian_question 0x055e ՞
IM_KEY_Armenian_ayb 0x0561 ա
IM_KEY_Armenian_ben 0x0562 բ
IM_KEY_Armenian_gim 0x0563 գ
IM_KEY_Armenian_da 0x0564 դ
IM_KEY_Armenian_yech 0x0565 ե
IM_KEY_Armenian_za 0x0566 զ
IM_KEY_Armenian_e 0x0567 է
IM_KEY_Armenian_at 0x0568 ը
IM_KEY_Armenian_to 0x0569 թ
IM_KEY_Armenian_zhe 0x056a ժ
IM_KEY_Armenian_ini 0x056b ի
IM_KEY_Armenian_lyun 0x056c լ
IM_KEY_Armenian_khe 0x056d խ
IM_KEY_Armenian_tsa 0x056e ծ
IM_KEY_Armenian_ken 0x056f կ
IM_KEY_Armenian_ho 0x0570 հ
IM_KEY_Armenian_dza 0x0571 ձ
IM_KEY_Armenian_ghat 0x0572 ղ
IM_KEY_Armenian_tche 0x0573 ճ

IM_KEY_Armenian_men 0x0574 մ
IM_KEY_Armenian_hi 0x0575 յ
IM_KEY_Armenian_nu 0x0576 ն
IM_KEY_Armenian_sha 0x0577 շ
IM_KEY_Armenian_vo 0x0578 ո
IM_KEY_Armenian_cha 0x0579 չ
IM_KEY_Armenian_pe 0x057a պ
IM_KEY_Armenian_je 0x057b ջ
IM_KEY_Armenian_ra 0x057c ռ
IM_KEY_Armenian_se 0x057d ս
IM_KEY_Armenian_vev 0x057e վ
IM_KEY_Armenian_tyun 0x057f տ
IM_KEY_Armenian_re 0x0580 ր
IM_KEY_Armenian_tso 0x0581 ց
IM_KEY_Armenian_vyun 0x0582 ւ
IM_KEY_Armenian_pyur 0x0583 փ
IM_KEY_Armenian_ke 0x0584 ք
IM_KEY_Armenian_o 0x0585 օ
IM_KEY_Armenian_fe 0x0586 ֆ
IM_KEY_Armenian_ligature_ew 0x0587 և
IM_KEY_Armenian_full_stop 0x0589 ։
IM_KEY_Armenian_hyphen 0x058a ֊
IM_KEY_hebrew_aleph 0x05d0 א
IM_KEY_hebrew_bet 0x05d1 ב
IM_KEY_hebrew_gimel 0x05d2 ג
IM_KEY_hebrew_dalet 0x05d3 ד
IM_KEY_hebrew_he 0x05d4 ה
IM_KEY_hebrew_waw 0x05d5 ו
IM_KEY_hebrew_zain 0x05d6 ז
IM_KEY_hebrew_chet 0x05d7 ח
IM_KEY_hebrew_tet 0x05d8 ט
IM_KEY_hebrew_yod 0x05d9 י
IM_KEY_hebrew_finalkaph 0x05da ך
IM_KEY_hebrew_kaph 0x05db כ
IM_KEY_hebrew_lamed 0x05dc ל
IM_KEY_hebrew_finalmem 0x05dd ם
IM_KEY_hebrew_mem 0x05de מ
IM_KEY_hebrew_finalnun 0x05df ן
IM_KEY_hebrew_nun 0x05e0 נ
IM_KEY_hebrew_samech 0x05e1 ס
IM_KEY_hebrew_ayin 0x05e2 ע
IM_KEY_hebrew_finalpe 0x05e3 ף
IM_KEY_hebrew_pe 0x05e4 פ
IM_KEY_hebrew_finalzade 0x05e5 ץ
IM_KEY_hebrew_zade 0x05e6 צ
IM_KEY_hebrew_qoph 0x05e7 ק

IM_KEY_hebrew_resh 0x05e8 ר
IM_KEY_hebrew_shin 0x05e9 ש
IM_KEY_hebrew_taw 0x05ea ת
IM_KEY_Arabic_comma 0x060c ،
IM_KEY_Arabic_semicolon 0x061b ؛
IM_KEY_Arabic_question_mark 0x061f ؟
IM_KEY_Arabic_hamza 0x0621 ء
IM_KEY_Arabic_maddaonalef 0x0622 آ
IM_KEY_Arabic_hamzaonalef 0x0623 أ
IM_KEY_Arabic_hamzaonwaw 0x0624 ؤ
IM_KEY_Arabic_hamzaunderalef 0x0625 إ
IM_KEY_Arabic_hamzaonyeh 0x0626 ئ
IM_KEY_Arabic_alef 0x0627 ا
IM_KEY_Arabic_beh 0x0628 ب
IM_KEY_Arabic_tehmarbuta 0x0629 ة
IM_KEY_Arabic_teh 0x062a ت
IM_KEY_Arabic_theh 0x062b ث
IM_KEY_Arabic_jeem 0x062c ج
IM_KEY_Arabic_hah 0x062d ح
IM_KEY_Arabic_khah 0x062e خ
IM_KEY_Arabic_dal 0x062f د
IM_KEY_Arabic_thal 0x0630 ذ
IM_KEY_Arabic_ra 0x0631 ر
IM_KEY_Arabic_zain 0x0632 ز
IM_KEY_Arabic_seen 0x0633 س
IM_KEY_Arabic_sheen 0x0634 ش
IM_KEY_Arabic_sad 0x0635 ص
IM_KEY_Arabic_dad 0x0636 ض
IM_KEY_Arabic_tah 0x0637 ط
IM_KEY_Arabic_zah 0x0638 ظ
IM_KEY_Arabic_ain 0x0639 ع
IM_KEY_Arabic_ghain 0x063a غ
IM_KEY_Arabic_tatweel 0x0640 -
IM_KEY_Arabic_feh 0x0641 ف
IM_KEY_Arabic_qaf 0x0642 ق
IM_KEY_Arabic_kaf 0x0643 ك
IM_KEY_Arabic_lam 0x0644 ل
IM_KEY_Arabic_meem 0x0645 م
IM_KEY_Arabic_noon 0x0646 ن
IM_KEY_Arabic_ha 0x0647 ه
IM_KEY_Arabic_waw 0x0648 و
IM_KEY_Arabic_alefmaksura 0x0649 ى
IM_KEY_Arabic_yeh 0x064a ي
IM_KEY_Arabic_fathatan 0x064b ً
IM_KEY_Arabic_dammatan 0x064c ٌ

