Caucus 4.0 Press Release

Who We Are
Caucus
On the Porch
Screen Porch

Screen Porch rolls out Caucus 4.0, the latest update to the premier Web-based conferencing software.

Woodbridge, VA, March 10, 1998—Screen Porch LLC today announced version 4.0 of Caucus, the world's leading independent Web-based conferencing so ftware. Available immediately, Caucus 4.0 can be downloaded for a free 30-day trial from the Screen Porch site on the Web (http://screenporch.com).

Built on solid conferencing technology proven over a period of ten years, Caucus 4.0 facilitates interactive discussion conferences where people can work to gether with all kinds of Web-based information: from HTML documents to databases, Java applets, images and audio.

Jerry Michalski, a computer industry analyst and managing editor of *Release 1.0,* says "Caucus delivers a key benefit—active collaboration— which organizations hope for but often don't get from shared databases like Lotus Notes or news server interfaces like Netscape Collabra—yet Caucus comple ments Notes and adds substantial value where that platform is already in place."

On the client side, Caucus-hosted meetings and discussions require only a common Web browser. On the server side, Caucus runs on any Windows NT or UNIX ho st.

As a conferencing platform, Caucus differs in three fundamental ways.

First, it minimizes "topic drift" with a format that keeps discussions to just two levels: conference items and participants' responses within those items. Participants can add plain text or HTML. They can also upload files of any sort, including Web media like audio and images. Users can review and edit their contributions. And they can include live links to Web documents or other Caucus discussions.

Second, it allows an enormous range of customization. An enterprise can easily create a virtual workspace consisting of a number of conferences, and manage the knowledge that grows in that space. Users can build knowledge notebooks using simple Caucus tools. Organizations can furnish workspaces with online book stores and other collections of internally-published documents or external links. The visual design of a Caucus workspace can be customized extensively using browser-based forms or standard HTML development tools. Organizations can also create interactive applications, using the Caucus Markup Language (a superset o f HTML). And when a project is finished, its Caucus workspace can be archived as a knowledge base and re-used as a template.

Third, Caucus integrates easily with a company's existing Web and intranet technologies. It turns static Web documents into dynamic material for collaborat ion and it works with a company's other Web applications. Caucus comes with a complete set of self-installing templates, so organizations only

Who We Are

Caucus Success Stories

Licensing and Sales

Job Opportunities

Contacting Screen Porch

- **Latest News**

For more information, contact:

Jan Searls
Screen Porch, LLC
703-578-3436
searls@screenporch.com
http://screenporch.com

Tom Mandel
Screen Porch, LLC
202-362-1679
tmandel@screenporch.com

need to add content and users.

"We designed Caucus to be the primary platform for collaboration on the Net," states Charles Roth, Chief Technology Officer of Screen Porch. "Caucus is architecturally simple and compatible with the Web technologies our customers already use. It is open to oncoming Web development, scaleable, programmable, easily integrated with network-based security and authentication—and very easy to maintain."

"Caucus dramatically increases an organization's productivity," adds company President Tom Mandel. "It takes the way people work in the real world—by getting together and discussing an issue—and puts it in the virtual space where more and more real work now takes place. No matter how separated they may be, they can share images, presentations, links and files of all kinds, much as they might in a meeting room. What's more, Caucus helps people keep their meetings in focus and on topic, rather than digressing into distracting threads that unravel all over the place."

"Conferencing will increasingly become a fundamental network service," says Doc Searls, president of The Searls Group and publisher of *Reality 2.0.* "Screen Porch is better-positioned to make that happen than any other company. Unlike Collabra, Caucus doesn't run on just one kind of browser. Unlike Notes, it doesn't require a colossal commitment by the enterprise to a proprietary collaboration platform. Essentially, it's the best social computing application of our time. And also the only one with roots that run back more than a decade to collaboration platforms that predate even the likes of Lotus Notes."

Caucus runs on Microsoft Windows NT and a wide variety of UNIX platforms, including Sun Solaris, HP-UX, Digital (OSF/1), Linux and SGI. Small-group licenses start at $695.00. Full pricing information is available at the company's Web site (http://screenporch.com).

Screen Porch is a privately held company that was founded in 1996, and is headquartered in Northern Virginia. It also has offices in California and Michigan, and distribution partners in the United States, Canada, and Japan. Commercial users of Screen Porch software include enterprises of every size, from British Telecom and Avery Dennison, to consulting firms like Metasystems Design Group and Community Intelligence Labs. Education customers include dozens of colleges, universities and other schools. Government customers include the U.S. Department of Defense, the City of Santa Monica, and many other bodies.

Who We Are Caucus On the Porch Screen Porch

Who We Are

Screen Porch was founded in 1996 to create software for teams and learning groups online. Our founders combine more than fifty years experience developing online teaming and learning environments for organizations worldwide.

Our first product Caucus is groundbreaking software. Based on the world's best and longest-established computer-conferencing technology, Caucus is already used worldwide as a focal point for critical collaboration—to support virtual teams, virtual campuses for distance learning, and virtual conference centers for communities of practice.

For more information about Screen Porch, see the company backgrounder.

Please email sales inquiries to: sales@screenporch.com

- **Who We Are**

Company Backgrounder

Caucus Success Stories

Licensing and Sales

Job Opportunities

Contacting Screen Porch

Latest News (Updated: Jun 19)

Corporate Backgrounder—Screen Porch, LLC

Who We Are Caucus On the Porch Screen Porch

Screen Porch was created in 1996 based on a strategic vision of how people work online, and how software can best support this work. The Screen Porch team brings together people with a long history of innovation, commitment and success in creating online software and environments for teamwork, learning and community.

Company CTO Charles Roth has been an innovator in online collaboration software for nearly twenty years, during the last fifteen years of which software he created has been continually in use by communities, teams, and educational environments online. He created technologies for computer conferencing software in the early 1980's—before there was anything called the Internet, and before the Web had ever been thought of. Initial prototypes of Caucus technology for online collaboration and community were created more than ten years ago and were tested and refined supporting communities of Caucus users directly connected to UNIX host computers. The innovative and proprietary Screen Porch technologies embodied in today's Caucus reflect the depth of that experience while leveraging and advancing the capabilities offered by the Web.

Metasystems Design Group, Inc. (MDG), whose principals were on the founding team of Screen Porch, has nearly fifteen years experience in building online virtual communities and supporting virtual teams online. MDG consults with government agencies, Fortune 1000 companies, and other organizations around the world, helping them use online collaboration to enhance their capacities for organizational development and learning. MDG also hosts The Meta Network—home to hundreds of online organizational and individual development learning programs as well as to many thousands of public online community participants.

CEO Tom Mandel is a writer with a world reputation and an enterpreneur with fifteen years of experience in the high tech industries and a decade of experience on the Internet. His previous ventures have included eye-to-i, a Website development firm which created sites for Washington DC organizations; TMA/One-Net, a consulting and systems integration firm aimed at shaping and executing programs to integrate the Internet and the Web in the operations of businesses and other organizations; and SMI (Spiegelman, Mandel Interactive), a pioneer in electronic design in the San Francisco Bay Area.

Screen Porch exists to provide organizations with powerful, effective software that is flexible and extensible yet simple to use, so that groups of all sizes can work together online without barriers either to capability or access, and individuals are able participate at any time, from virtually anywhere in the world, using virtually any kind of computer workstation.

For additional information about Screen Porch, please contact us:

Phone:	703-243-3001
Fax:	703-385-3209

Who We Are

- **Company Backgrounder**

Caucus Success Stories

Licensing and Sales

Job Opportunities

Contacting Screen Porch

Latest News

Email: info@screenporch.com

introducing Caucus 4.0

Who We Are Caucus On the Porch Screen Porch

Caucus has always provided the world's best online platform for collaborative projects of all kinds.

Caucus 4.0 builds on that feature-rich foundation with improvements that expand your ability to share and organize the information you need to work and learn with others online.

- A **redesigned interface** with new graphics and improved layout makes Caucus easier to use than ever.
- The new **personal Notebook** feature allows you to store references to any location in Caucus for quick access to the information you need most.
- A **more powerful searching function** enables you to search across multiple conferences to find the information you're looking for.
- Expanded options for **viewing and sorting** conference contents listings provide enhanced tools for browsing Caucus workspaces.
- New tools to **copy and move conference material** from one location to another facilitate sharing and reorganizing information within and between co nferences.
- Improved **conference organizer tools** simplify conference management:
  - New tools for maintaining conference user lists
  - Expanded options for defining conference backgrounds and colors
  - Organizer control over allowed and prohibited HTML tags
  - Flexible limits on when responses may be edited.
- A complete suite of **web-based Caucus management tools** provides powerful access to the full range of manager functions:
  - Conference creation, deletion, and archiving
  - Caucus user creation, deletion, and password management
  - User group management
  - Control over system-wide variables that affect the Caucus interface
  - Server management

Caucus 4.0 is 100% compatible with previous versions of Caucus, so you can take advantage of the new features without interrupting your workflow.

Try Caucus 4.0 now at our public conference center, or download a free trial copy for your own web site.

Product Information

A Guided Tour of Caucus

Demonstration and Support Conferences

Trial Download

Customer Download

Caucus 4.0 Software Downloads

Who We Are Caucus On the Porch Screen Porch

This page is for **registered licensees** of the Caucus software.

(If you are not a licensee, but would like to try out Caucus on your own host, you may download a trial version of Caucus.)

The current release of Caucus is version **4.05**, dated 10 March 1998.

If you have purchased a Caucus 4.0 license and are a registered licensee, or have previously purchased Caucus 3.x and have an active maintenance contract, you should have received an upgrade **id** and **password**. (If you have not received an upgrade id, please contact us at upgrade@screenporch.com.)

If you have your id and password, you can immediately upgrade your site by downloading Caucus 4.0 now.

Your Caucus kit includes a brief **readme.txt** file that includes basic installation instructions. But you should also see the full Unix Installation Guide or Windows/NT Installation Guide now!

And don't forget to check out our other Caucus documentation and guides from our documentation page.

Product Information

A Guided Tour of Caucus

Demonstration and Support Conferences

Trial Download

OS & Server Requirements
Caucus License

- **Customer Download**

Caucus Operating Systems and Web Servers

Who We Are Caucus On the Porch Screen Porch

Operating Systems

The Caucus server will interact with compatible web servers (see below) on any of these platforms:

Microsoft Windows NT 4.0

UNIX:
AIX 4.1
BSDI 3.0
DEC UNIX (OSF/1 v4.0B)
FreeBSD 2.2
HP-UX A.09.04
IRIX 5.3
Linux 1.2.13 & 2.0
Solaris 2.5 (Sparc)
Solaris 2.6 (x86)
SunOS 4.1 (Sparc)

Note: Caucus is upward-compatible with newer versions of these platforms and operating systems.

If you are interested in using Caucus on a different platform, please tell us about it.

Web Servers

Caucus works with most HTTP 1.0 compliant web servers available for the platforms listed above. We provide specific installation instructions for the following servers:

NSCA httpd server— http://hoohoo.ncsa.uiuc.edu
Apache httpd server— http://www.apache.org
Netscape Enterprise Server 2.x— http://home.netscape.com
Netscape Communications Server 1.x
CERN httpd Server— http://www.w3.org/pub/WWW/Daemon

Microsoft IIS— http://www.microsoft.com/iis
O'Reilly WebSite— http://website.ora.com

(Note: while we include directions for the CERN server, we specifically recommend **against** using it. Our experience suggests it is too buggy for practical use with Caucus, and we cannot guarantee support for sites using it.)

