NAME

   XkbListComponents - List of components for one or more component types

SYNOPSIS

   XkbComponentListPtr XkbListComponents (Display **dpy*, unsigned int *device_spec*,
      XkbComponentNamesPtr *ptrns*, **int ***max_inout*);**

ARGUMENTS

   dpy connection to X server

   device_spec
      device ID, or XkbUseCoreKbd

   ptrns
      namelist for components of interest

   max_inout
      max # returned names, # left over

DESCRIPTION

   You may ask the server for a list of components for one or more component types. The request takes
   the form of a set of patterns, one pattern for each of the component types, including a pattern for the
   complete keyboard description. To obtain this list, use *XkbListComponents.*

   XkbListComponents queries the server for a list of component names matching the patterns specified
   in *ptrns.* It waits for a reply and returns the matching component names in an XkbComponentListRec
   structure. When you are done using the structure, you should free it using *XkbFreeComponentList.*
   device_spec indicates a particular device in which the caller is interested. A server is allowed (but not
   required) to restrict its reply to portions of the database that are relevant for that particular device.

   ptrns is a pointer to an XkbComponentNamesRec. Each of the fields in *ptrns* contains a pattern naming
   the components of interest. Each of the patterns is composed of characters from the ISO Latin1
   encoding, but can contain only parentheses, the wildcard characters '?' and '*', and characters
   permitted in a component class or member name. A pattern may be NULL, in which case no
   components for that type is returned. Pattern matches with component names are case sensitive. The '?'
   wildcard matches any single character, except a left or right parenthesis; the '*' wildcard matches any
   number of characters, except a left or right parenthesis. If an implementation allows additional
   characters in a component class or member name other than those required by the Xkb extension, the
   result of comparing one of the additional characters to either of the wildcard characters is
   implementation-dependent.

If a pattern contains illegal characters, the illegal characters are ignored. The matching process is carried out as if the illegal characters were omitted from the pattern.

max_inout is used to throttle the amount of data passed to and from the server. On input, it specifies the maximum number of names to be returned (the total number of names in all component categories). Upon return from *XkbListComponents,* max_inout contains the number of names that matched the request but were not returned because of the limit.

Component Names

Component names have the form *class(member)* where *class* describes a subset of the available components for a particular type and the optional *member* identifies a specific component from that subset. For example, the name "atlantis(acme)" for a symbols component might specify the symbols used for the atlantis national keyboard layout by the vendor "acme." Each class has an optional *default* member - references that specify a class but not a member refer to the default member of the class, if one exists. Xkb places no constraints on the interpretation of the class and member names used in component names.

The *class* and *member* names are both specified using characters from the Latin-1 character set. Xkb implementations must accept all alphanumeric characters, minus ('-') and underscore ('_') in class or member names, and must not accept parentheses, plus, vertical bar, percent sign, asterisk, question mark, or white space. The use of other characters is implementation-dependent.

STRUCTURES

The component name patterns used to describe the request are passed to *XkbListComponents* using an XkbComponentNamesRec structure. This structure has no special allocation constraints or interrelationships with other structures; allocate and free this structure using standard *malloc* and *free* calls or their equivalent:

```
typedef struct _XkbComponentNames {
    char *      keymap;        /* keymap names */
    char *      keycodes;      /* keycode names */
    char *      types;         /* type names */
    char *      compat;        /* compatibility map names */
    char *      symbols;       /* symbol names */
    char *      geometry;      /* geometry names */
} XkbComponentNamesRec, *XkbComponentNamesPtr;
```

XkbListComponents returns a pointer to an XkbComponentListRec:

```
typedef struct _XkbComponentList {
    int             num_keymaps;  /* number of entries in keymap */
    int             num_keycodes; /* number of entries in keycodes */
    int             num_types;    /* number of entries in types */
    int             num_compat;   /* number of entries in compat */
    int             num_symbols;  /* number of entries in symbols */
    int             num_geometry; /* number of entries in geometry;
    XkbComponentNamePtr    keymap;     /* keymap names */
    XkbComponentNamePtr    keycodes;   /* keycode names */
    XkbComponentNamePtr    types;      /* type names */
    XkbComponentNamePtr    compat;     /* compatibility map names */
    XkbComponentNamePtr    symbols;    /* symbol names */
    XkbComponentNamePtr    geometry;   /* geometry names */
} XkbComponentListRec, *XkbComponentListPtr;

typedef struct _XkbComponentName {
    unsigned short     flags;     /* hints regarding component name */
    char *             name;      /* name of component */
} XkbComponentNameRec, *XkbComponentNamePtr;
```

SEE ALSO

XkbFreeComponentList(3)

NOTES

Note that the structure used to specify patterns on input is an XkbComponentNamesRec, and that used to hold the individual component names upon return is an XkbComponentNameRec (no trailing 's' in Name).