CEF Technical Design

This document describes the technical details of how CEF works within websites. This document is intended for developers interested in either integrating CEF into their website or those that would like to implement the CEF protocol on a new platform/language/technology.
CEF Overview & Technical architecture give an overall idea about CEF.
The CEF Scenarios document provides a business level overview of CEF and its applicability to business problems.

CEF Overview

CEF stands for Collaborative E-Business Framework. CEF is intended to improve collaboration between participating websites in a portal network or in an online marketplace.
CEF is based on open source technology using HTTP/XML. CEF protocol and the source code are both open source and can be found on the sourceforge website.
CEF is language /technology neutral and is intended to integrate and allow collaboration between websites that could potentially be written in diverse set of web technologies, languages and platforms. To achieve this CEF adopts a lowest common denominator approach to the technology requirements placed on the participating websites. Sites should only have to adopt or change the minimum technology possible to work with other sites. Just because a vendor decides to participate in a portal framework does not imply that they should have to reimplement their complete website.
CEF has the following features: -

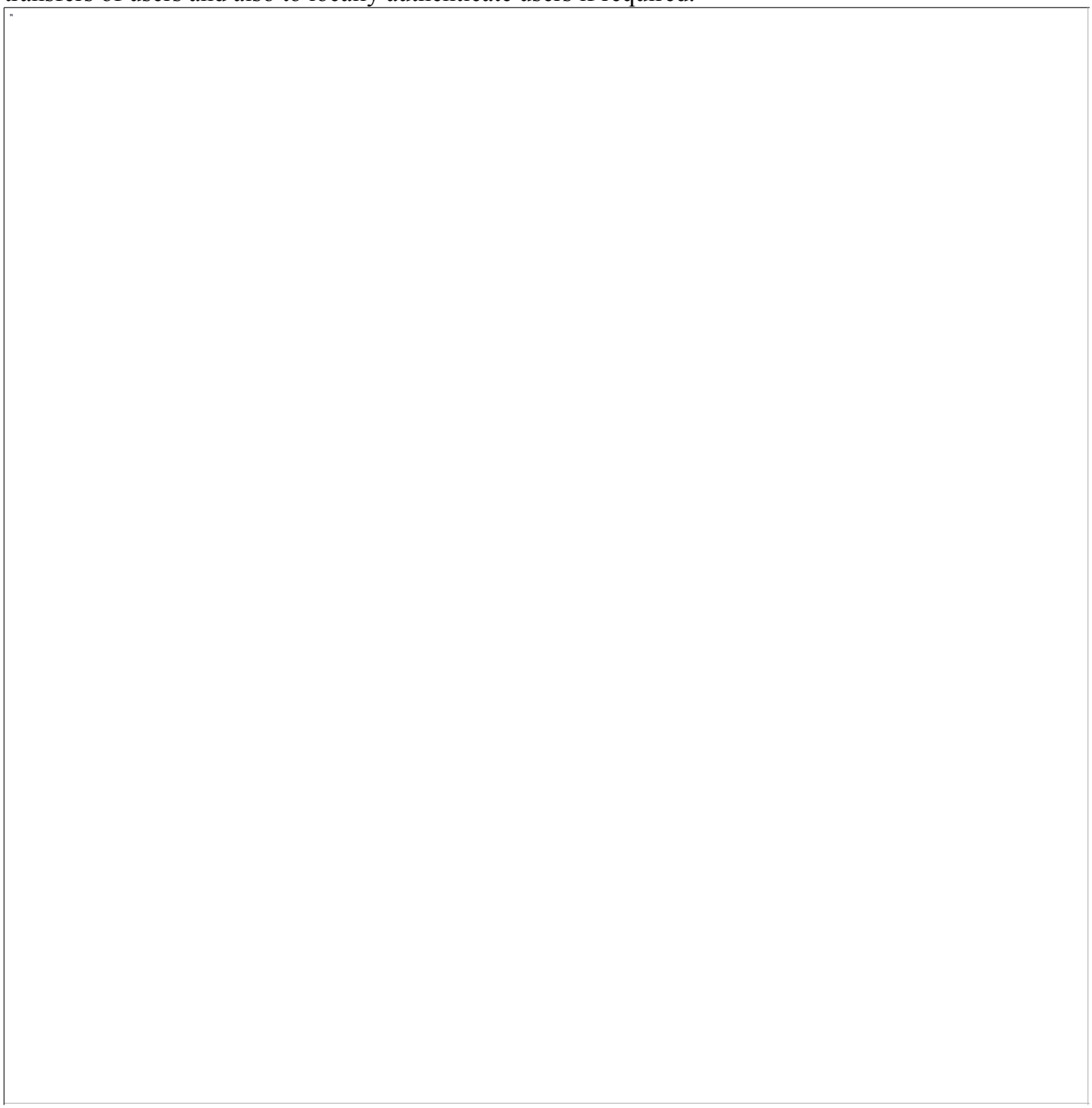
- *Single sign on/login* – CEF facilitates a single login of the user throughout the portal network. Thus the user does not have to deal with multiple logins and password maintenance issues at each and every site they visit within the portal network.
- *Session Sharing* – CEF allows websites in a portal network to share session information about users. This can be very useful when customizing/personalizing the user experience.
- *Customized & Selective Sharing of information with partners* – CEF makes it so that this sharing of user information between various sites is selective and can be modified based on the level of trust between the central portal and the various sites.
- *Enables Dynamic Gathering of User Attributes* – CEF enables gathering of user information as its needed and when its needed rather than having users fill out 2-3 page registration forms upfront.