Caucus will work with other standards-compliant httpd servers, but the person installing the software must understand in detail how to define CGI files or directories and how to set up access authorization controlled

Product Information

A Guided Tour of Caucus

Demonstration and Support Conferences

Trial Download

- **OS & Server Requirements**
Caucus License

directories.

Caucus Documentation

Who We Are · Caucus · On the Porch · Screen Porch

The following documents will help you understand Caucus. Please feel free to download any or all of them. To experience Caucus yourself, please visit our Conference Center.

Caucus FAQ
A collection of "Frequently Asked Questions" about Caucus.

Caucus Installation Guide (Unix)
Full details on how to install the Premier Conferencing System on the Web.

Caucus Installation Guide (Windows/NT)

Caucus Installation "readme.txt" file (Unix)
The quick-start "read me" file for installing Caucus. Included with each Unix Caucus kit.

CML Reference Guide
The Caucus interface is written in CML, the Caucus Markup Language. Here are the nuts and bolts of how CML works.

Caucus Architecture Description
A detailed description of how Caucus actually works with your web server.

Conference Organizer's How To
The person in charge of a Caucus conference is called the **organizer**. This guide details how to start and run a conference.

Guide for Conference Organizers
This guide talks more about the **why** of organizing the conference, and the human issues involved.

Caucus & Virtual Hosting
This page describes how to implement different Caucus interfaces for different "virtual hosts".

Caucus E-Mail Interface
This page describes how add e-mail only participants to an on-going Caucus conference.

Caucus 2.7 **(Text Interface) Documents**
An earlier version of Caucus supports several text-based interfaces. You can now get the guides for the text interface software:

- *Caucus 2.7* User's Guide (HTML).
- *Caucus 2.7* User's Guide (RTF).
- *Caucus 2.7* Menu User's Guide (RTF).
- *Caucus 2.7* Installation Guide (RTF).
- *Caucus 2.7* Customization Guide (RTF).

Product Information

The Caucus Server
The Caucus Center
Caucus Markup Language
- **Documentation**

A Guided Tour of Caucus

Demonstration and Support Conferences

Trial Download

Caucus FAQ

This is the *Caucus* **FAQ** (Frequently Asked Questions) document.

Caucus is Web-based conferencing software for teamwork, learning and community from Screen Porch LLC.

Follow the links in this document to find the answers to your questions about *Caucus*. If you have a specific question that is not covered, please click on the Ask Screen Porch link on any page.

1. What is *Caucus*?
2. How do I obtain *Caucus* for my Web site?
3. How can I see *Caucus* in action?
4. How do I install *Caucus* on Unix?
5. How do I install *Caucus* on Windows/NT?
6. How do I customize *Caucus* for my Web site?
7. How do I get technical support for *Caucus*?
8. What is the internal architecture of *Caucus*?

What is *Caucus*

[TOP] [Next] [Ask Screen Porch]

Caucus FAQ => What is Caucus?

Frequently Asked Questions (FAQ)

About Screen Porch, *Caucus* and New Ways to Work & Learn on the Web

1. **Who is Screen Porch?**
Screen Porch is a software company with a mission to create innovative environments for social computing – learning, teamwork, and community. We develop and market effective new models, technologies and products for people who work together online.

Screen Porch was founded in 1996 by an experienced team of software, networking, and organizational design professionals. Our proprietary technologies reflect the group's many decades of experience helping people work together using computers.

Out of aggregations of individuals, Screen Porch products create effective groups. Out of information possessed by individuals, Screen Porch products create shared knowledge. Our software is accessed via a browser and can include and integrate any other browser-accessible software.

2. **What is *Caucus*™?**
Screen Porch's first product, *Caucus*, is software for teamwork, group-learning, and community activity. *Caucus* workspaces on the Web allow people to work together at their convenience – across buildings, time-zones, or continents.

People work together in meetings and conversations – in short, in discussions. When we talk about *Caucus* we are simply talking about the world's best discussion software, software which models the ways work actually gets done in real-world discussions.

Caucus creates persistant online workspaces called conferences; they are available any time and offer rich contexts for information-sharing. *Caucus* workspaces are not dependent on marshalling resources and people for a "realtime" session, yet they create the sense that the team, learning community, committee, or other group using the space is actually present whenever an individual arrives to participate.

3. **How does *Caucus* work?**
Caucus combines a feature-rich computer-conferencing API, a workspace-scripting language which is thoroughly HTML-compatible, and facilities to capture and present literally any kind of information in the workspace. *Caucus* workspaces can be tailored to meet any need people have to work together.

   Visually, *Caucus* workspaces can take on any Web-compatible form. *Caucus* pages can be redesigned as easily as any other Web page. Workspace visual identity can easily be made to reflect and extend corporate identity, and a visual identity can be tailored for the application's participants and purpose.

   The templates included with *Caucus* offer immediate productivity, automatically installing an organizational workspace center. But, *Caucus* also includes facilities to develop custom discussion-centered applications, using CML (*Caucus* Markup Language), our HTML-compatible scripting language. These *Caucus* applications can extend from a single *Caucus* Discussion Object^{tm} to a full-blown enterprise-wide conference center.

   Caucus workspaces may include any specialized capability (presentation, multimedia, database, etc.) a customer requires. Not just Web *information* but any browser-accessible application – literally a*nything* which can appear on a Web page – can be included seamlessly in the discussion.

   Workspace resources may include files and documents from users' PCs, applets, diagrams, video, database applications, links to other resources, a Chat room, etc. These resources simply become material parts of the discussion – sources people draw on and information they use to create knowledge while working together.

   Visual customization, programmability, and the power to integrate any Web information or application are the keys to the extraordinarily rich contexts for work created by *Caucus*. Yet, *Caucus* is also easy to afford, simple to integrate, and offers immediate benefits. It adds no infrastructure and no new software to learn. And *Caucus* is compatible with all Web security methods. If an organization's intranet or Internet sites implement security methods that work in a Web environment, they will work with *Caucus*.

4. **What markets does Screen Porch target for *Caucus*?**
Screen Porch has developed *Caucus* to address three core needs of organizations:

   - Learning
   - Teamwork
   - Community

   Learning: Screen Porch focuses on distance-learning in the education market as well as applications for corporate-university and corporate-training markets. *Caucus* is used by numerous colleges, universities and other educational institutions around the world to create virtual campuses.

   Teamwork: Screen Porch aims *Caucus* at Fortune 1000 and government organizations that need support for distributed, virtual teams. *Caucus* has been chosen for these kinds of applications by some of the largest government and commercial entities in the world.

   Community: *Caucus* is used to support virtual communities around the world. Increasingly, it is being used to create organizational communities of skill and practice intranet/extranets in support of enterprises and products/services.

   On a longer view, virtually every Web site will benefit when its constituencies can create and share knowledge and community at the site, rather than simply viewing interactive billboards or querying static information. Every Web site is a *Caucus* conversation waiting to happen.

5. **What support issues should an organization anticipate when considering *Caucus*?**

Caucus features the affordability, ease of use, simplicity of deployment and low training cost associated with a browser-based environment. Software maintenance is of a single server application only.

Caucus can be installed and a full conference center created in a single day, allowing an organization to begin to recapture quickly the cost not only of the software but of much of their Web infrastructure. Moreover, the business benefits of *Caucus* are available immediately upon creation of conferences.

Caucus offers exceptionally rich functionality for creating and sharing information, building team and group unity and effectiveness, and turning information in organizational knowledge. Along with increased productivity and accuracy through shortened information-sharing cycles, some key benefits of *Caucus* include enhanced collaboration, a heightened sense of team and enterprise participation, and the opportunity to create branded constituencies and communities.

6. **What Operating Systems does *Caucus* support?**
   Caucus runs on Microsoft NT 4.0 (Intel) and a wide variety of Unix servers, on an intranet or the Internet. See below:

   AIX 4.1 (IBM RS/6000 series)
   BSDI 3.0 (Intel)
   DEC UNIX OSF/1 4.0 (Alpha)
   HP-UX A.09.04
   IRIX 5.3 (Silicon Graphics)
   Linux 1.2.13 (Intel)
   Solaris 2.4 (Sun)
   SunOS 4.1 (Sun)

   Caucus is accessed from any standard Web browser, such as Netscape Navigator (2.0 or higher) or Microsoft Internet Explorer (3.02 or higher), on any browser-supported client platform. (viz. any Windows version, Macintosh, UNIX).

7. **How is *Caucus* configured?**
   Caucus software is installed on a standard Web server. It includes the *Caucus* server and database, the HTML-compatible interpreted language CML (*Caucus* Markup Language), and *Caucus* workspace templates.

8. **Who should install *Caucus* at your site?**
   The Webmaster, network administrator, or systems operator who manages your Web server should install and configure *Caucus*. It may be managed by your Webmaster. Users throughout your organization will be able to organize and administer *Caucus* workspaces.

9. **Who should do customization and application development using CML?**
   Any HTML developer, using any HTML tools, can create custom designs for *Caucus* and *Caucus* workspaces and can also integrate any HTML software (viz. video, audio, chat services, etc.) into your *Caucus* environment. Anyone with experience in programming concepts and logic – including people who have used Perl, Visual Basic, or any other scripting or programming language – will find CML easy to use to create a wide variety of discussion-based applications.

10. **How does *Caucus* differ from Lotus Notes?**
    Despite increased Web services and browser accessibility, Notes is a soup-to-nuts integrated environment based on proprietary network protocols, servers and client software. Notes requires an organization to "bet the company" on the software.

    Caucus is an incremental application on an intranet, extranet, or Internet Web-server. *Caucus* workspaces can include literally anything that can appear on a Web page – including any Notes application that can be accessed from a Web browser, or any other browser-accessible database application. It is designed to interoperate smoothly with all other Web-compatible software. These and other resources are transformed by being brought into the *Caucus* context.

    Caucus is also much easier to install, configure and maintain than Notes, and it is much less expensive than Notes.

11. **How does *Caucus* differ from integrated Web groupware environments or "Notes-clones?"**
    It's a matter of architecture, and even more of a design philosophy.

The Web is a rapidly developing environment, a world in which creative people of high intelligence introduce new tools and technologies regularly, presenting important opportunities for organizations to gain productivity and effectiveness by continuing to enhance their intranets and Internet sites. This kind of power and flexibility will simply never be available from within "soup-to-nuts" integrated systems.

Hence, we designed *Caucus* not as a closed system or "integrated" package, but rather as a high-performance application for discussion-centered workspaces that can *include and integrate* any Web-standard software – software that exists now and software to come. We also gave *Caucus* a powerful application-development scripting system to allow customers to extend their workspace applications.

Caucus is, and will continue to be, compatible with and friendly to new Web technologies as they are developed. *Caucus* allows an organization to gain advantage by integrating these new tools and technologies in discussion-centered environments where information can be created and shared. *Caucus* is also compatible with any kind of intranet or other Web-compatible security technologies or measures.