IM_KEY_Arabic_kasratan 0x064d ٍ
IM_KEY_Arabic_fatha 0x064e َ
IM_KEY_Arabic_damma 0x064f ُ
IM_KEY_Arabic_kasra 0x0650 ِ
IM_KEY_Arabic_shadda 0x0651 ّ
IM_KEY_Arabic_sukun 0x0652 ْ
IM_KEY_Arabic_madda_above 0x0653 ~
IM_KEY_Arabic_hamza_above 0x0654 ٔ
IM_KEY_Arabic_hamza_below 0x0655 ٕ
IM_KEY_Arabic_0 0x0660 ٠
IM_KEY_Arabic_1 0x0661 ١
IM_KEY_Arabic_2 0x0662 ٢
IM_KEY_Arabic_3 0x0663 ٣
IM_KEY_Arabic_4 0x0664 ٤
IM_KEY_Arabic_5 0x0665 ٥
IM_KEY_Arabic_6 0x0666 ٦
IM_KEY_Arabic_7 0x0667 ٧
IM_KEY_Arabic_8 0x0668 ٨
IM_KEY_Arabic_9 0x0669 ٩
IM_KEY_Arabic_percent 0x066a ٪
IM_KEY_Arabic_superscript_alef 0x0670 ٰ
IM_KEY_Arabic_tteh 0x0679 ٹ
IM_KEY_Arabic_peh 0x067e پ
IM_KEY_Arabic_tcheh 0x0686 چ
IM_KEY_Arabic_ddal 0x0688 ڈ
IM_KEY_Arabic_rreh 0x0691 ڑ
IM_KEY_Arabic_jeh 0x0698 ژ
IM_KEY_Arabic_veh 0x06a4 ڤ
IM_KEY_Arabic_keheh 0x06a9 ک
IM_KEY_Arabic_gaf 0x06af گ
IM_KEY_Arabic_noon_ghunna 0x06ba ں
IM_KEY_Arabic_heh_doachashmee 0x06be ھ
IM_KEY_Arabic_heh_goal 0x06c1 ہ
IM_KEY_Arabic_farsi_yeh 0x06cc ی
IM_KEY_Arabic_yeh_baree 0x06d2 ے
IM_KEY_Arabic_fullstop 0x06d4 ۔
IM_KEY_Farsi_0 0x06f0 ۰
IM_KEY_Farsi_1 0x06f1 ۱
IM_KEY_Farsi_2 0x06f2 ۲
IM_KEY_Farsi_3 0x06f3 ۳
IM_KEY_Farsi_4 0x06f4 ۴
IM_KEY_Farsi_5 0x06f5 ۵
IM_KEY_Farsi_6 0x06f6 ۶
IM_KEY_Farsi_7 0x06f7 ۷
IM_KEY_Farsi_8 0x06f8 ۸
IM_KEY_Farsi_9 0x06f9 ۹

IM_KEY_Thai_kokai 0x0e01 ก
IM_KEY_Thai_khokhai 0x0e02 ข
IM_KEY_Thai_khokhuat 0x0e03 ฃ
IM_KEY_Thai_khokhwai 0x0e04 ค
IM_KEY_Thai_khokhon 0x0e05 ฅ
IM_KEY_Thai_khorakhang 0x0e06 ฆ
IM_KEY_Thai_ngongu 0x0e07 ง
IM_KEY_Thai_chochan 0x0e08 จ
IM_KEY_Thai_choching 0x0e09 ฉ
IM_KEY_Thai_chochang 0x0e0a ช
IM_KEY_Thai_soso 0x0e0b ซ
IM_KEY_Thai_chochoe 0x0e0c ฌ
IM_KEY_Thai_yoying 0x0e0d ญ
IM_KEY_Thai_dochada 0x0e0e ฎ
IM_KEY_Thai_topatak 0x0e0f ฏ
IM_KEY_Thai_thothan 0x0e10 ฐ
IM_KEY_Thai_thonangmontho 0x0e11 ฑ
IM_KEY_Thai_thophuthao 0x0e12 ฒ
IM_KEY_Thai_nonen 0x0e13 ณ
IM_KEY_Thai_dodek 0x0e14 ด
IM_KEY_Thai_totao 0x0e15 ต
IM_KEY_Thai_thothung 0x0e16 ถ
IM_KEY_Thai_thothahan 0x0e17 ท
IM_KEY_Thai_thothong 0x0e18 ธ
IM_KEY_Thai_nonu 0x0e19 น
IM_KEY_Thai_bobaimai 0x0e1a บ
IM_KEY_Thai_popla 0x0e1b ป
IM_KEY_Thai_phophung 0x0e1c ผ
IM_KEY_Thai_fofa 0x0e1d ฝ
IM_KEY_Thai_phophan 0x0e1e พ
IM_KEY_Thai_fofan 0x0e1f ฟ
IM_KEY_Thai_phosamphao 0x0e20 ภ
IM_KEY_Thai_moma 0x0e21 ม
IM_KEY_Thai_yoyak 0x0e22 ย
IM_KEY_Thai_rorua 0x0e23 ร
IM_KEY_Thai_ru 0x0e24 ฤ
IM_KEY_Thai_loling 0x0e25 ล
IM_KEY_Thai_lu 0x0e26 ฦ
IM_KEY_Thai_wowaen 0x0e27 ว
IM_KEY_Thai_sosala 0x0e28 ศ
IM_KEY_Thai_sorusi 0x0e29 ษ
IM_KEY_Thai_sosua 0x0e2a ส
IM_KEY_Thai_hohip 0x0e2b ห
IM_KEY_Thai_lochula 0x0e2c ฬ
IM_KEY_Thai_oang 0x0e2d อ