CEF Technical Architecture

Below is an overview of the various CEF components and the technical architecture
In this architecture, we show the portal, which represents the main entry point for the user. In a CEF architecture the portal is the point of user authentication within the network. The portal may also be maintaining profiles for the user. The portal will also be responsible for rendering the main menu page for a user that first visits it. Thus in essence the portal is a launching off point for the user. The portal may also be hosting other applications. But logically they should be treated as separate applications that just happen to reside on the same machine as the portal. With this level of architectural separation it becomes easy in the future to move different applications within the portal to different machines and even hosted locations, as scalability and performance demands on the portal increase.

The other major part of the CEF architecture is the remote website. In the diagram below, Website A and

Website B represent the remote websites. The remote websites are a part of the portal network. But the difference is they do not need to maintain profiles or login or personalization information about the user. The remote websites can thus concentrate on their core competency of providing the content they specialize it rather than getting sidetracked in dealing with user registration, authentication, profiling and other such tertiary issues. These websites when they need personalization information request and get it from the portal. In some cases, super secure sites may want to do local re-authentication of the user. However the expectation is that this will happen in the background between the portal and the remote website and will be transparent to the end users and their browser(s).
The difference between websites A and B is in just how they could be implemented. We will cover this in more detail in the CEF components below. But the major difference is that, some websites may just have static content/html with no authentication/ session management modules setup. In this case Web Site B shows how these websites can be integrated into the CEF architecture of the portal network to receive transfers of users and also to locally authenticate users if required.



Core CEF Components

As seen in the diagram above, the CEF components that form the core include : -

CEFSend

This module needs to reside both on the portal and a special version also resides on the remote website. The CEF Send module is primarily used to transfer the user securely from the central portal to the remote website securely. It allows a variable amount of information to be passed from the portal to the remote website. It performs the following functions :-