12. **How does *Caucus* differ from Web BBS software, newsgroups and "threaded" forum products?**
In BBS, newsgroup, or threaded forum software, the individual message – not the meeting, discussion, project or team – is the most significant object. This software relies on a metaphor taken from a real-world bulletin board, as the name implies. The metaphor is of an individual posting a message in order to receive a reply from another individual.

    Essentially such software creates a database of individual messages. As with any other database, the goal is either to quickly retrieve information and remove it from context or to quickly add specific information to the database. When a user selects a message, the screen blanks and is taken over by that individual message.

    There is no context, no persistent sense of place, and no room to structure an application. It would not be easy, for example, to support a strategic team or create an online training course in a BBS. Such software provides no foundation for creating discussion-centered workspaces or applications to support teamwork, learning groups, or organizational communities.

    Moreover, because these products rely on the basic metaphor of thumbtacking a query to a bulletin board in hope of an individual reply, they all rely on threading and subthreading as a way to organize their message databases. The goal is to create a way to match reply to query individually.

    Threading and subthreading are antithetical to the creation of a team and team knowledge. Instead of focusing contributions to add to a rich shared space, they disperse replies in multiplying threads which get thinner and thinner.

    Caucus has been designed based on decades of experience in discussion-centered application development. *Caucus* workspaces are rich networks of people, resources, applications, interests, identities, databases, and multimedia information – all designed to support and enhance group commitment and achievement.

13. **How does *Caucus* differ from "Web conferencing" products?**
Caucus has been designed to allow an organization to build custom discussion-based applications. Any such set of workspaces can have its own look, feel, and custom-designed interface. No Web conferencing product allows this kind of advanced customization or application development.

    Particular *Caucus* discussions – and even items within these discussions – can be attached as objects to anything whatever on a Web page, referenced as normal URLs. No Web conferencing product was conceived with this immensely useful facility in mind.

14. **How does *Caucus* differ from Netscape Collabra?**
Collabra essentially offers an interface to NNTP servers, an outdated technology which implements BBSes on the Internet. Moreover Collabra requires that an organization use Netscape's server products and a Netscape browser environment. Like Lotus Notes, it locks an organization into a proprietary and single-branded server choice, even though other Web server software may offer benefits in price, performance,

or flexibility. *Caucus* will work well with Netscape Web servers and browsers, but will also work equally well with any other Web server. Unlike Collabra, *Caucus* is also browser-independent.

15. **How does *Caucus* differ from Microsoft's offerings?**
Caucus is available in a version for Microsoft NT. *Caucus* is 100%-compatible with Microsoft's Web development standards and tools, including ActiveX. These facilities can only enhance the kinds of functionality available within *Caucus* workspaces.

16. **How does *Caucus* differ from Java? Other Web development tools?**
Caucus applications are compatible with Java-driven Web pages, and Java applets can be included in *Caucus* discussions and even in user's responses if a site wishes to allow this inclusion. Web development tools, whether for publishing, database access or other programming purposes, do not provide facilities for creating rich discussion-centered workspaces. They offer complementary functionality, rather than competition, for *Caucus*.

17. **Will VARs and Web site developers create *Caucus* applications and *Caucus*-enabled Web sites for their clients?**
Yes, they are already doing so. We will be happy to share solutions stories with interested people.

18. **How does Screen Porch distribute *Caucus*?**
Caucus is sold through direct sales, via the Internet, and through VARs and consultants/developers.

19. **Is *Caucus* currently available? On the Web?**
Caucus is currently available. A 30-day trial version of *Caucus* is available for download from Screen Porch's public Web site at http://screenporch.com. Product information and a demonstration *Caucus* workspace are also at the Web site.

20. **What future developments are in store for *Caucus*?**
Caucus will continue to be enhanced. A new version of *Caucus* will be released by early 1998. New CML templates and sample applications are often available in our technical support conferences, created either by us or by *Caucus* developers worldwide.

21. **How can I learn more about *Caucus*?**
For more information, visit our Web site at http://screenporch.com, send email to the Screen Porch sales department at sales@screenporch.com, or call Screen Porch sales at 703-243-3001.

Installing Caucus 4.0 on Unix

Screen Porch

This guide, from the Screen Porch Documentation Library, describes the details involved in downloading and installing Caucus on your Unix system. If you run into a problem or question not covered in this page, please join the appropriate support conferences on our Caucus site at http://screenporch.com.

1. **System requirements to install Caucus.**
2. **Downloading a Caucus kit for Unix.**

   If you'd like to try Caucus on your own Unix host, you can download a 30-day trial kit.

   If you have purchased a Caucus license, you can download your official kit now. You will need the download id and password sent to you by Screen Porch. (If you did not receive a download id, please contact sales@screenporch.com.)
3. **Installing your Caucus kit.**

   Your Caucus kit is downloaded as an compressed "tar" archive called **caucus40.t.Z**. (If you've downloaded a 30-day trial kit, you will also receive an "activation key" via e-mail. You'll need this key to activate your trial kit.)

   1. Before you begin
   2. Create the Caucus Userid
   3. Installation Procedure
   4. Starting the Caucus daemon
4. **Configuring your Web Server**

   - General Server Configuration
   - NCSA or Apache server Instructions
   - Netscape Enterprise Server 2.x Instructions
   - Netscape Communications Server 1.x
   - CERN Server Instructions
5. **Accessing Caucus**

   The installation procedure creates a standard HTML page that you can use to access Caucus. The URL for that page is:

   > http://**yourhost.com**/~**caucus**/caucus.html

   where **yourhost.com** and **caucus** are the hostname and Caucus userid.

   This file is located in the public_html directory of the "caucus" userid. It is just a template for accessing Caucus. If your organization already has a set of web pages, you will probably want to integrate this file with your existing pages. You might choose to copy the links in this file to the appropriate places on your existing web pages; or you might decide to edit the caucus.html file and simply make it look more like your other web pages.
6. **Managing Caucus**

During the installation procedure, you were asked for the userid (in Caucus) of the "primary manager". That person is given full management capabilities of your Caucus site, including the ability to add other people as managers.

To see the management capabilities, access Caucus with that userid, and on the "Caucus Center" page, click on "You can manage this site" (underneath your name near the top right corner of your browser window).

If for some reason you need to change the userid of the primary manager, edit the plain text file MISC/managers (from the Caucus home directory), and change the userid in the very first line of that file.

7. **Upcoming Information about Caucus**

   Several enhancements are planned for Caucus, and Caucus documentation, in the near future. These include precise web server configuration instructions for Netscape Enterprise Server 3, a "how-to" on using other authentication methods for Caucus, a description of the new Caucus "macro" features, and a detailed FAQ about installation problems and solutions.

   Stay tuned to this site for further information!

Before You Begin

Screen Porch

Normally, *the system manager must install Caucus.* Ordinary users do not have the proper permissions to install Caucus and integrate it with the primary web server.

(It *is* possible to install Caucus and a web server in a single, regular userid on a Unix platform -- for example, as a way to evaluate a trial Caucus kit. But a production Caucus system should always be installed by a system manager.)

If you are **upgrading** a Caucus site, we recommend that you have or make an adequate backup of the Caucus home directory. Then skip directly to Install the Caucus Software. Your Caucus software will be updated without harming any of your existing conferences. If you are installing Caucus for the first time, follow all of steps shown below.

The installation procedure installs the Caucus database, the World Wide Web interface, and the text interface. The normal Caucus 4.0 kit is licensed for unlimited number Web users, but limits the text-interface users to one at a time. If you have purchased the text-interface license option, you will also be allowed an unlimited number of text interface users.

(If you find that you need assistance with installing Caucus, start by joining the appropriate support conferences at http://screenporch.com.)

Create the Caucus Userid

Screen Porch

Create a new userid, called "caucus", with its own home directory, such as **/home/caucus**. (You may use a different name or home directory if you prefer. The installation procedure will adapt to whichever name you choose.)

The home directory for Caucus must have enough free disk space to contain all of the Caucus programs and data files, and all of the anticipated conference data. A minimum of 100 megabytes is recommended. (The software itself is less than 20 megabytes.)

Only the system manager, or a designated Caucus manager, should know the password to the Caucus userid.

Installation Procedure

The software installation procedure is the same whether you are:

- installing Caucus for the first time
- upgrading your existing Caucus software

The installation procedure automatically determines if this is a new installation of Caucus or an upgrade to an existing Caucus site. If you are upgrading Caucus, your existing conferences will not be affected by the upgrade. (This includes upgrading from a Caucus Trial Kit.)

1. **Login to the Caucus userid**.

   Unless otherwise stated, all commands in this installation guide must be typed while logged in as "caucus".

2. **Cease using Caucus and shut it down**.

   If you are upgrading Caucus, *all* Caucus users should exit or quit the program while you are performing the upgrade, and the program should be shut down.

   You can shut down Caucus from the Caucus Management page (from within Caucus), or by typing:

   ```
   stopcaucus 0 0 /home/caucus
   ```

   where **/home/caucus** is the home directory of the Caucus userid.

3. **Unpackage the software**

   The Caucus software is delivered in a file called **caucus40.t.Z**. In the Caucus home directory, type the following command to unpackage this file.

   ```
   zcat caucus40.t.Z | tar xvf -
   ```

4. **Run the installation script**

   The software includes an installation script that will automatically create the proper script files, set the proper file permissions, and so on.

   The script will ask for the hostname (and port number, if needed) of your web server. Be prepared to provide these. To run the script, type:

   ```
   ./cinstall
   ```

5. **Messages & Warnings**

   The cinstall script will produce some warnings and informative messages on your screen. A copy of these warnings is also placed in the file **caucus.warn**. These warnings should be self-explanatory. They fall into three categories:

   - Information about specific files, paths, or URLs. For example, cinstall tells you the full URL for accessing Caucus.

Warnings about new versions of old files (in case you are uggrading from a previous version of Caucus. For example, cinstall creates a new SWEB/swebd.conf file, and renames your old swebd.conf file.

- Warnings about upward compatibility. For example, if you have conferences created in an old text-interface version of Caucus (prior to version 2.6), you will be warned to run the "fixdate" script on your conferences.

You should examine these warnings carefully and determine if any of them apply to your Caucus installation.

6. **Check hostname and port number**

   When you ran the cinstall script, it asked for a hostname and port number (such as "www.xyz.com" or "host.mycompany.com:8001"). If you need to change this information, now or at any future time, edit the files listed below, and change the hostname or port number appropriately.

   - SWEB/swebd.conf
   - SWEB/start.cgi
   - public_html/caucus.html

7. **Public HTML directory**

   Caucus requires that certain files be placed in the Caucus userid's public HTML directory. The standard name for this directory is **public_html**. The Caucus distribution includes a public_html directory with the necessary files already in it. If your httpd server uses a different name, rename public_html to that directory name now.

   For example, if your httpd server uses "WWW" as a user's public HTML directory, from the Caucus home directory type:

   ```
   mv public_html WWW
   ```

   You may also need to change the definition of the Caucus parameter "Caucus_Lib". See your SWEB/swebd.conf configuration file for details.

8. **Text Interface**

   In addition to the Web interface, there is a (largely historical) text interface to Caucus. If for some reason you wish to use this text interface, see the files **cv2** and **cv2check**. Cv2 is the script used to run the text interface to Caucus. Cv2check provides a quick summary of how much new information there is in the conferences that you belong to.