IM_KEY_Thai_honokhuk 0x0e2e ฮ
IM_KEY_Thai_paiyannoi 0x0e2f ฯ
IM_KEY_Thai_saraa 0x0e30 ะ
IM_KEY_Thai_maihanakat 0x0e31 ั
IM_KEY_Thai_saraaa 0x0e32 า
IM_KEY_Thai_saraam 0x0e33 ำ
IM_KEY_Thai_sarai 0x0e34 ิ
IM_KEY_Thai_saraii 0x0e35 ี
IM_KEY_Thai_saraue 0x0e36 ึ
IM_KEY_Thai_sarauee 0x0e37 ื
IM_KEY_Thai_sarau 0x0e38 ุ
IM_KEY_Thai_sarauu 0x0e39 ู
IM_KEY_Thai_phinthu 0x0e3a ฺ
IM_KEY_Thai_maihanakat_maitho 0x0e3e
IM_KEY_Thai_baht 0x0e3f ฿
IM_KEY_Thai_sarae 0x0e40 เ
IM_KEY_Thai_saraae 0x0e41 แ
IM_KEY_Thai_sarao 0x0e42 โ
IM_KEY_Thai_saraaimaimuan 0x0e43 ใ
IM_KEY_Thai_saraaimaimalai 0x0e44 ไ
IM_KEY_Thai_lakkhangyao 0x0e45 ๅ
IM_KEY_Thai_maiyamok 0x0e46 ๆ
IM_KEY_Thai_maitaikhu 0x0e47 ็
IM_KEY_Thai_maiek 0x0e48 ่
IM_KEY_Thai_maitho 0x0e49 ้
IM_KEY_Thai_maitri 0x0e4a ๊
IM_KEY_Thai_maichattawa 0x0e4b ๋
IM_KEY_Thai_thanthakhat 0x0e4c ์
IM_KEY_Thai_nikhahit 0x0e4d ำ
IM_KEY_Thai_leksun 0x0e50 ๐
IM_KEY_Thai_leknung 0x0e51 ๑
IM_KEY_Thai_leksong 0x0e52 ๒
IM_KEY_Thai_leksam 0x0e53 ๓
IM_KEY_Thai_leksi 0x0e54 ๔
IM_KEY_Thai_lekha 0x0e55 ๕
IM_KEY_Thai_lekhok 0x0e56 ๖
IM_KEY_Thai_lekchet 0x0e57 ๗
IM_KEY_Thai_lekpaet 0x0e58 ๘
IM_KEY_Thai_lekkao 0x0e59 ๙
IM_KEY_Georgian_an 0x10d0 ა
IM_KEY_Georgian_ban 0x10d1 ბ
IM_KEY_Georgian_gan 0x10d2 გ
IM_KEY_Georgian_don 0x10d3 დ
IM_KEY_Georgian_en 0x10d4 ე
IM_KEY_Georgian_vin 0x10d5 ვ
IM_KEY_Georgian_zen 0x10d6 ზ

IM_KEY_Georgian_tan 0x10d7 თ
IM_KEY_Georgian_in 0x10d8 ი
IM_KEY_Georgian_kan 0x10d9 კ
IM_KEY_Georgian_las 0x10da ლ
IM_KEY_Georgian_man 0x10db მ
IM_KEY_Georgian_nar 0x10dc ნ
IM_KEY_Georgian_on 0x10dd ო
IM_KEY_Georgian_par 0x10de პ
IM_KEY_Georgian_zhar 0x10df ჟ
IM_KEY_Georgian_rae 0x10e0 რ
IM_KEY_Georgian_san 0x10e1 ს
IM_KEY_Georgian_tar 0x10e2 ტ
IM_KEY_Georgian_un 0x10e3 უ
IM_KEY_Georgian_phar 0x10e4 ფ
IM_KEY_Georgian_khar 0x10e5 ქ
IM_KEY_Georgian_ghan 0x10e6 ღ
IM_KEY_Georgian_qar 0x10e7 ყ
IM_KEY_Georgian_shin 0x10e8 შ
IM_KEY_Georgian_chin 0x10e9 ჩ
IM_KEY_Georgian_can 0x10ea ც
IM_KEY_Georgian_jil 0x10eb ძ
IM_KEY_Georgian_cil 0x10ec წ
IM_KEY_Georgian_char 0x10ed ჭ
IM_KEY_Georgian_xan 0x10ee ხ
IM_KEY_Georgian_jhan 0x10ef ჯ
IM_KEY_Georgian_hae 0x10f0 ჰ
IM_KEY_Georgian_he 0x10f1 ჱ
IM_KEY_Georgian_hie 0x10f2 ჲ
IM_KEY_Georgian_we 0x10f3 ჳ
IM_KEY_Georgian_har 0x10f4 ჴ
IM_KEY_Georgian_hoe 0x10f5 ჵ
IM_KEY_Georgian_fi 0x10f6
IM_KEY_Hangul_J_Kiyeog 0x11a8
IM_KEY_Hangul_J_SsangKiyeog 0x11a9
IM_KEY_Hangul_J_KiyeogSios 0x11aa
IM_KEY_Hangul_J_Nieun 0x11ab
IM_KEY_Hangul_J_NieunJieuj 0x11ac
IM_KEY_Hangul_J_NieunHieuh 0x11ad
IM_KEY_Hangul_J_Dikeud 0x11ae
IM_KEY_Hangul_J_Rieul 0x11af
IM_KEY_Hangul_J_RieulKiyeog 0x11b0
IM_KEY_Hangul_J_RieulMieum 0x11b1
IM_KEY_Hangul_J_RieulPieub 0x11b2
IM_KEY_Hangul_J_RieulSios 0x11b3
IM_KEY_Hangul_J_RieulTieut 0x11b4

