CSE 420W
Mozilla Group
Design Document

Overview

This document describes the proposed CDE Calendar connection design for the Calendar component of Mozilla. This design is based on a request for enhancement "bug" (Bugzilla Bug # 188660 http://bugzilla.mozilla.org/show_bug.cgi?id=188660). The design will inherit the existing design of the Mozilla code base. Any new technologies to fulfill the requirements for the enhancement will be described here.

Goals of the system

The goal of this project is to allow a Mozilla Calendar Client to connect to and utilize a Sun CDE Calendar Server. This will include the ability to:

- Connect to, Authenticate with, and Disconnect from a Sun calendar server
- Create and Remove calendars
- Add, Modify, and Delete events
- View Events
- Update the Mozilla User Interface to accommodate our enhancements

System Requirements

The following will be required for proper operation of our project:

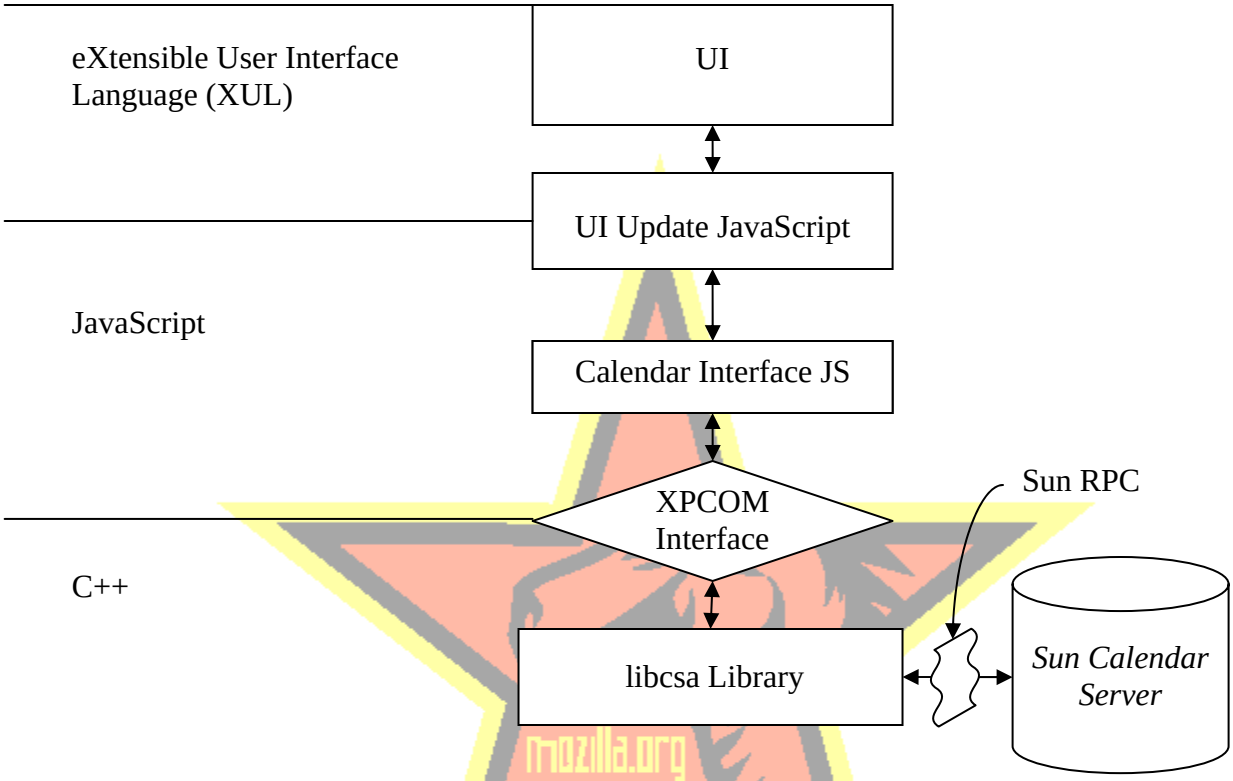
- Mozilla 1.3b
- Mozilla Calendar component
- Sun Workstation running Solaris 8
- Sun CDE Calendar Server version 5 data version 4
- libcsa calendar library

Approach

Our approach will follow the Mozilla development model. We will utilize Mozilla's cross platform component architecture (XPCOM). XPCOM allows for binary compatibility with the libcsa calendar interface. XPCOM will allow JavaScript to communicate with our C++ components.