   If you have purchased the unlimited text-interface license option for Caucus, and are providing access to the text interface to your users, you probably want to copy these scripts to a public directory, such as /usr/local or /usr/local/bin. You may also wish to rename the scripts to something more mnemonic, perhaps **caucus** and **caucuscheck**.

Starting the Caucus daemon

Caucus runs as a single master "daemon" program called **swebd**, which accepts requests for new Caucus sessions. It spawns off a "sub-server child" called **swebs**, one for each user's session. (See the Caucus architecture description in the Documentation Library for more information.)

Swebd is normally started from **root**, so that it may start as many children as needed. Each swebs child runs as effective userid "caucus", and real userid "nobody".

> **Note:** if you do not have a "nobody" userid, or if the user number of "nobody" is negative, you should create a user with no rights (say, "noone") and a postive user number. Then edit ***/home/caucus***/SWEB/swebd.conf, and change the "Real_ID" line to use the "noone" userid.

To start the Caucus daemon, login as **root**, and type the commands below:

```
rm -f /home/caucus/SOCKET/sweb
rm -f /home/caucus/SOCKET/sweb0*
/home/caucus/SWEB/swebd /home/caucus/SWEB/swebd.conf
```

where ***/home/caucus*** is the home directory of the "caucus" userid.

You must also add these same commands to your system start-up file (such as /etc/rc.d/rc.local, or whatever it is called on your host) so that the Caucus daemon will automatically start when your system reboots.

> **Note:** if you are just testing Caucus, or if your system policies discourage running outside software as **root**, you may instead start Caucus logged in as the "caucus" userid. On most Unix systems, however, this will limit the number of simultaneous Caucus sessions.

General Unix Web Server Configuration

Screen Porch

This section describes, in the abstract, the changes that must be made to your web server configuration to make it work properly with Caucus. Subsequent sections describe the precise details of these changes for the servers listed above.

Read this section to learn **what** you need to change; then skip to the section for your web server to learn **how** to make those changes.

1. **Define CGI directories**

   Caucus uses several different CGI programs in the directories **SWEB** and **REG** to communicate with the web server. The best way to identify these programs to the web server is to declare SWEB and REG as CGI directories.

   Specifically, declare the following mappings of URLs to CGI directories:

   ```
   http://yourhost.com/sweb/  to /home/caucus/SWEB/
   http://yourhost.com/reg/   to /home/caucus/REG/
   ```

   where **/home/caucus** is the Caucus home directory and **yourhost.com** is the hostname (and port number, if any) of your server.

   If for some reason you cannot declare a CGI directory, enable your server in some other way to treat the files:

   - **/home/caucus**/SWEB/swebsock
   - **/home/caucus**/REG/swebsock

   as CGI programs.

2. **Define special "/caucus" URL**

   Caucus users who have already registered a userid may go directly to specific conferences, items, or responses through the special URLs shown below:

http://yourhost.com/caucus	("Caucus Center" page)
http://yourhost.com/caucus/conference_name	(conference home page)
http://yourhost.com/caucus/conference_name/item	(go to that item)
http://yourhost.com/caucus/conference_name/item/response	(go to that response)

   In order to make these special URLs work, the web server must be configured to map URLs that begin "http://yourhost.com/caucus" to the CGI file **/home/caucus**/SWEB/start.cgi. If this is not possible for your web server, your users may still access Caucus through the regular caucus.html page.)

3. **Restrict Access with userids and passwords**

   Caucus' security requires that each user be identified by a unique userid and password. Caucus uses the standard web "access authorization" protocol to implement userid and password checking.

   To enable access authorization for Caucus, you must declare that the directory /home/caucus/SWEB is protected by a userid and password database file. For some web servers, this is done automatically by the Caucus installation script. See the section for your web server to learn whether this is done automatically for the web server you are using.

Netscape Communications Server Configuration

Screen Porch

This section describes the precise details of configuring the Netscape Communications Server, version 2.x, to work with Caucus. (This server is now obsolete, but may still be in use in many locations.) It assumes that you have already installed your web server and are generally familiar with server configuration.

(Throughout this page, wherever you see **/home/caucus**, replace it with the actual path of the Caucus home directory.)

1. **Define CGI directories**

   From the Netscape server manager, under the section **CGI and Server Parsed HTML**, choose Specify a directory that will contain CGI programs only.

   In the **URL prefix** box, enter "sweb". In the **CGI directory** box, enter "**/home/caucus**/SWEB". Make those changes.

   Repeat the same process for the **URL prefix** "reg", **CGI directory** "**/home/caucus**/REG".

2. **Define special "/caucus" URLs**

   Repeat the process for the **URL prefix** "caucus", **CGI directory "/home/caucus**/SWEB/start.cgi".

3. **Restrict Access with userids and passwords**

   To ensure full compatibility with Caucus, you must manually create a Netscape user database sub-directory that is owned by the Caucus userid. Assuming that your Netscape server is installed in **/var/ns-home**, and that the Caucus userid is called **caucus**, type the commands below. (Note that you must know the "root" password in order to do this.)

   ```
   su -
   cd /var/ns-home/userdb
   mkdir caucus
   cp /home/caucus/caucus_passwd caucus/passwd.pwf
   chown -R caucus caucus
   exit
   ```

   If you wish to allow your users to change their own passwords or self-register their own userids, you must also edit **/home/caucus**/SWEB/swebd.conf, and change the parameter "PW_Path" so that the line reads:

   ```
   PW_Path  /var/ns-home/userdb/caucus/passwd.pwf
   ```

   From the Netscape server manager, under the section **Access Control and Dynamic Configuration**, choose Restrict access to part of your server through authentication.

   The **Restrict Access** page should say "You are currently modifying the directory **/home/caucus**/SWEB/*". If it says "You are currently modifying the entire server", then choose Browse Files. In the **Choose a directory to list from** box, enter "**/home/caucus**/SWEB".

   Select the radio buttons for **Only list directories**, and **Follow sym-links**. Make those changes. If there is a link for **Choose this directory**, select it.

   Return to the **Restrict Access** page. It should say that "You are currently modifying the directory

/home/caucus/SWEB/*".

In the **Which Database?** box, select or type "caucus/passwd". Leave the **Which users?** box blank (unless you have a reason to restrict Caucus to specific set of users).

In the **Realm** box, type "Caucus". Make these changes.

Back at the server manager page, under **Server Control**, choose Start up, restart, or shutdown your server, and press the **Restart!** button.

To add userids, Caucus managers may select "Manage Individual User Accounts" from the Caucus management page. Or your users may self-register their own userid and password from the link in the caucus.html page. (Or you may use the **Database user manipulation** section of the Netscape server manager pager.)

NCSA and Apache Server Configuration

This section describes the precise details of configuring the NCSA or Apache web server to work with Caucus. It assumes that you have already installed your web server and are generally familiar with the server configuration files.

1. **Define CGI directories**

   Find the httpd configuration file **srm.conf**. Edit it, and add the lines:

   ```
   ScriptAlias  /sweb/   /home/caucus/SWEB/
   ScriptAlias  /reg/    /home/caucus/REG/
   ```

   (replacing **/home/caucus** with the home directory of the Caucus userid on your system).

2. **Define special "/caucus" URLs**

   Also in srm.conf, add the lines:

   ```
   ScriptAlias  /caucus/ /home/caucus/SWEB/start.cgi/
   ScriptAlias  /caucus  /home/caucus/SWEB/start.cgi
   ```

3. **Restrict Access with userids and passwords**

   Access authorization for NCSA and Apache servers is set up automatically by the Caucus installation script. It creates the file **/home/caucus**/SWEB/.htaccess, which declares that the directory is password-protected. That file points in turn to the userid and password database file **/home/caucus**/caucus_passwd, which is also set up by the Caucus installation script.

   To add userids, Caucus managers may select "Manage Individual User Accounts" from the Caucus management page. Or your users may self-register their own userid and password from the link in the caucus.html page.

Enterprise Server 2 Server Configuration

This section describes the precise details of configuring the Netscape Enterprise Server, version 2.x, to work with Caucus. It assumes that you have already installed your web server and are generally familiar with server configuration.

(Throughout this page, wherever you see **/home/caucus**, replace it with the actual path of the Caucus home directory.)

1. **Define CGI directories**

   From the server configuration page, choose "Programs", sub-selection "CGI directory". Add entries for:

   ```
   sweb/ to map to /home/caucus/SWEB/
   reg/  to map to /home/caucus/REG/
   ```

2. **Define special "/caucus" URLs**

   (It is not known at this writing if this is possible with the Netscape Enterprise Server.)

3. **Restrict Access with userids and passwords**

   In the directory **/home/caucus**/SWEB, create a world-readable file called **.nsconfig**, containing the lines:

   ```
   <Files *>
   RequireAuth userfile=/home/caucus/caucus_passwd realm=Caucus userpat=*
   </Files>
   ```

   From the server configuration page, chose "System Settings", sub-selection "Dynamic Configuration Files". In the "file name" field, type ".nsconfig".

   To add userids, Caucus managers may select "Manage Individual User Accounts" from the Caucus management page. Or your users may self-register their own userid and password from the link in the caucus.html page.

   (Do **not** use the Netscape server user database functions.)

CERN Server Configuration

This section describes some of the details of configuring the CERN web server to work with Caucus. It assumes that you have already installed your web server and are generally familiar with server configuration. (**Note**: Screen Porch recommends **against** the use of the CERN web server. This information is provided here purely for those who wish to, or have some need to, experiment with the CERN server.)

(Throughout this page, wherever you see **/home/caucus**, replace it with the actual path of the Caucus home directory.)

1. **Define CGI directories**

   Find your httpd configuration file, typically **httpd.conf**. Edit it, and add the lines:

   ```
   Exec  /sweb/*    /home/caucus/SWEB/*
   Exec  /reg/*     /home/caucus/REG/*
   ```

2. **Define special "/caucus" URLs**

   In **httpd.conf**, add the line:

   ```
   Exec  /caucus/*  /home/caucus/SWEB/start.cgi/*
   ```

3. **Restrict Access with userids and passwords**

   In httpd.conf, add the lines below:

   ```
   Protection PROT-SETUP-USERS {
      UserId       nobody
      GroupId      nogroup
      ServerId     caucus
      AuthType     Basic
      PasswdFile   /home/caucus/caucus_passwd
      GroupFile    /home/caucus/groups
      GET-Mask     users
   }
   Protect  /sweb/*     PROT-SETUP-USERS
   ```

   Restart your httpd server.

   Change the **/home/caucus**/SWEB/swebsock to be a shell script that invokes the swebsock program by its full pathname. To do this, go to the directory **/home/caucus**/SWEB, and type:

   ```
    mv swebsock swebsock2
    chmod 4711  swebsock2
    echo "#!/bin/sh"  >swebsock
    echo "exec /home/caucus/SWEB/swebsock2" >>swebsock
    chmod  755  swebsock
   ```

Installing Caucus 4.0 on Windows/NT

This guide, from the Screen Porch Documentation Library, describes the details involved in downloading and installing Caucus on your Windows/NT system. If you run into a problem or question not covered in this page, please join the appropriate support conferences on our Caucus site.

1. **System requirements to install Caucus.**

2. **Downloading a Caucus kit for Windows/NT.**

   If you'd like to try Caucus on your own NT host, you can download a 30-day trial kit.