IM_KEY_Hangul_J_RieulPhieuf 0x11b5
IM_KEY_Hangul_J_RieulHieuh 0x11b6
IM_KEY_Hangul_J_Mieum 0x11b7
IM_KEY_Hangul_J_Pieub 0x11b8
IM_KEY_Hangul_J_PieubSios 0x11b9
IM_KEY_Hangul_J_Sios 0x11ba
IM_KEY_Hangul_J_SsangSios 0x11bb
IM_KEY_Hangul_J_Ieung 0x11bc
IM_KEY_Hangul_J_Jieuj 0x11bd
IM_KEY_Hangul_J_Cieuc 0x11be
IM_KEY_Hangul_J_Khieuq 0x11bf
IM_KEY_Hangul_J_Tieut 0x11c0
IM_KEY_Hangul_J_Phieuf 0x11c1
IM_KEY_Hangul_J_Hieuh 0x11c2
IM_KEY_Hangul_J_PanSios 0x11eb
IM_KEY_Hangul_J_YeorinHieuh 0x11f9
IM_KEY_Babovedot 0x1e02 Ḃ
IM_KEY_babovedot 0x1e03 ḃ
IM_KEY_Dabovedot 0x1e0a Ḋ
IM_KEY_dabovedot 0x1e0b ḋ
IM_KEY_Fabovedot 0x1e1e Ḟ
IM_KEY_fabovedot 0x1e1f ḟ
IM_KEY_Lbelowdot 0x1e36 Ḷ
IM_KEY_lbelowdot 0x1e37 ḷ
IM_KEY_Mabovedot 0x1e40 Ṁ
IM_KEY_mabovedot 0x1e41 ṁ
IM_KEY_Pabovedot 0x1e56 Ṗ
IM_KEY_pabovedot 0x1e57 ṗ
IM_KEY_Sabovedot 0x1e60 Ṡ
IM_KEY_sabovedot 0x1e61 ṡ
IM_KEY_Tabovedot 0x1e6a Ṫ
IM_KEY_tabovedot 0x1e6b ṫ
IM_KEY_Wgrave 0x1e80 Ẁ
IM_KEY_wgrave 0x1e81 ẁ
IM_KEY_Wacute 0x1e82 Ẃ
IM_KEY_wacute 0x1e83 ẃ
IM_KEY_Wdiaeresis 0x1e84 Ẅ
IM_KEY_wdiaeresis 0x1e85 ẅ
IM_KEY_Xabovedot 0x1e8a Ẋ
IM_KEY_xabovedot 0x1e8b ẋ
IM_KEY_Abelowdot 0x1ea0 Ạ
IM_KEY_abelowdot 0x1ea1 ạ
IM_KEY_Ahook 0x1ea2 Ả
IM_KEY_ahook 0x1ea3 ả
IM_KEY_Acircumflexacute 0x1ea4 Ấ

IM_KEY_acircumflexacute 0x1ea5 ấ
IM_KEY_Acircumflexgrave 0x1ea6 Ầ
IM_KEY_acircumflexgrave 0x1ea7 ầ
IM_KEY_Acircumflexhook 0x1ea8 Ẩ
IM_KEY_acircumflexhook 0x1ea9 ẩ
IM_KEY_Acircumflextilde 0x1eaa Ẫ
IM_KEY_acircumflextilde 0x1eab ẫ
IM_KEY_Acircumflexbelowdot 0x1eac Ậ
IM_KEY_acircumflexbelowdot 0x1ead ậ
IM_KEY_Abreveacute 0x1eae Ắ
IM_KEY_abreveacute 0x1eaf ắ
IM_KEY_Abrevegrave 0x1eb0 Ằ
IM_KEY_abrevegrave 0x1eb1 ằ
IM_KEY_Abrevehook 0x1eb2 Ẳ
IM_KEY_abrevehook 0x1eb3 ẳ
IM_KEY_Abrevetilde 0x1eb4 Ẵ
IM_KEY_abrevetilde 0x1eb5 ẵ
IM_KEY_Abrevebelowdot 0x1eb6 Ặ
IM_KEY_abrevebelowdot 0x1eb7 ặ
IM_KEY_Ebelowdot 0x1eb8 Ẹ
IM_KEY_ebelowdot 0x1eb9 ẹ
IM_KEY_Ehook 0x1eba Ẻ
IM_KEY_ehook 0x1ebb ẻ
IM_KEY_Etilde 0x1ebc Ẽ
IM_KEY_etilde 0x1ebd ẽ
IM_KEY_Ecircumflexacute 0x1ebe Ế
IM_KEY_ecircumflexacute 0x1ebf ế
IM_KEY_Ecircumflexgrave 0x1ec0 Ề
IM_KEY_ecircumflexgrave 0x1ec1 ề
IM_KEY_Ecircumflexhook 0x1ec2 Ể
IM_KEY_ecircumflexhook 0x1ec3 ể
IM_KEY_Ecircumflextilde 0x1ec4 Ễ
IM_KEY_ecircumflextilde 0x1ec5 ễ
IM_KEY_Ecircumflexbelowdot 0x1ec6 Ệ
IM_KEY_ecircumflexbelowdot 0x1ec7 ệ
IM_KEY_Ihook 0x1ec8 Ỉ
IM_KEY_ihook 0x1ec9 ỉ
IM_KEY_Ibelowdot 0x1eca Ị
IM_KEY_ibelowdot 0x1ecb ị
IM_KEY_Obelowdot 0x1ecc Ọ
IM_KEY_obelowdot 0x1ecd ọ
IM_KEY_Ohook 0x1ece Ỏ
IM_KEY_ohook 0x1ecf ỏ
IM_KEY_Ocircumflexacute 0x1ed0 Ố
IM_KEY_ocircumflexacute 0x1ed1 ố