1. Receive a secure transfer request from the browser. E.g. http://www.coolportal.com/CEF/cgi-bin/CEFSend.cgi?app_id=websiteA-mainpage &return_url=http://www.coolportal.com/main.cgi.
2. CEFSend will first off all invoke the portal's session mgr module to ensure that the user has a valid session (since the user might bookmark and try to directly invoke the CEFSend url without having done a login at the portal).Once the Session Mgr returns a session valid ok, the CEFSend module now extracts the app_id parameter from the url and looks up a local configuration file called AppId2ServerId.ini.
3. The AppId2ServerId.ini maps the CEF application Id to a configuration file typically named after the URL of the remote application. CEFSend now open this server id config file.
4. There is one server id config file per application. It contains details about the remote application, like the url, the information to be passed during transfer, if symmetric encryption is used, then it can store the key to use to encrypt data to be sent to the remote site.
5. If required the CEFSend module can be overloaded so that the portal can now check to determine if the user is authorized to access the application. Although preferably if the user does not have access to the url, then the link should not have come up on the main page itself.
6. CEFSend now determines what information needs to be sent to the remote website during the transfer. To make it dynamic this can be easily configured in the config file for the remote website. E.g. Remote Web Site A would only like to receive session id and user id. Whereas another site might want to get more information about the user during the transfer.
7. CEFSend now marshals all of this information along with standard CEF transfer information like user_id, target_app_id & source_app_id.
8. CEFSend has now constructed the URL for the remote website. E.g. http://www.remote-website-A.com/CEF/CEFReceive.cgi?params=params_to_be_passed_here
9. CEFSend now redirects the user's browser to the remote website CEF Receive url constructed above.

Refer to the CEF Programmer's guide for examples, parameters and methods for the CEF Send module.

CEFReceive

CEF Receive represents the flip side or the receiving end of the secure transfer. It is used to receive secure user sessions on the remote web site from the portal. A special version also resides on the portal to receive users that are transferred back from the remote websites for whatever reasons.
It performs the following functions :-

1. Determines the received information from CEF Send.
2. Decodes the received information if it was sent encrypted by the central portal.
3. It then invokes the local web site's session manager to validate the session or create a new one as long as the user is authentic.
4. It then transfers the user's browser to the actual content page.

Refer to the CEF Programmer's guide for examples, parameters and methods for the CEF Send module.

Apache API Session Manager/ Content Protector

As mentioned in the basic flow below, one of the main assumptions being made is that all CGIs on a website can be redirected to use a single session manager, user authentication and login scripts for protecting content on the web site.
But what happens if we consider a relatively static web site, which just has html and other content, but no CGIs. What if the portal needs to bring in a site like this into their framework and send secure user transfers to this site (also ensuring that unauthorized users do not gain access to the content on the website).
In this case , CEF contains an Apache API module that acts as the Session Manager/Content Protector. It can receive secure user session transfers from the portal, it can create local session for the users and can transfer the user back to the portal and even send billing or reporting events about content used.
This module can also be useful to do billing of static content based on one time or pay per use kinds of models, since the Apache API module is the gatekeeper and also knows how to communicate back to the portal. Thus the rest of the static content does not have to be changed or the site does not need to be revamped to enable participation in the portal framework.
The apache API module functionality can be easily duplicated into a NSAPI or an ISAPI module for (Netscape or IIS web servers).
Refer to the CEF Programmer's guide for examples, parameters and configuration options for this module.

CEF Events API Handler (CEF Portal)

So far in CEF we have covered session management, secure session transfer, user authentication (local and remote). But the other key component of collaboration is back end communication between the portal and the remote website. This is facilitated using the CEF Portal module.
Backend communication with the portal may be required for :-

- Determining if a user is authorized to view a particular page (if the remote site does not implement its own authorization policies).
- Retrieving information from the user profile to personalize the user experience on the remote web site.
- Send reporting, billing , usage information about the services provided to the user back to the central portal. This can also be useful for reconciliation of fees to be paid between the central portal and the remote service provider web sites.

Local Website Components

Below is a brief explanation of some of the local website components that are required for CEF to operate.

Session Manager

The session manager module is typically a key component of any website these days. A session manager is responsible typically for

- Validating that a user has a valid session.
- Creating a session.
- Modifying session information.
- Providing methods to allow other modules to access information from the user session.
- Terminating the session when a session timeout occurs.

CEF utilizes some of the functionality above when transferring a user from the portal to the remote website.