   If you have purchased a Caucus license, you can download your official kit now. You will need the download id and password sent to you by Screen Porch. (If you did not receive a download id, please contact sales@screenporch.com.)

   In either case, you will need the 17+ digit activation key that was e-mailed to you. (If you did **not** receive an activation key, contact sales@screenporch.com.)

3. **Installing your Caucus kit.**

   Your Caucus kit is downloaded as an InstallShield self-extracting archive called **caucus40.exe**. Login as **Administrator**, and double-click on this file to start the installation process. Follow the instructions that appear on your screen.

   Notes:

   - Caucus *must* be installed on a *local* NTFS disk partition. (It will not work on a FAT disk, nor installed on a remote network drive.)
   - If you have previously installed Caucus 4.0, you can install this kit "over top of" your existing installation. Any existing conferences will remain unharmed; just the software itself will be updated.
   - During the installation a command window will open up and display some messages. It will close automatically. You may ignore the messages; they are there only in case the installation fails for some reason, in which case they may provide diagnostic information.
   - The Caucus Service "swebd" program uses TCP/IP port 8023 to communicate with other tasks. This means that you may not install multiple copies of Caucus in different locations on the same server.

4. **Configure your Caucus Service**

   The next-to-last screen in the installation procedure tells you to configure your Caucus Service to "Log On As" the Caucus userid. This must be done manually, through the Control Panel.

   See the service configuration page for details.

5. **Configuring your Web Server**
   1. General Server Configuration
   2. Microsoft IIS Web Server
   3. O'Reilly WebSite Web Server

6. Accessing Caucus

The installation procedure creates a standard HTML page that you can use to access Caucus. The URL for that page is:

> http://**yourhost.com**/~**caucus**/caucus.htm

where **yourhost.com** and **caucus** are the hostname and NT account you supplied during the installation procedure.

7. If Something Goes Wrong...

Should a problem occur, especially during installation, start by checking over each step of the installation procedure. Most problems are caused by missing details in configuring your web server to work with Caucus.

If the problem persists, join our support conferences, especially the **Installation and Setup** conference.

In special circumstances, we may be able to help resolve a problem by "telneting" into your NT host. See installing a telnet server for Caucus for more information.

8. Managing Caucus

During the installation procedure, you were asked for the userid (in Caucus) of the "primary manager". That person is given full management capabilities of your Caucus site, including the ability to add other people as managers.

To see the management capabilities, access Caucus with that userid, and on the "Caucus Center" page, click on "You can manage this site" (underneath your name near the top right corner of your browser window).

If for some reason you need to change the userid of the primary manager, edit the plain text file **c:\caucus\**misc\managers, and change the userid in the very first line of that file (where "c:\caucus" is the home directory of the NT "Caucus" account).

9. Uninstalling Caucus

If you wish to completely remove (uninstall) Caucus from your NT host, note that "Add/Remove programs" in the Control Panel does not work with Caucus.

You may use the following procedure to completely remove Caucus, where **c:\caucus** is the home directory of Caucus, **caucus** is the Caucus account, and *Caucus* is the caucus service name:

1. Start a "cmd" window (click on Start, Run, and type "cmd"). Type the commands that follow in this window.
2. `net stop caucus`
3. `c:\caucus\services\instsrv caucus remove`
4. `net user caucus /delete`
5. `rmdir c:\caucus /s`

10. Upcoming Features for Caucus/NT

Several enhancements are planned for Caucus/NT in the near future. These include detailed web server configuration instructions for later versions of Microsoft IIS, and automatic creation and management of Caucus userids for sites using the O'Reilly "WebSite" server.

Stay tuned to this site for further information!
(Last revised 6 May 1998.)

General Web Server Configuration on NT

Screen Porch

This section describes, in the abstract, the changes that must be made to your web server configuration to make it work properly with Caucus.

If you have one of the web servers listed on the installation page, you can skip directly to the page for that web server.

Otherwise, read this section to understand the kinds of things you need to change or configure, and then apply that to your own web server.

1. **Define public_html mapping**

   Your web server must map the URL http://yourhost.com/~caucus into the directory **c:\caucus**\public_html. This is where some HTML pages, and the images used by Caucus, live. (As usual, you must replace **c:\caucus** with the actual path of the Caucus home directory.)

2. **Define CGI directory**

   Caucus uses some CGI programs in the directory **c:\caucus\**SWEB to communicate with the web server. The best way to identify these programs to the web server is to declare SWEB as a CGI directory.

   Specifically, declare the following mapping of a URL to a CGI directory:

   > http://yourhost.com/sweb/ to **c:\caucus**\SWEB\

   If for some reason you cannot declare a CGI directory, enable your server in some other way to treat the files:

   > **d:\caucus**\SWEB\swebsock.exe
   > **d:\caucus**\SWEB\start.exe

   as a CGI program.

   (Note: in all of the above, replace **c:\caucus** with the full pathname of the Caucus home directory.)

3. **Define special "/caucus" URL**

   Caucus users who have already registered a userid may go directly to specific conferences, items, or responses through the special URLs shown below:

   > http://yourhost.com/caucus ("Caucus Center" page) http://yourhost.com/caucus/conference_name (conference home page) http://yourhost.com/caucus/conference_name/item (go to that item) http://yourhost.com/caucus/conference_name/item/response (go to that response)

   In order to make these special URLs work, the web server must be configured to map URLs that begin "http://yourhost.com/caucus" to the CGI file **c:\caucus**\SWEB\start.exe. (This may not be possible for all sweb servers. Users of such servers can still access Caucus through the regular caucus.html page.)

4. **Restrict Access with userids and passwords**

   Caucus' security requires that each user be identified by a unique userid and password. Caucus uses the standard web "access authorization" protocol to implement userid and password checking.

To enable access authorization for Caucus, you must declare that the directory **c:\caucus**\SWEB is protected by a userid and password database file. The implementation of this varies enormously across the different NT web servers; check your web server's documentation for details.

Configuring the IIS Web Server

This page describes the configuration needed to run Caucus with the Microsoft IIS version 2 Web server. Some details may be different for later versions of IIS.

To use Caucus with your Microsoft IIS Web server, you must make a few changes to the IIS configuration. This should not affect any other use of your IIS web server.

(As usual, anywhere you see **c:\caucus**, replace it with the actual path of the Caucus home directory.)

1. As Administrator, run the Internet Service Manager. (Click Start, Programs, Microsoft Peer Web Services, Internet Service Manager.) Double click the entry for the WWW service.

2. On the "Service" tab, under "Password Authentication", make sure "Basic (Clear Text)" is checked.

3. On the "Directories" tab, Click "Add". This brings up a "Directory Properties dialog box. In the Directory field, type "**c:\caucus**\public_html". Make sure the "Virtual Directory" radio button is checked, and in the "Alias" field type "/~**caucus**". Make sure the "Access" Read checkbox is checked. Click OK.

4. Back at the Directories tab, click "Add" again. In the Directory field, type "**c:\caucus**\sweb". Make sure the "Virtual Directory" radio button is checked, and in the "Alias" field type "/sweb". Make sure that **both** the Access Read and Execute checkboxes are checked. Click OK.

   Exit the Internet Service Manager.

5. Lastly, deny access to the **c:\caucus**\sweb directory to the IIS userid. For example, if your NT machine name is "**XYZ**", you must deny access to the userid (account) IUSR_XYZ. Login to the **caucus** userid (or as **Administrator**), and open a command ("MSDOS") window, and type the command shown below:

   ```
   cacls c:\caucus\sweb\*.* /d IUSR_XYZ /e
   ```

A note about using Microsoft IIS. Due to the nature of IIS, each Caucus user **must** have their own, unique userid (account) on your NT host.

If you would like to support "web-only" userids (that work with Caucus but do not otherwise allow access to your NT host), you must use another web server, such as O'Reilly's WebSite Professional or the *(free!)* Website 1.1. You can run both IIS and WebSite on the same NT host, simply by assigning them different "port numbers".

Configuring the WebSite Server

Screen Porch

The O'Reilly WebSite server comes in two flavors: the souped-up WebSite Professional Web server, and the *(free!)* basic nuts-and-bolts WebSite 1.1 server.

To use Caucus with either web server, you must make a few changes to the WebSite configuration. This should not affect any other use of your web server.

(As usual, anywhere you see **c:\caucus**, replace it with the actual path of the Caucus home directory.)

1. As Administrator, bring up the WebSite Server Properties dialog box. (Click Start, Programs, WebSite, Server Properties.) Click on the "Mapping" tab.

2. In the "List Selector" section, make sure the "Documents" radio button is checked. (If that area is greyed-out, press the "Apply" button near the bottom of the dialog box.) In the "Document URL Path" field, type "/~**caucus**/". In the "Directory" field, type "**c:\caucus**\public_html\". Click on "Add", and then "Apply".

3. In the "List Selector" section, click on the "Standard CGI" radio button. In the "Standard CGI URL Path" field, type "/sweb/". In the "Directory" field, type "**c:\caucus**\sweb\". Click on "Add", and then "Apply".

4. Repeat the previous step, this time with URL Path "/caucus" and Directory "**c:\caucus**\sweb\start.exe".

The rest of this page describes how to set up the userid and password access to Caucus for your Web users.

5. Click on the "Groups" tab, and in the "Authentication Realm" section, click "New". Add a new authentication realm called "caucus". In the "Group" section, click "New", and add a new group called "caucus_users".

6. Click on the "Access Control" tab, and then the "New" button. In the "URL Path" field, enter "/sweb/", and select "caucus" from the "Realm:" pull-down list.

   In the "Authorized Users & Groups" section, click on "Add", select the group "caucus_users", and click OK.

   This step forces the WebSite server to only allow those users in the "caucus_users" group to access the "caucus" realm (and thus any URL containing "/sweb"), and furthermore, insists that each user have a unique userid and password.

7. Finally, create those users. Click on the "Users" tab again, select the "caucus" realm, and click "New" to add a new "web" userid and password for Caucus. (This is the userid and password that each individual user will supply when they begin using Caucus. It is **not** the same as adding a new user account to your NT machine.)

   In the "Group Membership" section, select "caucus_users", and click on "Add".

   Repeat this process for each Caucus user that you wish to add. You may come back to this step at any time to add more users; all of the previous steps 1 through 6 are only performed when you first install Caucus.

Customizing Caucus

[Top] [Next] [Previous] [Ask Screen Porch]

Caucus FAQ => Customizing Caucus

Introduction

This chapter describes some simple changes that may be made to the default Caucus interface. Although you may customize Caucus at any time, the changes described in this chapter would typically be made at the point of installation, as you are considering what Caucus defaults to set, how to customize Caucus to match the visual look of your Internet site or your intranet, and how to give access to Caucus to your users.

Caucus may be customized in two distinct ways. You can customize Caucus by changing system "switches," substituting your own icons, buttons and images for our defaults, inserting your own logo, and by editing the html pages which give access to Caucus. **Changes of this kind are discussed in the present chapter.**

You may also customize the Caucus interface in far more radical and thorough-going ways, including writing your own Web interface, to meet the needs of your organization. The Web interface to Caucus is written entirely in a simple interpreted programming language called CML, the Caucus Markup Language. CML is an extension of HTML, the Hypertext Markup Language, which includes programming constructs (conditionals, loops, etc.) and access to the Caucus database. CML scripts or "pages" generate dynamic "on-the-fly" HTML pages which then appear on a user's Web browser.