IM_KEY_Ocircumflexgrave 0x1ed2 Ồ
IM_KEY_ocircumflexgrave 0x1ed3 ồ
IM_KEY_Ocircumflexhook 0x1ed4 Ổ
IM_KEY_ocircumflexhook 0x1ed5 ổ
IM_KEY_Ocircumflextilde 0x1ed6 Ỗ
IM_KEY_ocircumflextilde 0x1ed7 ỗ
IM_KEY_Ocircumflexbelowdot 0x1ed8 Ộ
IM_KEY_ocircumflexbelowdot 0x1ed9 ộ
IM_KEY_Ohornacute 0x1eda Ớ
IM_KEY_ohornacute 0x1edb ớ
IM_KEY_Ohorngrave 0x1edc Ờ
IM_KEY_ohorngrave 0x1edd ờ
IM_KEY_Ohornhook 0x1ede Ở
IM_KEY_ohornhook 0x1edf ở
IM_KEY_Ohorntilde 0x1ee0 Ỡ
IM_KEY_ohorntilde 0x1ee1 ỡ
IM_KEY_Ohornbelowdot 0x1ee2 Ợ
IM_KEY_ohornbelowdot 0x1ee3 ợ
IM_KEY_Ubelowdot 0x1ee4 Ụ
IM_KEY_ubelowdot 0x1ee5 ụ
IM_KEY_Uhook 0x1ee6 Ủ
IM_KEY_uhook 0x1ee7 ủ
IM_KEY_Uhornacute 0x1ee8 Ứ
IM_KEY_uhornacute 0x1ee9 ứ
IM_KEY_Uhorngrave 0x1eea Ừ
IM_KEY_uhorngrave 0x1eeb ừ
IM_KEY_Uhornhook 0x1eec Ử
IM_KEY_uhornhook 0x1eed ử
IM_KEY_Uhorntilde 0x1eee Ữ
IM_KEY_uhorntilde 0x1eef ữ
IM_KEY_Uhornbelowdot 0x1ef0 Ự
IM_KEY_uhornbelowdot 0x1ef1 ự
IM_KEY_Ygrave 0x1ef2 Ỳ
IM_KEY_ygrave 0x1ef3 ỳ
IM_KEY_Ybelowdot 0x1ef4 Ỵ
IM_KEY_ybelowdot 0x1ef5 ỵ
IM_KEY_Yhook 0x1ef6 Ỷ
IM_KEY_yhook 0x1ef7 ỷ
IM_KEY_Ytilde 0x1ef8 Ỹ
IM_KEY_ytilde 0x1ef9 ỹ
IM_KEY_enspace 0x2002
IM_KEY_emspace 0x2003
IM_KEY_em3space 0x2004
IM_KEY_em4space 0x2005
IM_KEY_digitspace 0x2007
IM_KEY_punctspace 0x2008

IM_KEY_thinspace 0x2009
IM_KEY_hairspace 0x200a
IM_KEY_figdash 0x2012 ‒
IM_KEY_endash 0x2013 –
IM_KEY_emdash 0x2014 —
IM_KEY_Greek_horizbar 0x2015 ―
IM_KEY_hebrew_doublelowline 0x2017 ‗
IM_KEY_leftsinglequotemark 0x2018 ‘
IM_KEY_rightsinglequotemark 0x2019 ’
IM_KEY_singlelowquotemark 0x201a ‚
IM_KEY_leftdoublequotemark 0x201c “
IM_KEY_rightdoublequotemark 0x201d ”
IM_KEY_doublelowquotemark 0x201e „
IM_KEY_dagger 0x2020 †
IM_KEY_doubledagger 0x2021 ‡
IM_KEY_enfilledcircbullet 0x2022 •
IM_KEY_doubbaselinedot 0x2025 ‥
IM_KEY_ellipsis 0x2026 …
IM_KEY_minutes 0x2032 ′
IM_KEY_seconds 0x2033 ″
IM_KEY_caret 0x2038 ‸
IM_KEY_overline 0x203e ‾
IM_KEY_zerosuperior 0x2070 ⁰
IM_KEY_foursuperior 0x2074 ⁴
IM_KEY_fivesuperior 0x2075 ⁵
IM_KEY_sixsuperior 0x2076 ⁶
IM_KEY_sevensuperior 0x2077 ⁷
IM_KEY_eightsuperior 0x2078 ⁸
IM_KEY_ninesuperior 0x2079 ⁹
IM_KEY_zerosubscript 0x2080 ₀
IM_KEY_onesubscript 0x2081 ₁
IM_KEY_twosubscript 0x2082 ₂
IM_KEY_threesubscript 0x2083 ₃
IM_KEY_foursubscript 0x2084 ₄
IM_KEY_fivesubscript 0x2085 ₅
IM_KEY_sixsubscript 0x2086 ₆
IM_KEY_sevensubscript 0x2087 ₇
IM_KEY_eightsubscript 0x2088 ₈
IM_KEY_ninesubscript 0x2089 ₉
IM_KEY_EcuSign 0x20a0 ₠
IM_KEY_ColonSign 0x20a1 ₡
IM_KEY_CruzeiroSign 0x20a2 ₢
IM_KEY_FFrancSign 0x20a3 ₣
IM_KEY_LiraSign 0x20a4 ₤
IM_KEY_MillSign 0x20a5 ₥