Login & User Authentication

Each site needs to implement its own mechanisms for login and for user authentication. The login module is expected to request credentials from the user depending on the authentication methods supported by the portal. Some portals may be satisfied with userid /password and hence may require just a simple userid password form. Others may require digital certificates and hence require passing those credentials.
Once the user credentials have been collected the login module will typically relay them to a security or user authentication subsystem within the portal which will be the one that actually authenticates the user's credentials by potentially looking them up in some user repository (could be an LDAP based directory or a relational database).
Typically for personalization, it is expected that the login module should allow for an additional parameter representing the local web page URL to transfer the user to. This allows users that started off from some other page on the portal to be sent over for login and can then be transferred back to their page, rather than having the users re-navigate through the menus to reach their destination page. But this is more of a personalization issue and is up to the portal to decide.

Profile Management

Another key component that most portals will need and will have to provide access to CEF is profile manager. The profile manager typically

- Maintains user profiles.
- Provides methods/means to get /update profile user profile information (by local components).

CEF may require access to the profile during session transfer if additional information needs to be transferred along with the user session to the remote website. It may also need access when the remote website needs more profile information.

Basic Flow of CEF

The flow is based on the numbering in the diagram above

User Authentication at the Portal

Steps 1-5 in the above diagram relate to user authentication at the portal the first time the user visits the portal

1. Lets start with the User visiting the main page of the portal (e.g. http://www.coolportal.com/main.cgi).
2. The main.cgi on the website invokes a local Session Mgr module to check for user authentication determine if the user is authenticated and has a valid session.
    a. Lets assume the Session Mgr uses cookies to create sessions for the portal
    b. Session Mgr now checks the http cookie cgi environment variable to determine if the user has a valid session.
    c. Since this is the first time around, the user does not have a valid session.
3. Session Mgr now redirects the browser (using http redirect) to a local login cgi on the portal (perhaps with a url reference to the page the user should be sent back to after login e.g. http://www.coolportal.com/login.cgi?return_url=/main.cgi).
    a. Browser now contacts the login cgi module on the portal.
    b. Login.cgi requests user for credentials. (This can be either by presenting a standard userid password form or via redirecting the user to a secure web server that accepts certificates).

   c. Browser now collects the user credentials and presents them to the login cgi module (either via http get or post)
4. The local login.cgi now passes these credentials to the portal's standard user authentication module which may lookup the user's credentials in the User database to make sure the user is authentic and is a valid user of the portal.
   a. Once the validation comes back positive, the login.cgi could ask the session manager to generate a session token for the user (by passing it the credentials sent over by the user authentication system).
   b. The login.cgi gets the session token from the session manager, converts it into a cookie format string and sends it in the http_cookie header field back to the browser as a response.
   c. The browser will now have a session established for this user on the portal.
5. Login.cgi now uses the return url to transfer the browser (via a url redirect) to the main page of the portal.
   a. The browser will contact the main.cgi on the portal.
   b. The main.cgi will now invoke session mgr to check session validity as before.
   c. Since this time the user has a session, the main.cgi will now render the main page for the user.

Assumptions & Notes :

- Here the assumption is that the Session Mgr is a standalone module/library that can be invoked from the CEFSend /CEF Receive modules and from all the other CGIs to determine if a user has a valid session at the portal.
- The User Session at the portal is assumed to be nothing more than a client side cookie on the browser. Since the local session mgr module on the portal checks whether the session is valid, the contents of the cookie are totally up to the local site to determine. Of course common attributes that need to be stored include userid, session id, etc.
- One major assumption being made here is that all CGIs/ programs within the portal are able to check for session validity using the local session mgr module.
- What is proposed above is just one implementation of session management and user authentication. CEF does not dictate how these happen on the portal or on the remote web site.

Session Transfer of the user to Remote Website A

Steps 6-10 in the diagram above represent the actions that take place during a secure session transfer from the portal to another website

6. Lets say the user now clicks on some link in the menu or content on the portal website that points to content on remote website A.
   a. In this case actually the browser points to the local CEFSend cgi with parameters requesting a secure transfer to the remote URL.
   b. CEF Send does some munging of the information – refer to CEF Programmers guide for the nuts and bolts.
7. CEF Send now redirects the Browser to the CEF Receive URL over on the remote web site A. CEF Receive on the remote website does processing of the information – refer to CEF Programmers guide for the nuts and bolts.
8. CEF Receive after local authentication (optionally) now is ready to let the user in and hence sets a cookie on the browser establishing a session for the remote web site A. In the process of establishing a session, website A may need more information about the user from their profile maintained at the portal. In this case the CEFPortal module can be used to hold a backend conversation with the main portal to request additional user profile information or authorization information.
9. CEF Receive now redirects the browser to the local cgi on website A that can render the requested user

content.

