Description of the Application Group Extension

Implementation for the X11 Sample Server

Kaleb KEITHLEY, FUJITSU Limited.

Version 1.0

The following document explains the server side of the Application Group Extension.

Abstract

The following document explains the server side of the Application Group Extension.

Table of Contents

To understand this document and the accompanying source code, you should know the C language, should be familiar with X server internals, and should also have a general knowledge of the X Window System.

AppGroup Server Public Functions

The AppGroup extension adds seven new functions that are called from elsewhere in the server. They are: XagExtensionInit, XagDefaultColormap, XagRootVisual, XagLeader, XagIsControlledRoot, XagConnectionInfo, XagCallClientStateChange.

XagExtensionInit is the extension initialization function called from InitExtension in mi/miinitext.c. Note that an new resource type, RT_APPGROUP, is created, specifying the destructor function XagAppGroupFree.

XagDefaultColormap returns the colormap ID that was specified in the creation of the AppGroup. Any time CopyFromParent is specified for a top-level window's colormap, i.e. in a CreateWindow or ChangeWindowAttributes request, this function is called to see if there is an AppGroup specific colormap to use. If there is one, its ID is returned, otherwise None is returned.

XagRootVisual returns the visual ID that was specified in the creation of the Appgroup. Like XagDefaultColormap, when CopyFromParent is specified for a top-level window's visual in a CreateWindow request, this function is called to see if there is an AppGroup specific visual to use. If there is one, its ID is returned, otherwise 0 (zero) is returned.

XagLeader returns the ClientPtr of the client that is the AppGroup Leader. Normally when an application maps or configures a top-level window a MapRequest or ConfigureRequest event is delivered to the client, e.g. a window manager, that has selected SubstructureRedirect on the root window. However, when the application is part of an AppGroup, the MapRequest and ConfigureRequest events are delivered to the AppGroup Leader instead.

XagIsControlledRoot returns a boolean: True if the window is a top-level window of a client in an AppGroup, False otherwise. In a combined server, i.e. one that provides both UI and printing, the application may create and map windows on the "printing" screens; thus it becomes necessary to discriminate between the AppGroup's root window and other root windows. If an AppGroup member creates and maps a [top-level] window then the window's parent [the root window] is tested to determine whether to send MapRequest or ConfigureRequest events to the AppGroup Leader to to some other client.

In the trivial case XagIsControlledRoot returns True if the parent window has no parent itself, i.e. it is a root window. In the case where the application is embedded, indicated by the singleScreen attribute being True, the parent's drawable ID is compared to the AppGroup's root window ID, and if it is the same, True is returned. If neither case is true, then False is returned.

XagConnectionInfo returns an abreviated version of the connection setup information. When an embedded AppGroup is created the server returns only the information about the [UI] screen that the application is embedded within in the connection setup in order to prevent the application from creating windows on other screens; thus attempting to guarantee that any window that should be embedded can be reparented into the AppGroup Leader's window hierarchy.

XagCallClientStateChange is called to invoke the extension's client state change callback additional times as necessary -- currently only once, after the auth data becomes available between ClientStateInitial and ClientStateConnected. Client state change callbacks were introduced in the Record extension, which specifies when the callbacks are invoked. Unfortunately the points at which they are called are not necessarily the best as far as the AppGroup Extension is concerned. Adding an additional state and calling all the callbacks works too, however this seemed unnecessary overkill.

AppGroup Server Private APIs

The AppGroup extension adds the following functions which are private to the extension: ProcXagDispatch and SProcXagDispatch, ProcXagQueryVersion and SProcXagQueryVersion, ProcXagCreate and SProcXagCreate, ProcXagDestroy and SProcXagDestroy, ProcGetAttr and SProcGetAttr, ProcXagQuery and SProcXagQuery, ProcXagCreateAssoc and SProcXagCreateAssoc, ProcXagDestroyAssoc and SProcXagDestroyAssoc, XagResetProc, and XagAppGroupFree.

The ProcXagDispatch, SProcXagDispatch, and XagResetProc functions should be familiar to anyone familiar with X server internals and I won't elaborate on them here. Similarly the wrapper functions: SProcXagQueryVersion, SProcXagCreate, SProcXagDestroy, SProcXagGetAttr, SProcXagQuery, SProcXagCreateAssoc, and SProcXagDestroyAssoc, as wrappers which handle swapping integer data into the host's byte order will not be explained in any detail.

ProcXagQueryVersion returns the major and minor versions of the AppGroup extension supported by the server.

ProcXagCreate creates an AppGroup. A new record in a linked list of AppGroups is allocated and initialized. The attributes from the request are validated and copied to the AppGroup record. If necessary an abbreviated version of the connection setup information is compiled and also stored in the AppGroup record. The first time an AppGroup is created a client-state-change callback is registered and a reference count is incremented.

ProcXagDestroy destroys an AppGroup an AppGroup by calling FreeResource specifying the AppGroup ID. This will result in the destructor function XagAppGroupFree being called. The reference count is decremented and when it reaches zero the client-state-change callback is deleted.

ProcXagGetAttr returns the AppGroup Attributes to the requesting client.

ProcXagQuery returns the AppGroup ID of an arbitrary resource to the requesting client.

ProcXagCreateAssoc creates an association between an X window ID and system-specific data. In native X this functionality is unnecessary but for various personal computers, e.g. Macintosh, OS/2, and MS-Windows it is necessary to associate an X window ID with the system's native window identifier so that when the AppGroup Leader issues a ReparentWindow request the personal computer X server can lookup the system-specific window ID and make the necessary function call(s) with it.

ProcXagDestroyAssoc destroys the association created with ProcXagCreateAssoc.

XagResetProc removes the client-state-change callback, sets the reference count to zero, and frees all the AppGroup records in the linked list by calling XagAppGroupFree.

XagAppGroupFree calls CloseDownClient for each client in an AppGroup if the AppGroup has a leader, unlinks the AppGroup record from the linked list, frees allocated memory referenced by the record, and finally frees the record itself.

Known Problems in this release.

In a combined UI/Print server the connection setup returned to an embedded application will not have information about the print screens.

The LBX proxy caches connection setup information and will return incorrect connection setup information to an embedded client.