IM_KEY_NairaSign 0x20a6 ₦
IM_KEY_PesetaSign 0x20a7 ₧
IM_KEY_RupeeSign 0x20a8 ₨
IM_KEY_WonSign 0x20a9 ₩
IM_KEY_Korean_Won 0x20a9 ₩
IM_KEY_NewSheqelSign 0x20aa ₪
IM_KEY_DongSign 0x20ab ₫
IM_KEY_EuroSign 0x20ac €
IM_KEY_careof 0x2105 ℅
IM_KEY_numerosign 0x2116 №
IM_KEY_phonographcopyright 0x2117 ℗
IM_KEY_prescription 0x211e ℞
IM_KEY_trademark 0x2122 ™
IM_KEY_onethird 0x2153 ⅓
IM_KEY_twothirds 0x2154 ⅔
IM_KEY_onefifth 0x2155 ⅕
IM_KEY_twofifths 0x2156 ⅖
IM_KEY_threefifths 0x2157 ⅗
IM_KEY_fourfifths 0x2158 ⅘
IM_KEY_onesixth 0x2159 ⅙
IM_KEY_fivesixths 0x215a ⅚
IM_KEY_oneeighth 0x215b ⅛
IM_KEY_threeeighths 0x215c ⅜
IM_KEY_fiveeighths 0x215d ⅝
IM_KEY_seveneighths 0x215e ⅞
IM_KEY_leftarrow 0x2190 ←
IM_KEY_uparrow 0x2191 ↑
IM_KEY_rightarrow 0x2192 →
IM_KEY_downarrow 0x2193 ↓
IM_KEY_implies 0x21d2 ⇒
IM_KEY_ifonlyif 0x21d4 ⇔
IM_KEY_partdifferential 0x2202 ∂
IM_KEY_partialderivative 0x2202 ∂
IM_KEY_emptyset 0x2205 ∅
IM_KEY_nabla 0x2207 ∇
IM_KEY_elementof 0x2208 ∈
IM_KEY_notelementof 0x2209 ∉
IM_KEY_containsas 0x220b ∋
IM_KEY_jot 0x2218 ∘
IM_KEY_squareroot 0x221a √
IM_KEY_radical 0x221a √
IM_KEY_cuberoot 0x221b ∛
IM_KEY_fourthroot 0x221c ∜
IM_KEY_variation 0x221d ∝
IM_KEY_infinity 0x221e ∞
IM_KEY_logicaland 0x2227 ∧

IM_KEY_upcaret 0x2227 ∧
IM_KEY_logicalor 0x2228 ∨
IM_KEY_downcaret 0x2228 ∨
IM_KEY_upshoe 0x2229 ∩
IM_KEY_intersection 0x2229 ∩
IM_KEY_union 0x222a ∪
IM_KEY_downshoe 0x222a ∪
IM_KEY_integral 0x222b ∫
IM_KEY_dintegral 0x222c ∬
IM_KEY_tintegral 0x222d ∭
IM_KEY_therefore 0x2234 ∴
IM_KEY_because 0x2235 ∵
IM_KEY_approximate 0x223c ~
IM_KEY_similarequal 0x2243 ≃
IM_KEY_approxeq 0x2245 ≅
IM_KEY_notapproxeq 0x2247 ≇
IM_KEY_notequal 0x2260 ≠
IM_KEY_identical 0x2261 ≡
IM_KEY_notidentical 0x2262 ≢
IM_KEY_stricteq 0x2263 ≣
IM_KEY_lessthanequal 0x2264 ≤
IM_KEY_greaterthanequal 0x2265 ≥
IM_KEY_includedin 0x2282 ⊂
IM_KEY_leftshoe 0x2282 ⊂
IM_KEY_includes 0x2283 ⊃
IM_KEY_rightshoe 0x2283 ⊃
IM_KEY_lefttack 0x22a2 ⊢
IM_KEY_righttack 0x22a3 ⊣
IM_KEY_uptack 0x22a4 ⊤
IM_KEY_downtack 0x22a5 ⊥
IM_KEY_upstile 0x2308 ⌈
IM_KEY_downstile 0x230a ⌊
IM_KEY_telephonerecorder 0x2315 ⌕
IM_KEY_topintegral 0x2320 ⌠
IM_KEY_botintegral 0x2321 ⌡
IM_KEY_quad 0x2395 ⎕
IM_KEY_topleftparens 0x239b ⎛
IM_KEY_botleftparens 0x239d ⎝
IM_KEY_toprightparens 0x239e ⎞
IM_KEY_botrightparens 0x23a0 ⎠
IM_KEY_topleftsqbracket 0x23a1 ⎡
IM_KEY_botleftsqbracket 0x23a3 ⎣
IM_KEY_toprightsqbracket 0x23a4 ⎤
IM_KEY_botrightsqbracket 0x23a6 ⎦
IM_KEY_leftmiddlecurlybrace 0x23a8 ⎨