Our user interface will be developed using Mozilla's User Interface tools (XUL-pronounced "Zool").

Diagram 1 - Basic Infrastructure:



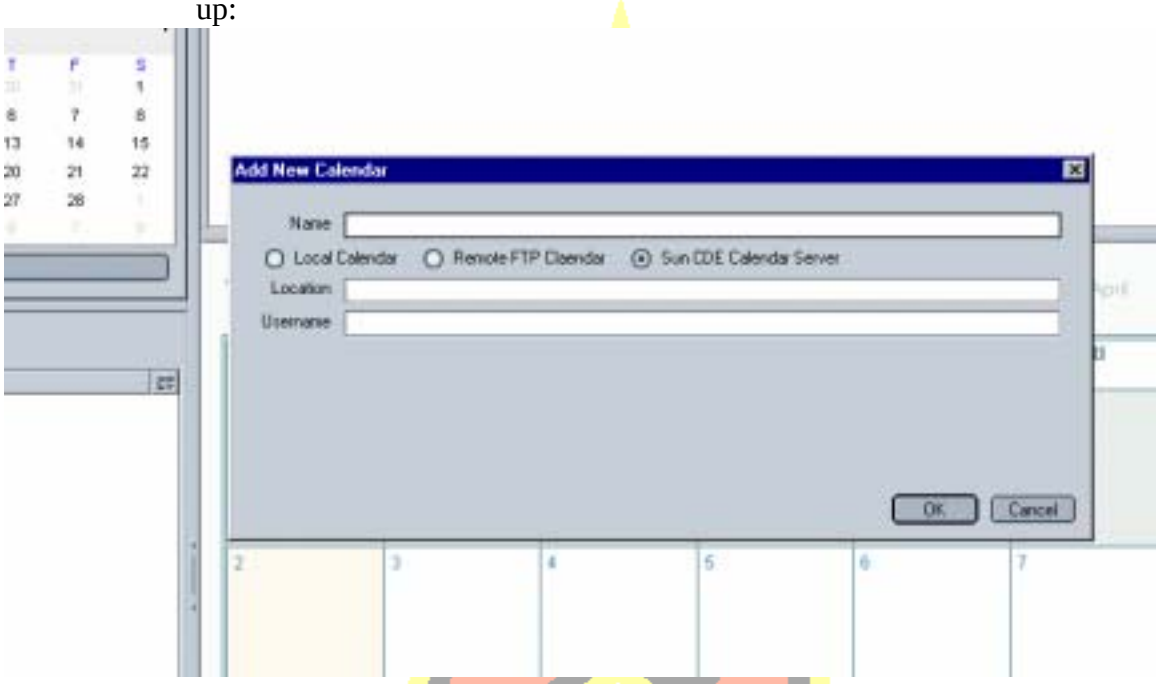
System Flow

Our system will function as follows:

1. The user will interface with our JavaScript/XUL front end
2. JavaScript handlers will communicate with our XPCOM/IDL interface and the polymorphic objects from which we inherit
3. Our XPCOM component will interact with libcsa
4. The libcsa library, through a series of RPC calls, will instruct the Sun Calendar Server to perform the requested actions
5. The Sun Calendar Server will then return the requested data which will bubble up through the series of steps above