Screen Porch distributes a default set of CML pages that make up the standard Web Caucus interface. Each site is welcome to customize these pages, or even to create completely different interfaces. (A single site may have many different interfaces to the same set of conferences.)

For more information on CML, see section 6.4 of this guide and the separate CML Reference Guide. A CML Programmer's Guide is also forthcoming from Screen Porch. It will be available for download from our site. Keep in touch with "News" at the Screen Porch web site (http://screenporch.com) for more information. Customizing Caucus with CML is also under active discussion in the technical support conferences at our web site. Please make use of this important resource.

Caucus Web interface "switches"

Most sites will want to consider and perhaps change several Caucus options. These options are called "switches," because their values either turn a particular option "on" or "off," or else supply a particular value for a string (a file name, for example).

One example: you use a Caucus "switch" to specify the graphic image for the page background at your Caucus conference center.

There are two kinds of Caucus switches: those that control the configuration of the Caucus ("swebd") server program, and those that control options in the CML interface files.

Configuration Switches

The configuration switches are contained in the file **SWEB/swebd.conf**. This is an ordinary ("plain text") file which you can modify with an ordinary text editor. The file contains detailed comments that explain each of the options. **Note**: if you change this file, you **must** restart the Caucus server in order for your changes to take effect.

If you are upgrading your Caucus software, the installation procedure will create a file called **swebd.conf.new**. You should compare this file with your existing **swebd.conf** file, there may be new switches that you will wish to use in **swebd.conf**.

CML Interface Switches

The CML interface option switches are located in the file **CML/SP31/Local/switch.i**. Simply edit this file with a text editor to change the switches. Please see the detailed comments in **switch.i** for a list of the switches and how to set each one.

The default values for all of these switches is kept in the file **CML/SP31/defaults.i**. This enables you to return to the default values at any time by copying this file to **CML/SP31/Local/switch.i**. The values in switch.i override these defaults.

As you install new releases of Caucus, however, new switches may appear in the new **defaults.i** file. You can then copy these new switches to the **switch.i** file and set their values to meet your preferences.

Caucus Icons, Buttons, and Logos

The 'img' directory

Caucus includes a collection of icons, buttons, and logos that are stored as gif and jpeg files. These files are all stored in a single directory for convenience. The name of that directory is set in the file **CML/SP31/startup.i**. Look in that file, and you will see a line that looks something like:

```
set img /~$caucus_id()/GIF31
```

This line defines a CML variable called "img" that contains the location (the directory containing) the gif and jpeg image files. In this case, the image files are stored under the Caucus home directory, under the sub-directory **public_html/GIF31**.

If you wish to change some or all of these image files, the best approach is to create a new directory and put the new image files in that directory. For example:

1. Create a new directory, such as **public_html/GIF4**.
2. Copy all of the files in **GIF31** to **GIF4**.
3. Put your changes in **GIF4**.

   Edit **CML/SP31/startup.i** to point to **GIF4**, as in:

   ```
   set img /~$caucus_id()/GIF4
   ```

   It is <u>**not**</u> a good idea simply to change the images in **public_html/GIF31** directly. Your changes might then be wiped out the next time you install a Caucus upgrade.

Changing Individual Graphics

If you wish to just change a few graphics, it may be easier to create new files (with new names, and put them in **GIF31** with the other graphic files.

In that case, look in the file **CML/SP31/defaults.i**, and you will find the definition of a set of CML variables that reference the pre-defined logos and icons. Copy the definitions of the relevant graphics to **CML/SP31/Local/switch.i**, and then change the definitions there to point to your new files.

Caucus.html

There is one special HTML file that you may also wish to customize. It is located in the **public_html** directory.

Caucus.html is the template access-to-caucus HTML page. It provides links to the conferences, and also to the "self-register a userid" pages. You may wish to completely rewrite this page, or else extract the links and place them in the appropriate places in other HTML pages for your organization.

Changing HTML text in CML files

If you wish to make more extensive modifications to the interface, you can edit the HTML text inside the .cml files in **CML/SP31**. Before undertaking such edits, please copy all the files in this directory to a backup directory, in case you need to restore the originals.

All lines that produce HTML text must begin with a double-quote ("). Everything after the double-quote is HTML, or HTML mixed with CML functions (that eventually evaluate into plain HTML text).

Creating Distinct Interfaces

If you wish to create a very different interface, or wish to provide multiple interfaces (such as your own, and the default SP31 pages), you should create separate interface directories.

For example, to create a new interface called **XYZ** that is loosely based on the SP31 pages, follow these steps:

1. Create the directory **CML/XYZ**

2. Copy everything in **CML/SP31** (including the sub-directory **Local** and its contents) into **CML/XYZ**.

3. Edit the files in **CML/XYZ** to create your new interface.

4. Create a new .html file, based on public_html/caucus.html, to provide a URL that points to your new interface.

Upgrading Caucus

New releases of the Caucus software, and of the default **SP31** Caucus interface pages, will appear regularly on the Screen Porch home pages.

Currently, when you install a new release, the installation script renames your CML/**SP31** directory to CML/**SP31date**, where **date** is today's date.

If you have only made changes to the files under the "Local" subdirectory, or to the graphics, then you should copy everything from CML/**SP31date**/Local to CML/**SP31**/Local.

If you have made more extensive changes to the interface, you have three choices:

1. **Ignore the new release**. Rename CML/**SP31** to something else, and then rename CML/**SP31date** back to CML/**SP31**.

2. **Apply your changes to the new release**. Whatever changes you made before, make them again to the files in CML/**SP31**. This requires that you keep track of the changes that you made. (Or that you kept an unchanged copy of the original **SP31** files, and then you can use a utility such as diff or merge2 to find your changes.)

3. **Apply new features in the new release to your pages**. This is the inverse of choice number 2. Here you would use diff or merge2 to find the new features in the new **SP31** pages, and apply them to your pages.

All of these choices have disadvantages. This is a classic "fork" dilemma in software development, that happens whenever two or more parties add features to an existing product. The best advice is to *always* keep a detailed history of all changes that you make to your site, and to keep a copy of the unmodified **SP31** pages in a separate directory.

Technical Support for *Caucus*

[TOP] [Previous] [Ask Screen Porch]

Caucus FAQ => Technical Support for Caucus

Technical support is available via our *Caucus* Maintenance Agreement or in the form of Screen Porch consulting services.

Caucus **Maintenance Agreement**
Maintenance agreements assure an organization of receiving all *Caucus* upgrades and priority access to Screen Porch technical support. For more information about *Caucus* maintenance agreements and other Screen Porch support products, please contact sales@screenporch.com.

CML Reference Guide

Chapter 1: Introduction

[TOP] [NEXT]

This document is the reference guide for CML, the **C**aucus **M**arkup **L**anguage. CML is a "mark up" language that combines HTML tags with simple programming constructs and database functions. The CML language interpreter is the core of the World Wide Web interface to the Caucus conferencing system (hence the name).

This guide assumes considerable familiarity with HTML, the World Wide Web, Web browsers, and the Caucus conferencing system. For more information about Caucus, see the Screen Porch home page at http://screenporch.com, and the Caucus FAQ at http://screenporch.com/FAQ.

This document is copyright © 1996-98 by Screen Porch LLC. It may not be distributed or reprinted without permission from Screen Porch. This is a work-in-progress, and will be frequently revised. This edition corresponds to the CML interpreter provided with the "Caucus 4.0 beta" package. Corrections and comments (but not questions) should be sent to the author at roth@screenporch.com.

1.1 Incompatible Changes in 4.0

For the most part, CML pages written for Caucus version 3.1 will work properly under version 4.0. There is only one known change in version 4.0 that may cause incompatibilities. The shell functions $shell(), $jshell(), $silent(), and $asynch() now always execute as the **real** userid of the Caucus subserver process (rather than, in some cases in 3.1, as the effective userid).

To run a shell command as the effective userid, use the new $xshell() function.

1.2 What's New in version 4.0

The following functions and features have been added since Caucus version 3.1.

Newitem & response functions

$re_epoch()	response time in seconds
$re_bits()	special property bits of response
$add_resp()	Expanded way to add a response
$add_item()	Expanded way to add an item
$re_copied()	Information about "copied" responses
$re_copier()	Who copied a "copied" response?

Newlogic, math, and string functions

$bit_and()	bitwise and
$bit_or()	bitwise or
$bit_not()	bitwise negation
$hex2dec()	hexadecimal to decimal conversion
$dec2hex()	decimal to hexadecimal conversion
$random()	Produce a random number
$epoch()	Convert date to epoch timing
$timezone()	Difference in seconds to UTC
$dateof()	Convert epoch time to full date
$quote()	quote text as one word
$unquote()	undoes effect of $quote()
$asc2dec()	decimal values of characters in string

Newconference creation and manipulation functions

$create_conf()	create a new conference
$co_remove()	delete an existing conference
$co_makeorg()	add yourself as an organizer to a conference
$set_co_org()	change primary organizer of a conference
$co_rename()	rename a conference
$jshell()	japanese shell output
$xshell()	Comprehensive shell access function

Newfile access functions

$rename()	rename a file
$dirlist()	list contents of a directory
$delfile()	delete a file
$file_data()	check contents of file against a range of values

NewCML directives and language features

Eval	New directive (to replace "set ignore")
Quit	Immediately cease CML page processing
Macros	Dynamically defineable and expandable macros

Newsystem, license, and state information functions

$admin_mail()	Administrator E-mail
$opsys()	host server operating system
$caucus_lib()	Caucus library directory path
$disk_failure()	Did a disk-write error occur?
$cml_path()	CML directory path
$caucus_path()	full pathname of Caucus home directory
$http_lib()	full URL of Caucus library directory
$lice_max_users()	total # users allowed by license
$lice_act_users()	actual # users
$lice_expires()	epoch time when license expires
$all_users()	list of all caucus userids
$new_win()	set size of windows created by $t2url()
$browser_format()	browser language code
$set_browser_format()	set browser language code

Newsorting functions

$gen_sort()	sort arbitrary list of words
$num_sort()	sort arbitrary list of numbers
$triplet_sort()	sort triplet list of item numbers
$item_sort()	sort items by title, author, response date

Newmanagement functions

$mgr_list()	display list of managers and their capabilities
$set_mgr_list()	set list of managers and capabilities
$pw_...()	new password functions
$per_delete()	delete person

Newmiscellaneous functions

$cl_visible()	is conference visible to this user?
$clear...()	clear user/conf/item/site variable caches

CML Reference Guide

Chapter 2: Purpose of CML

[TOP] [PREV] [NEXT]

The Caucus conferencing system was first released in 1986 as a text-based, command driven conferencing (groupware) package. Over the next 8 years, Caucus versions 1 and 2 were extended in a variety of ways that made it extremely customizable -- but still fundamentally text-based.

With the enormous growth of the World Wide Web in 1994-95, it became clear that a Web-based interface for Caucus could greatly increase its ease of use, and its popularity. At the same time, the Web lacked any significant discussion or conferencing tools, and it was clear that a Web interface for Caucus could fill this gap.

