XtConvertAndStore, XtCallConverter – invoke resource converters

Boolean XtConvertAndStore(*widget***,** *from_type***,** *from***,** *to_type***,** *to_in_out***)**
  Widget *widget***;**
  String *from_type***;**
  XrmValuePtr *from***;**
  String *to_type***;**
  XrmValuePtr *to_in_out***;**

void XtCallConverter(*dpy*, *converter*, *args*, *num_args*, *from*, *to_in_out*, *cache_ref_return*)
  Display* *dpy*;
  XtConverter *converter*;
  XrmValuePtr *args*;
  Cardinal *num_args*;
  XrmValuePtr *from*;
  XrmValuePtr *to_in_out*;
  XtCacheRef* *cache_ref_return*;

args	Specifies the argument list that contains the additional arguments needed to perform the conversion, or NULL.
converter	Specifies the conversion procedure that is to be called.
from	Specifies the value to be converted.
from_type	Specifies the source type.
num_args	Specifies the number of additional arguments (often zero).
to_type	Specifies the destination type.
to_in_out	Returns the converted value.
widget	Specifies the widget to use for additional arguments, if any are needed, and the destroy callback list.
dpy	Specifies the display with which the conversion is to be associated.

The XtConvertAndStore function looks up the type converter registered to convert from_type to to_type, computes any additional arguments needed, and then calls **XtCallConverter**. (or **XtDirectConvert** if an old-style converter was registered with **XtAddConverter** or **XtAppAddConverter**.) with the from and to_in_out arguments.

The **XtCallConverter** function looks up the specified type converter in the application context associated with the display and, if the converter was not registered or was registered with cache type **XtCacheAll** or **XtCacheByDisplay** looks in the conversion cache to see if this conversion procedure has been called with the specified conversion arguments. If so, it checks the success status of the prior call, and if the conversion failed, **XtCallConverter** returns **False** immediately; otherwise it checks the size specified in the *to* argument and, if it is greater than or equal to the size stored in the cache, copies the information stored in the cache into the location specified by *to->addr*, stores the cache size into *to->size*, and returns **True**. If the size specified in the *to* argument is smaller than the size stored in the cache, **XtCallConverter** copies the cache size into the *to->size* and returns **False**. If the converter was registered with cache type **XtCacheNone** or no value was found in the conversion cache, **XtCallConverter** calls the converter and, if it was not registered with cache type **XtCacheNone**, enters the result into the cache. **XtCallConverter** then returns what the converter returned.

The *cache_ref_return* field specifies storage allocated by the caller in which an opaque value will be stored. If the type converter has been registered with the **XtCacheRefCount** modifier and if the value returned in in *cache_ref_return* is non-NULL, then the call should store the *cache_ref_return* value in order to decrement the reference count when the converted value is no longer required. The *cache_ref_return* argument should be ULL if the caller is unwilling or unable to store the value.

XtAppReleaseCacheRefs(3Xt)
X Toolkit Intrinsics – C Language Interface
Xlib – C Language X Interface