IM_KEY_rightmiddlecurlybrace 0x23ac ⎬
IM_KEY_leftradical 0x23b7 ⎷
IM_KEY_horizlinescan1 0x23ba ⎺
IM_KEY_horizlinescan3 0x23bb ⎻
IM_KEY_horizlinescan7 0x23bc ⎼
IM_KEY_horizlinescan9 0x23bd ⎽
IM_KEY_ht 0x2409 ␉
IM_KEY_lf 0x240a ␊
IM_KEY_vt 0x240b ␋
IM_KEY_ff 0x240c ␌
IM_KEY_cr 0x240d ␍
IM_KEY_blank 0x2422 ␢
IM_KEY_signifblank 0x2423 ␣
IM_KEY_nl 0x2424 ␤
IM_KEY_horizlinescan5 0x2500 ─
IM_KEY_horizconnector 0x2500 ─
IM_KEY_vertbar 0x2502 │
IM_KEY_vertconnector 0x2502 │
IM_KEY_upleftcorner 0x250c ┌
IM_KEY_topleftradical 0x250c ┌
IM_KEY_uprightcorner 0x2510 ┐
IM_KEY_lowleftcorner 0x2514 └
IM_KEY_lowrightcorner 0x2518 ┘
IM_KEY_leftt 0x251c ├
IM_KEY_rightt 0x2524 ┤
IM_KEY_topt 0x252c ┬
IM_KEY_bott 0x2534 ┴
IM_KEY_crossinglines 0x253c ┼
IM_KEY_checkerboard 0x2592 ▒
IM_KEY_enfilledsqbullet 0x25aa ▪
IM_KEY_enopensquarebullet 0x25ab ▫
IM_KEY_filledrectbullet 0x25ac ▬
IM_KEY_openrectbullet 0x25ad ▭
IM_KEY_emfilledrect 0x25ae ▮
IM_KEY_emopenrectangle 0x25af ▯
IM_KEY_filledtribulletup 0x25b2 ▲
IM_KEY_opentribulletup 0x25b3 △
IM_KEY_filledrighttribullet 0x25b6 ▶
IM_KEY_rightopentriangle 0x25b7 ▷
IM_KEY_filledtribulletdown 0x25bc ▼
IM_KEY_opentribulletdown 0x25bd ▽
IM_KEY_filledlefttribullet 0x25c0 ◀
IM_KEY_leftopentriangle 0x25c1 ◁
IM_KEY_soliddiamond 0x25c6 ◆
IM_KEY_emopencircle 0x25cb ○
IM_KEY_circle 0x25cb ○

IM_KEY_emfilledcircle 0x25cf ●
IM_KEY_enopencircbullet 0x25e6 ◦
IM_KEY_openstar 0x2606 ☆
IM_KEY_telephone 0x260e ☎
IM_KEY_signaturemark 0x2613 ☓
IM_KEY_leftpointer 0x261c ☜
IM_KEY_rightpointer 0x261e ☞
IM_KEY_femalesymbol 0x2640 ♀
IM_KEY_malesymbol 0x2642 ♂
IM_KEY_club 0x2663 ♣
IM_KEY_heart 0x2665 ♥
IM_KEY_diamond 0x2666 ♦
IM_KEY_musicalflat 0x266d ♭
IM_KEY_musicalsharp 0x266f ♯
IM_KEY_checkmark 0x2713 ✓
IM_KEY_ballotcross 0x2717 ✗
IM_KEY_latincross 0x271d ✝
IM_KEY_maltesecross 0x2720 ✠
IM_KEY_leftanglebracket 0x27e8 ⟨
IM_KEY_rightanglebracket 0x27e9 ⟩
IM_KEY_kana_comma 0x3001 、
IM_KEY_kana_fullstop 0x3002 。
IM_KEY_kana_openingbracket 0x300c 「
IM_KEY_kana_closingbracket 0x300d 」
IM_KEY_voicedsound 0x309b ゛
IM_KEY_semivoicedsound 0x309c ゜
IM_KEY_kana_a 0x30a1 ァ
IM_KEY_kana_A 0x30a2 ア
IM_KEY_kana_i 0x30a3 ィ
IM_KEY_kana_I 0x30a4 イ
IM_KEY_kana_u 0x30a5 ゥ
IM_KEY_kana_U 0x30a6 ウ
IM_KEY_kana_e 0x30a7 ェ
IM_KEY_kana_E 0x30a8 エ
IM_KEY_kana_o 0x30a9 ォ
IM_KEY_kana_O 0x30aa オ
IM_KEY_kana_KA 0x30ab カ
IM_KEY_kana_KI 0x30ad キ
IM_KEY_kana_KU 0x30af ク
IM_KEY_kana_KE 0x30b1 ケ
IM_KEY_kana_KO 0x30b3 コ
IM_KEY_kana_SA 0x30b5 サ
IM_KEY_kana_SHI 0x30b7 シ
IM_KEY_kana_SU 0x30b9 ス
IM_KEY_kana_SE 0x30bb セ