Version 3.0 of Caucus was developed from the ground up as web-based client-server system. It was designed to serve Caucus conference information to an HTTPD server, which in turn feeds HTML to any Web browser. But the Caucus server (called "swebs") needs to know what data to serve, and how (what format) to serve it in. This is the purpose of CML.

CML pages (files) are analogous to HTML pages. They contain Caucus directives (e.g., "display the text of such-and-such response) in an HTML-like format. They may also contain embedded HTML.

When a Web user wants to access (or add to) a Caucus conference, s/he points the browser at a special "entry" HTML page. This entry page points to a CML page. (The actual implementation of "pointing to a CML page" is done via the Web CGI standard.) CML pages point to other CML pages, exactly analogous to the way HTML pages point to other HTML pages.

When an HTTPD server gets a request for a CML page, it passes the request on (via CGI) to the Caucus swebs server. Caucus interprets the contents of the CML page, *producing a dynamic HTML page*, and passes it in turn on to the HTTPD server, which sends it to the browser.

(For more detailed information about this process, see the Caucus architecture description in the Caucus FAQ.)

CML Reference Guide

Chapter 3: What's in a CML page?

[TOP] [PREV] [NEXT]

Each CML page (or file) describes a page that will appear on the user's Web browser. (In some cases it just produces an HTTP "Location" directive which points in turn to another CML or HTML file.) CML can be thought of as a superset of HTML. More precisely, HTML is embedded in CML scripts; swebs does not actually understand or parse the HTML codes. A CML page contain five kinds of text:

1. Comments. In the Unix tradition, all lines beginning with "#" are comments and are ignored. Entirely blank lines are also ignored.

2. HTML code. All lines beginning with a double quote (") are parsed for CML functions and macros, but are otherwise passed on to the browser unchanged. (The quote is removed.) There may be leading blanks before the quote; they are ignored.

3. CML functions. Strings of the form $xyz(), $xyz(value), or $(value) are parsed by swebs, and replaced by the appropriate Caucus values.

4. CML macros. Strings of the form %abc(args) are macro invocations, and are expanded by swebs. See CML macro definitions and expansions for more information.

5. CML directives. Directives are like C program code: they describe actions to be taken. Directives start with one of the keywords "if", "elif", "else", "for", "count", "while", "set", "include", "return", "quit", "break", or "end".

A single logical line in a CML file may be broken across several physical lines; a "\" as the last character means "continued on next (physical) line". Most of the time this is not needed, since HTML mostly ignores line boundaries. However, the "\" is useful for assembling long lines that will appear inside HTML <PRE> code, or to improve readability of the CML code.

Here's a sample CML page, typical of a page a Web Caucus user would see early on:

```
#
#---CENTER.CML.   "Caucus Center" Page.
#
#   Overview of (and initial entry to) conferences.
#
#------------------------------------------------------------------------

if $empty ($(href))
   include $cml_dir()/startup.i center.cml
end

set nch $unique()
set nxt $page_save (1 center.cml \
              $arg(2)+$arg(3)+$arg(4)+$arg(5)+$arg(6)+$arg(7)+$arg(8) \
              # $(center_name) )
set last_conf x

#---HTML declaration, header, and BODY tag.
"Content-type: text/html
"
"<HTML>
"<HEAD>
"<TITLE>$(center_name)</TITLE>
"</HEAD>

"<BODY $(body_bg) >
```

```
#---Caucus header.
include $(dir)/header.i

#---Tell the user what this page is about.
"<P>
"<TABLE WIDTH=100% CELLSPACING=0 CELLPADDING=0>
"<TR>
"<TD><FONT SIZE=+1><B>Caucus Center</B></FONT></TD>
"<TD ALIGN=right>
   include $(dir)/youare.i
"</TD>
"</TABLE>

"<P>
"From here, you may go to specific conferences, or
"<A HREF="$(href)/allconfs.cml?$(nch)+$(nxt)+x+x+x+x+x+x">
"see a list of <B>all</B> conferences</A> on this host.
"<P>

#---Prepare to actually put up various kinds of links to the
#   conferences.  Create some variables with lists of
#   conference names.  Apply $cl_list() to the entire list
#   of conferences.
#     L_CONFS are the "popular" conferences.
#     M_CONFS are from the user's personal conference list
set l_confs $file($(inc)/l_confs.i)
set m_confs $user_var($userid() my_confs)
set ignore  $cl_list ( $(l_confs) $(m_confs) )

#---The various ways of getting to the conferences all appear
#   as numbered entries, within one large table.
#   To avoid unpleasant spacing, and because the "JOIN" choice requires
#   being in a <FORM>, the entire table must be inside a <FORM>.
"<FORM METHOD=POST ACTION="$(href)/centerf.cml?$(nch)+$(nxt)" NAME="joiner">

"<TABLE CELLSPACING=0 CELLPADDING=0 >

#---Personal conference list access:
set way_in 1
include $(dir)/cen_pers.i $(way_in)


#---"Popular" conference access:
if $sizeof ($(l_confs))
   include $(dir)/cen_pop.i way_in
end


#---Type a conference name directly:
set way_in $plus ($(way_in) 1)
include $(dir)/cen_type.i $(way_in)


#---See a list of all conferences:
#set way_in $plus ($(way_in) 1)
#include $(dir)/cen_all.i $(way_in)

"</TABLE>
"</FORM>
"<P>

#---Advertisement:
include $(dir)/cen_adv.i
```

CML Reference Guide

Chapter 4: CML Functions

[TOP] [PREV] [NEXT]

CML contains a large number of functions, which provide much of the power of the language. They are grouped by category and described in the subsections listed below:

CML functions serve several purposes:

1. Extract data (from the Caucus conference database) for display.

2. Manipulate or compare data (such as addition, subtraction, testing equality, etc.)

3. Put new data back into the Caucus database.

4. Maintain "state" information between CML pages.

The syntax of a function must always be "$name(arguments)". There must be no spaces between "$" and "name". Spaces may be used freely around the "(" and ")". Anything (including spaces or other functions) may be in the *arguments*. Some functions have no arguments.

If you wish to display a "$" in your HTML text, and not have it be confused with a CML function, escape it with a preceeding "\", i.e. "\$".

CML variables

CML supports one type of variable, that contains arbitrary string data. To get the contents of a variable (called "evaluating" a variable), you use a syntax similar to function evaluation:

```
$(name)
```

This means "evaluate the variable *name*, and place its value here". For more information about variables, see the CML directives "for", "count" and "set".

CML Reference Guide

Chapter 4.1: CML State Functions

[TOP] [UP] [NEXT]

The CML state functions are the glue that ties a group of CML pages and a Sweb server together. To understand more about why they exist, see the design document "The Web Caucus". For the CML author, it is only necessary to understand **where** they must be used.

$host()
: Evaluates to the host name (and http port number) of the current host. This is a useful way to build HTML links that require the current host name, and still keep your CML code portable. Example:

```
"<A HREF="http://$host()/dir/page.html">some text</A>
```

$pid()
: Evaluates to the pid (process id) and security code for the swebs server that is dedicated to your browser. You **must** include this in links to CML pages. Example:

```
"<A HREF="http://$host()/sweb/swebsock/$pid()/SP/test.cml?15+bye">
" name</A>
```

$arg(n)
: Evaluates to the *N*'th argument to this CML page. In the previous example, clicking on link "name" will bring up the CML page test.cml. In test.cml, $arg(1) will then evaluate to "15", and $arg(2) will evaluate to "bye".

$inc(n)
: Evaluates to the *N*'th argument to this "include" file. See the include directive.

$form(name)
: When a CML page is the "recipient" of an HTML form (as in <FORM ACTION="...">), the form data is available through the CML $form function. The function evaluates to the data entered by the user in field *name* (as in NAME="*name*" in an <INPUT> or <TEXTAREA> HTML tag), or (in the case of TYPE="submit" fields) to the VALUE string for the button with NAME=*name*. If there are multiple values for the field *name* (as in a <SELECT MULTIPLE> field), the values are concatenated together, separated by single spaces.

The $form() function transparently handles both standard ("application/x-www-form-urlencoded") and "multipart/form-data" forms. $form() may be used only with METHOD=POST forms.

Multipart forms may be used with some browsers to upload an entire file, with an HTML tag of the form <INPUT TYPE="file" NAME="*name*">. In this case, $form(*name*) evaluates to the name of a temporary file on the server host. (The uploaded data has been placed in that file). The temporary file will be automatically deleted when the swebs process exits (i.e., when the user's session is over). The original name of the file is also available as $form(*name*.upload)

$debug(n)
: *N* = 1 turns on debugging, which writes data to a logging file in /tmp. *N* = 0 turns off debugging. The default is 0.

$caucus_id()
: Evaluates to the name of the caucus userid, i.e. the userid that owns the Caucus files.

CML Reference Guide

Chapter 4.2: Browser & Server Information

[TOP] [UP] [PREV] [NEXT]

$userid()
: Userid of the current user.

$cml_dir()
: Evaluates to the directory name of the current CML file. For example, if the URL is

```
http://screenporch.com/spi/swebsock/0008404/0083664/SP31/center.cml?1+x+x
```

then $cml_dir() will evaluate to "SP31".

$cml_path()
: Evaluates to the full pathname of the top level CML directory, as set in swebd.conf.

$caucus_path()
: Evaluates to the full pathname of the Caucus home directory, as set in swebd.conf.

$caucus_lib()
: Evaluates to the full pathname of the Caucus library directory, as set in swebd.conf.

$http_lib()
: Evaluates to the full URL of the Caucus library directory, as set in swebd.conf.

$http_user_agent()
: Contents of the CGI environment variable HTTP_USER_AGENT. Usually a multi-word string that describes the browser client program.

$browser_format()
: Evaluates to the browser language code number, as originally set by the parameter BROWSER_FORMAT in swebd.conf.

$set_browser_format(code)
: Sets the browser format to language code number *code*, overriding the original value from swebd.conf.

$goodbye()
: Tells the swebs server dedicated to this user to change its timeout period to one minute. This is a graceful way to exit Caucus, and lowers system load. It is not required, the swebs server will eventually timeout by itself.

$new_win(width height)
: Functions like $t2url() translate URLs into HTML code that pops up a new window containing the URL. New_win() sets the size of such a new window to be *width* pixels wide by *height* pixels high. If *width* or *height* are not specified, new window size is left unchanged.

Evaluates to the new window width and height, respectively, separated by a space.

$unique()
: Return a unique number each time. Useful for tagging distinct instances of a particular page.

$version()

Returns version number of Caucus server software (e.g., "3.1.04").

$is_passwd()

Obsolete. See new password functions. Evaluates to '1' if a password changer program was defined in the configuration file swebd.conf, and '0' otherwise.

$reval(string)

Recursively evaluates *string* for CML functions. If *string* contains a CML function, which when evaluated expands to a CML function, reval() makes sure that *string* is continually interpreted until no CML functions remain.

Without reval(), CML text is scanned only once for CML functions.

$protect(string)

Prevents certain CML functions from taking effect. Any CML functions in *string* operate in a "protected" mode. This is useful, for example, in evaluating CML code that may have been placed (by a user) in the text of an HTML response.

Functions which have no effect when evaluated inside $protect() include: shell(), silent(), passwd(), set_wrap(), any set_co...(), any set_it...(), any set_my...(), any ad_...(), any chg_...(), mac_define(), set_user_var(), and set_conf_var().

