

**NAME**

XAddHost, XAddHosts, XListHosts, XRemoveHost, XRemoveHosts, XSetAccessControl, XEnableAccessControl, XDisableAccessControl, XHostAddress, XServerInterpretedAddress - control host access and host control structure

**SYNTAX**

```
int XAddHost(Display *display, XHostAddress *host);
```

```
int XAddHosts(Display *display, XHostAddress *hosts, int num_hosts);
```

```
XHostAddress *XListHosts(Display *display, int *nhosts_return, Bool *state_return);
```

```
int XRemoveHost(Display *display, XHostAddress *host);
```

```
int XRemoveHosts(Display *display, XHostAddress *hosts, int num_hosts);
```

```
int XSetAccessControl(Display *display, int mode);
```

```
int XEnableAccessControl(Display *display);
```

```
int XDisableAccessControl(Display *display);
```

**ARGUMENTS**

- |                      |  |
|----------------------|--|
| <i>display</i>       | Specifies the connection to the X server.                                      |
| <i>host</i>          | Specifies the host that is to be added or removed.                             |
| <i>hosts</i>         | Specifies each host that is to be added or removed.                            |
| <i>mode</i>          | Specifies the mode. You can pass <b>EnableAccess</b> or <b>DisableAccess</b> . |
| <i>nhosts_return</i> | Returns the number of hosts currently in the access control list.              |
| <i>num_hosts</i>     | Specifies the number of hosts.   |
| <i>state_return</i>  | Returns the state of the access control.                                       |

**DESCRIPTION**

The **XAddHost** function adds the specified host to the access control list for that display. The server must be on the same host as the client issuing the command, or a **BadAccess** error results.

**XAddHost** can generate **BadAccess** and **BadValue** errors.

The **XAddHosts** function adds each specified host to the access control list for that display. The server must be on the same host as the client issuing the command, or a **BadAccess** error results.

**XAddHosts** can generate **BadAccess** and **BadValue** errors.

The **XListHosts** function returns the current access control list as well as whether the use of the list at connection setup was enabled or disabled. **XListHosts** allows a program to find out what machines can make connections. It also returns a pointer to a list of host structures that were allocated by the function. When no longer needed, this memory should be freed by calling **XFree**.

The **XRemoveHost** function removes the specified host from the access control list for that display. The server must be on the same host as the client process, or a **BadAccess** error results. If you remove your machine from the access list, you can no longer connect to that server, and this operation cannot be reversed unless you reset the server.

**XRemoveHost** can generate **BadAccess** and **BadValue** errors.

The **XRemoveHosts** function removes each specified host from the access control list for that display. The X server must be on the same host as the client process, or a **BadAccess** error results. If you remove your machine from the access list, you can no longer connect to that server, and this operation cannot be reversed unless you reset the server.

**XRemoveHosts** can generate **BadAccess** and **BadValue** errors.

The **XSetAccessControl** function either enables or disables the use of the access control list at each connection setup.

**XSetAccessControl** can generate **BadAccess** and **BadValue** errors.

The **XEnableAccessControl** function enables the use of the access control list at each connection setup.

**XEnableAccessControl** can generate a **BadAccess** error.

The **XDisableAccessControl** function disables the use of the access control list at each connection setup.

**XDisableAccessControl** can generate a **BadAccess** error.

**STRUCTURES**

The **XHostAddress** structure contains:

```
typedef struct {
    int family; /* for example FamilyInternet */
    int length; /* length of address, in bytes */
    char *address; /* pointer to where to find the address */
} XHostAddress;
```

The family member specifies which protocol address family to use (for example, TCP/IP or DECnet) and can be **FamilyInternet**, **FamilyInternet6**, **FamilyServerInterpreted**, **FamilyDECnet**, or **FamilyChaos**. The length member specifies the length of the address in bytes. The address member specifies a pointer to the address.

For the ServerInterpreted family, the length is ignored and the address member is a pointer to a **XServerInterpretedAddress** structure which contains:

```
typedef struct {
    int typelength; /* length of type string, in bytes */
    int valuelength; /* length of value string, in bytes */
    char *type; /* pointer to where to find the type string */
    char *value; /* pointer to where to find the address */
} XServerInterpretedAddress;
```

The type and value members point to strings representing the type and value of the server interpreted entry. These strings may not be NULL-terminated so care should be used when accessing them. The typelength and valuelength members specify the length in byte of the type and value strings.

**DIAGNOSTICS**

**BadAccess** A client attempted to modify the access control list from other than the local (or otherwise authorized) host.

**BadValue** Some numeric value falls outside the range of values accepted by the request. Unless a specific range is specified for an argument, the full range defined by the argument's type is accepted. Any argument defined as a set of alternatives can generate this error.

**SEE ALSO**

XFree(3)

*Xlib - C Language X Interface*