IM_KEY_kana_SO	0x30bd	ソ
IM_KEY_kana_TA	0x30bf	タ
IM_KEY_kana_CHI	0x30c1	チ
IM_KEY_kana_tsu	0x30c3	ッ
IM_KEY_kana_TSU	0x30c4	ツ
IM_KEY_kana_TE	0x30c6	テ
IM_KEY_kana_TO	0x30c8	ト
IM_KEY_kana_NA	0x30ca	ナ
IM_KEY_kana_NI	0x30cb	ニ
IM_KEY_kana_NU	0x30cc	ヌ
IM_KEY_kana_NE	0x30cd	ネ
IM_KEY_kana_NO	0x30ce	ノ
IM_KEY_kana_HA	0x30cf	ハ
IM_KEY_kana_HI	0x30d2	ヒ
IM_KEY_kana_FU	0x30d5	フ
IM_KEY_kana_HE	0x30d8	ヘ
IM_KEY_kana_HO	0x30db	ホ
IM_KEY_kana_MA	0x30de	マ
IM_KEY_kana_MI	0x30df	ミ
IM_KEY_kana_MU	0x30e0	ム
IM_KEY_kana_ME	0x30e1	メ
IM_KEY_kana_MO	0x30e2	モ
IM_KEY_kana_ya	0x30e3	ャ
IM_KEY_kana_YA	0x30e4	ヤ
IM_KEY_kana_yu	0x30e5	ュ
IM_KEY_kana_YU	0x30e6	ユ
IM_KEY_kana_yo	0x30e7	ョ
IM_KEY_kana_YO	0x30e8	ヨ
IM_KEY_kana_RA	0x30e9	ラ
IM_KEY_kana_RI	0x30ea	リ
IM_KEY_kana_RU	0x30eb	ル
IM_KEY_kana_RE	0x30ec	レ
IM_KEY_kana_RO	0x30ed	ロ
IM_KEY_kana_WA	0x30ef	ワ
IM_KEY_kana_WO	0x30f2	ヲ
IM_KEY_kana_N	0x30f3	ン
IM_KEY_kana_conjunctive	0x30fb	・
IM_KEY_prolongedsound	0x30fc	ー
IM_KEY_Hangul_Kiyeog	0x3131	ㄱ
IM_KEY_Hangul_SsangKiyeog	0x3132	ㄲ
IM_KEY_Hangul_KiyeogSios	0x3133	ㄳ
IM_KEY_Hangul_Nieun	0x3134	ㄴ
IM_KEY_Hangul_NieunJieuj	0x3135	ㄵ
IM_KEY_Hangul_NieunHieuh	0x3136	ㄶ
IM_KEY_Hangul_Dikeud	0x3137	ㄷ

IM_KEY_Hangul_SsangDikeud 0x3138 ㄸ
IM_KEY_Hangul_Rieul 0x3139 ㄹ
IM_KEY_Hangul_RieulKiyeog 0x313a ㄺ
IM_KEY_Hangul_RieulMieum 0x313b ㄻ
IM_KEY_Hangul_RieulPieub 0x313c ㄼ
IM_KEY_Hangul_RieulSios 0x313d ㄽ
IM_KEY_Hangul_RieulTieut 0x313e ㄾ
IM_KEY_Hangul_RieulPhieuf 0x313f ㄿ
IM_KEY_Hangul_RieulHieuh 0x3140 ㅀ
IM_KEY_Hangul_Mieum 0x3141 ㅁ
IM_KEY_Hangul_Pieub 0x3142 ㅂ
IM_KEY_Hangul_SsangPieub 0x3143 ㅃ
IM_KEY_Hangul_PieubSios 0x3144 ㅄ
IM_KEY_Hangul_Sios 0x3145 ㅅ
IM_KEY_Hangul_SsangSios 0x3146 ㅆ
IM_KEY_Hangul_Ieung 0x3147 ㅇ
IM_KEY_Hangul_Jieuj 0x3148 ㅈ
IM_KEY_Hangul_SsangJieuj 0x3149 ㅉ
IM_KEY_Hangul_Cieuc 0x314a ㅊ
IM_KEY_Hangul_Khieuq 0x314b ㅋ
IM_KEY_Hangul_Tieut 0x314c ㅌ
IM_KEY_Hangul_Phieuf 0x314d ㅍ
IM_KEY_Hangul_Hieuh 0x314e ㅎ
IM_KEY_Hangul_A 0x314f ㅏ
IM_KEY_Hangul_AE 0x3150 ㅐ
IM_KEY_Hangul_YA 0x3151 ㅑ
IM_KEY_Hangul_YAE 0x3152 ㅒ
IM_KEY_Hangul_EO 0x3153 ㅓ
IM_KEY_Hangul_E 0x3154 ㅔ
IM_KEY_Hangul_YEO 0x3155 ㅕ
IM_KEY_Hangul_YE 0x3156 ㅖ
IM_KEY_Hangul_O 0x3157 ㅗ
IM_KEY_Hangul_WA 0x3158 ㅘ
IM_KEY_Hangul_WAE 0x3159 ㅙ
IM_KEY_Hangul_OE 0x315a ㅚ
IM_KEY_Hangul_YO 0x315b ㅛ
IM_KEY_Hangul_U 0x315c ㅜ
IM_KEY_Hangul_WEO 0x315d ㅝ
IM_KEY_Hangul_WE 0x315e ㅞ
IM_KEY_Hangul_WI 0x315f ㅟ
IM_KEY_Hangul_YU 0x3160 ㅠ
IM_KEY_Hangul_EU 0x3161 ㅡ
IM_KEY_Hangul_YI 0x3162 ㅢ
IM_KEY_Hangul_I 0x3163 ㅣ
IM_KEY_Hangul_RieulYeorinHieuh 0x316d ㅭ

IM_KEY_Hangul_SunkyeongeumMieum 0x3171 ㅁ
IM_KEY_Hangul_SunkyeongeumPieub 0x3178 ㅍ
IM_KEY_Hangul_PanSios 0x317f ㅿ
IM_KEY_Hangul_SunkyeongeumPhieuf 0x3184 ㆄ
IM_KEY_Hangul_YeorinHieuh 0x3186 ㆆ
IM_KEY_Hangul_AraeA 0x318d ㆍ
IM_KEY_Hangul_AraeAE 0x318e ㆎ

Annex B. Information to implement (informative)

There is no clause in 2nd draft, it will provide before final draft.