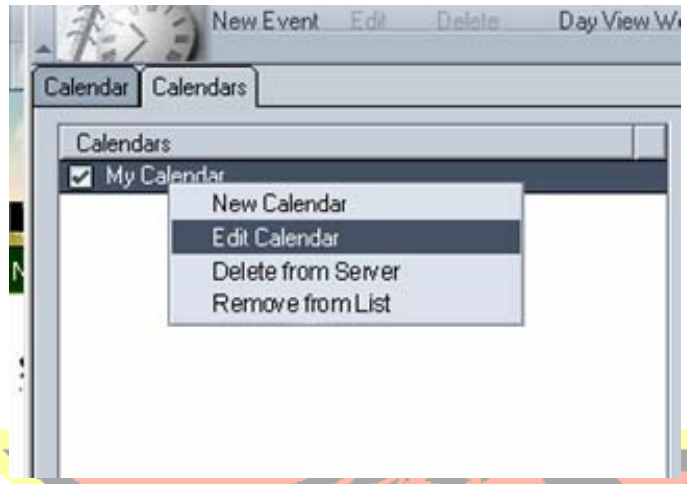
Functional Development

- Connect to a Sun calendar server
    - The user will enter his or her username and location into the current UI. The password field is ignored by the libcsa library in Solaris 8 version 5 data version 4, and as such will not be included. Here is a proposed mock-up:



    - The UI will call preexisting functions within Mozilla. These functions already communicate with a layer above the layer at which we are developing.
    - JavaScript will call our interface functions which will, in turn call our XPCOM component.
    - The XPCOM component will call functions specified in libcsa to create a connection with a calendar server. The function we will be calling is *csa_logon()*.
- Authenticate with a Sun calendar server
    - Authentication will be performed implicitly upon connection. The libcsa library handles authentication on a RPC level.
- Disconnect from a Sun calendar server
    - This will disconnect from the calendar server with *csa_close()*.
    - This will occur when the UI closes a calendar window.
    - The calendar will remain on the server awaiting connections.
- Create a calendar
    - If a user enters a calendar name that doesn't exist, the XPCOM component will instruct the libcsa backend to create a calendar of that name on the server.
    - The libcsa command is *csa_add_calendar()*.

- Remove a calendar
  - There are 2 methods to remove a calendar from the User Interface: Deleting the calendar from the server; Removing the calendar from the list of calendars presented. Here is a proposed mock-up:



    - **Delete from Server -** Deleting the calendar from the server will entail deallocating the calendar from the Mozilla instance. A call will be made to our XPCOM interface to delete the calendar, and then the libcsa command *csa_delete_calendar()* will be used.
    - **Remove from List** - Removing the calendar from the list of calendars will deallocate the calendar from Mozilla, and force a "Disconnect from a Sun calendar server" event.
- Add events
  - The user will add an event using the preexisting Mozilla add event dialog. This dialog will send a command to our JavaScript interface which will communicate with our XPCOM component. The XPCOM component will call the libcsa command *csa_add_entry()*.
- Delete events
  - When a user deletes an event through the preexisting Mozilla interface, the JavaScript will call our XPCOM component which will run the libcsa command *csa_delete_entry()*.
- Modify events
  - The user will change properties of events through the preexisting Mozilla interface. Any properties modified will be sent to our XPCOM interface which will use one of the many libcsa update commands.
- Get Events
  - Get by Month
    - When the view changes to a month view, the JavaScript will call the XPCOM *getEventsByDateRange()* function which will call the appropriate libcsa commands.

- Get by Week
    - When the view changes to a week view, the JavaScript will call the XPCOM *getEventsByDateRange()* function which will call the appropriate libcsa commands.
- Get by Day
    - When the view changes to a day view, the JavaScript will call the XPCOM *getEventsByDateRange()* function which will call the appropriate libcsa commands.
- These functions will return the appropriate events which will be rendered to the screen.

Security

We will be following the Mozilla Security Guidelines which can be found at:
http://www.mozilla.org/projects/security/components/reviewguide.html
Due to the current implementation of the libcsa library, we will not require the user to enter a password. This is due to the fact that the current version of the libcsa library (as stated above) ignores the password parameter in function calls. The server relies on the underlying RPC calls for user level authentication.

Estimated Time Requirements

- Getting to know Mozilla — 80 person hours
- Requirements Document — 50 person hours
- Design Document — 80 person hours
- Prototype
    - UI front end — 40 person hours
    - JavaScript middleware — 80 person hours
    - XPCOM component — 80 person hours
    - libcsa implementation — 120 person hours
- Alpha Testing — 60 person hours
- Beta Development — 100 person hours
- Final Version Development — 60 person hours