10. Once the user has received the service, if required, the remote website A can send events back to the portal using CEFPortal. This can be very useful if billing, usage or reporting information need to be sent back to the portal for further processing or reconciliation.

Assumptions and Notes

- The big assumption being made here is that all remote website links that need to be secured will have to be translated from direct links, into CEF Secure Session links as shown in step 6a above.
- In step 6a when CEFSend is invoked the parameters passed are an id representing the application to visit rather than the actual URL of the remote website. This allows for additional validation and checking.
- It is assumed that the SessionMgr module provides whatever profile information is needed for the secure transfer.

CEF Configuration

In order for CEF to work between the portal and the remote web site, some configuration work needs to be done. This configuration needs to happen both on the portal and the remote website
Below is a summary of the configuration files

- *cef.ini* – the main configuration file for both CEF & the Secure Portal
- *AppId2ServerId.ini* – Maps application IDs to servers (used on the sending server)
- *AppId2LocalUrl.ini* – Maps applications IDs to URLs on the local server (used on the receiving server)
- *Servers/server_id_files* – A set of files, which contain the configuration for remote servers. The "server_id" is taken from the "AppId2ServerId.ini" file based on the given application ID.

Refer to the CEF Programmers guide for detailed explanations of the configuration options

Common Issues / Questions

1. What if user visits the remote site without a login or using a bookmark?
Ans> In this case, the remote site CEF cgi or the apache API content protector can be configured with the login URL of the central portal.
So if a user does show up without login to the remote site, the CEF receive will be invoked or the Apache API will be invoked and it will redirect the user's browser to the login URL of the portal. Once the login has been performed, the login module can directly invoke the local CEFSend module to do a secure transfer of the user to the requested URL

2. How does CEF maintain cross web site session?
Ans> CEF does not maintain the user's session across web site. The local sites and the portal are responsible for maintaining their own user's sessions. CEF just provides the ability to transfer the user's session from the portal to the remote website. Thus both the portal and the remote web site have separate sessions with the browser. This allows different web sites to establish their session in whatever way makes most sense to their web site. The only thing CEF expects is that both the portal and the remote site makes the session creation, validation and information about the session available to CEF

3. Does CEF use cookies? If it does then what about sites that do not use cookies for session management. How do they interface with other CEF based sites?
Ans> Most sites use cookies to maintain session id information across user requests in the browser. In our sample implementation of CEF we do use cookies. But again, when CEF needs to create a session or validate

a session or get information from a session, it just invokes a local module on the web site. This module, written by the local web site developers is the one that knows how to manage sessions for their applications
In general besides cookies, URL rewriting and hidden frames are two other common mechanisms used for session management.
In URL Rewriting, all urls are dynamic and go through a centralized gatekeeper type module on the way out or in from the web server. This gatekeeper module (sometimes even implemented as an apache API module plugged into the web server) is responsible for appending the session information to the urls being sent out in the response to the browser. When a request comes in from the browser, these modules then extract this session information from the URL and convert them into normal urls for the actual processing by the web site. Of course this technique does not work if URLs are generated dynamically using JavaScript or other mechanisms. But it works for most static content websites
Using Hidden frame variables is another approach. In this case, the website content contains a hidden frame in all WebPages that contains the session information. This frame information is then extracted and sent over with every request to the web site by some JavaScript code embedded in the web content. The problem with this approach is that all the content of a web site needs to be retrofitted to support this approach.
Of the three approaches, cookies are the lowest cost and most efficient, especially since most browsers these days are equipped with cookies. However this is not true for other devices like WAP Enabled phones and browsers. In this case CEF does support URL rewriting