$time()

Returns the current time on the server, in "epoch" timing, i.e. an integral number of seconds since 00:00 Jan 1, 1970 GMT. (See $epoch().)

$timezone() *(4.06)*

Returns the difference, in seconds, between the local timezone and UTC (aka GMT). This includes the effect of daylight savings time. For example, for a host using Eastern Standard (not Daylight) Time, $timezone() returns -18000.

$lice_max_users()

Evaluates to the total number of Caucus users (or "seats") allowed by this license. (A value of 0 means "unlimited".)

$lice_act_users()

Evaluates to the actual number of users who have "registered" with Caucus.

$lice_expires()

Returns the epoch time at which this license expires, or 0 if it does not expire.

$opsys()

Evaluates to a string describing the operating system of the host server. The first word is either "unix" or "nt". Subsequent words describe the particular version or platform.

$disk_failure()

Normally this function evaluates to an empty string. If Caucus encounters a disk-write error (such as caused by a suddenly completely full disk), $disk_failure() evaluates to an error code number, followed by the full pathname of the file at which the error was encountered.

CML Reference Guide

Chapter 4.3: File Access

[TOP] [UP] [PREV] [NEXT]

$file(name)

Include the entire text of file *name* at this point.

The $file() function should only be used to include relatively short (a couple of lines, maximum) files, such as when you need to include the contents of a file in the middle of an HTML or SET string that you are building. *Name* is relative to the CML_Path directory specified in the swebd.conf file. (See the Caucus installation guide for details.)

If you need to include a large file, or one that contains CML directives, see the "include" directive in section 5.

$readfile(name)

Evaluate to the entire contents of text file *name*. *Name* should be the full pathname of a file on the server host. Whereas $file() is meant as a way to include additional CML code in a page, $readfile() is meant for reading data that will somehow be processed or displayed by a CML page.

$write(name text)

Write *text* to file with absolute pathname *name*. Overwrites previous contents of *name*, if any.

$append(name text)

Append *text* to file with absolute pathname *name*.

$dosfile(name)

Truncates *name* to the first 8 characters, and replaces all dots (".") with underscores ("_"). Useful when *name* refers to a file on the client machine.

$copy2lib(file libname)

Copies *file* (a full pathname) to a new file called *libname*, in the Caucus file library. (See the parameters Caucus_Lib and HTTP_Lib in the swebd configuration file swebd.conf for more information about the Caucus file library.) *Libname* may contain sub-directory names, and is always treated as relative to the root of the Caucus file library. Sub-directories are created automatically. Thus a *libname* of "demo/xyz" would copy *file* to a file called "xyz" in a sub-directory "demo" under the Caucus file library, and would automatically create the "demo" directory if needed.

The function evaluates to the full URL of the newly created file, thus making it possible to make the file immediately available on the Web in any subsequently produced HTML.

$open(name perm)

Open a file *name* for reading (if *perm* is "r"), for writing (if *perm* is "w"), or to append to (if *perm* is "a"). Evaluates to a number which is the file "handle", or to "0" if the file could not be opened.

$readln(handle var)

Read a line from the file open on *handle*, and put the text into variable *var*. Evalutes to "1" if successful, or to "0" on end-of-file.

$writeln(handle text)

Writes *text* to the file open on *handle*. Evaluates to "1" on success, or "0" if *handle* does not refer to an open file.

$close(handle)
: Close file open on *handle*.

$rename(a b)
: Rename file *a* to file *b*.

$delfile(a)
: Delete file *a*. Must use full pathname of file.

$file_data(name bytes range...)
: Checks the first *bytes* characters of file *name* to see if its contents fit within certain byte-value ranges. (A *bytes* value of -1 means "check the entire file".)

  Range is one or more byte values or byte value ranges, expressed as decimal numbers. For example "32" is the decimal code for a "blank" character, and the range "65-90" covers the codes for the upper-case letters A through Z.

  $file_data() evaluates to 1 if the first *bytes* characters of *name* are included in the *range* values, and 0 otherwise. Thus, for example, $file_data(name 500 32 65-90) would evaluate to 1 if the first 500 characters were either blanks or the letters A-Z, and 0 otherwise.

  A good use of $file_data() is to determine if a file contains only "plain text", or is some other kind of file (word processor file, image file, etc.) If $file_data (name -1 9-26 32-126) is 1, *name* is most likely a text file.

$dirlist(format dir)
: If *format* is 0, evaluates to a space-separated list of the files (and directories) in directory *dir*. If *format* is 1, each filename is immediately followed by a space and the size of the file in bytes.

$output(name mask)
: Normally, CML lines that begin with a double-quote (") are interpreted and sent directly to the user's browser. The $output() function redirects this text, and writes it to a file *name*, instead. *Mask* is the numeric Unix file permission mask, e.g. a value of "644" means read/write owner, read group, and read world.

  The redirection takes effect on all quoted lines that follow the use of $output(). Another call to $output(), with no arguments, returns subsequent output from quoted lines to the browser, in effect "closing" the file.

CML Reference Guide

Chapter 4.4: Shell Access

[TOP] [UP] [PREV] [NEXT]

CML provides one comprehensive function (**$xshell()**) for accessing the Unix (or NT) command interpreter or "shell" (and thereby running commands or scripts from the shell).

For historical reasons (and to provide backward compatibility with earlier versions CML), several other shell functions are provided that implement subcases of $xshell().

$xshell(output synch effective command)

Runs *command* in a shell. *Output* controls what happens to the output from the command. If it is 0, the output is ignored. If greater than 0, the function evaluates to the output from the command. Values of 1, 2, and 3 cause default translation, translation from EUC, and translation from SJIS, respectively.

Synch controls synchronous operation. If 1, the *command* executes synchronously, i.e. $xshell() does not "return" until the *command* is complete. If 0, the *command* executes asynchronously, and $xshell() returns immediately (with no output, regardless of the value of *output*).

Finally, on Unix systems, *effective* controls the permissions with which *command* is run. If 1, the *command* runs with its real id as the Caucus id (see the parameter Caucus_ID in the swebd.conf configuration file). If 0, the *command* runs as the "real" id, as specified by the parameter Real_ID in swebd.conf.

Warnings and Notes:

- Unix: unless it is specifically necessary to execute shell commands that can access the Caucus data files, it is **strongly** recommended that all shell functions run as a "real" id that has limited access to Caucus and the rest of the system. This can best be accomplished by (a) using a value of 0 for *effective*, **and** (b) by making sure that Caucus is started from "root", with a swebd.conf Real_ID parameter of "nobody", or some similarly unprivileged id.
- NT: *eff_real* has no effect; all xshell() commands run as the "caucus" userid and can thus affect the Caucus data files. Be careful!
- NT: if *command* references a .EXE or .BAT file, you must include the extension as part of the file name.
- NT: using stdin ("standard input") inside a batch file may cause the batch file (and thus the Caucus process, which is waiting for it) to "hang". For example, batch commands like "echo Y|deltree xyz" seem susceptible to this problem.

Note also that all of the shell functions are ignored (have no effect) when inside a $protect() function.

The remaining functions are all subcases of $xshell(), and are provided for historical compatibility.

$shell(command)

Runs *command* in a shell. The function evaluates to the output from *command*. Example:

```
"It is now: $shell(date)
```

Equivalent to $xshell (1 1 0 command).

$jshell(type command)

Runs *command* in a shell. The function evaluates to the output from *command*, expecting that the command produces japanese text in EUC coding (*type*=2) or SJIS coding (*type*=3). (For more information about language type codes, see the DISKFORMAT parameter in swebd.conf.)

Equivalent to $xshell (type 1 0 command).

$silent(command)

Runs *command* in a shell. The output is ignored. The function evaluates to nothing, i.e. it effectively disappears. The example logs a user's userid to a temporary file.

```
" $silent(echo $userid() >>/tmp/log)
```

Equivalent to $xshell (0 1 0 command).

$asynch(command)

Runs *command* in a shell, immediately, without waiting for *command* to finish. Equivalent to $xshell (0 0 0 command).

CML Reference Guide

Chapter 4.5: Comparisons & Logic

[TOP] [UP] [PREV] [NEXT]

$and(a b ...)
Evaluates to the logical "and" of *a* and *b* and ... May have any number of arguments.

$or(a b ...)
Evaluates to the logical "or" of *a* and *b* and ... May have any number of arguments.

$not(a)
Evaluates to the logical negation of *a*.

$equal(x y)
If *x* and *y* are identical (they may be numbers or strings), evaluates to "1". Otherwise it is "0". **Note**: $equal() expects that *x* and *y* do **not** contain blanks; if they do, the results are unpredictable. If you must compare multi-word strings, see $quote().

$not_equal(x y)
Reverse of $equal().

$empty(str)
Evaluates to "1" if *str* is completely empty, and "0" otherwise.

$not_empty(str)
Evalutates to "1" if *str* is not empty, and "0" if it is completely empty.

$if(a b c)
If *a* is true, evaluates to *b*. Otherwise, evaluates to *c*. The classic triadic "if then else" operator. Arguments *a* and *b* must be only one word; *c* may be any length.

$gen_sort(direction words...)
Sorts the list of *words* in ascending (*direction* > 0) alphabetical order, or descending (*direction* < 0) alphabetical order. Evaluates to a list of indices to the original list, in sorted order. I.e., $gen_sort (1 a c b) evaluates to "1 3 2".

$num_sort(direction numbers...)
If *direction* is > 0, evaluates to the list of numbers, sorted in ascending numerical order. If direction is < 0, descending numerical order is used.

$triplet_sort(direction triplet_list...)
If *direction* is > 0, evaluates to the list of item triplets, sorted in ascending numerical order. If direction is < 0, descending numerical order is used. In this case, numerical order means the conference number is considered first, then the item number, and finally the response number.

$item_sort(direction code triplet_list...)
Evaluates to a sorted triplet list of items. If *direction* is > 0, sorting is done in ascending order; if < 0, descending order.

Code selects the property used to sort the items. A value of 1 sorts the items alphabetically by title. A value of 2 sorts alphabetically by the author's full name. A value of 3 sorts by the date of the last (undeleted) response to the item. (For *code* 1 and 2, the items must all be in the same conference, or the results will be unpredictable.)

CML Reference Guide

Chapter 4.6: Mathematics

[TOP] [UP] [PREV] [NEXT]

$plus(a b)
Evaluates to the sum of numbers *a* and *b*.

$plusmod(a b x)
Evaluates to sum of *a* and *b*, modulo *x*.

$minus(a b)
Evaluates to the difference, *a* - *b*.

$mult(a b)
Evaluates to the product of *a* and *b*.

$divide(a b)
Evaluates to the integer quotient of *a* / *b*.

$greater(a b)
Evaluates to "1" if *a* is greater than *b*. Otherwise "0".

$gt_equal(a b)
Evaluates to "1" if *a* is greater than or equal to *b*. Otherwise "0".

$less(a b)
Evaluates to "1" if *a* is less than *b*. Otherwise "0".

$between(a x b)
Evaluates to "1" if *x* is between *a* and *b* (*a* <= *x* <= *b*). Otherwise "0". Very useful for processing the result of server-side image maps.

$max(a b)
Evalutes to the larger of numbers *a* and *b*.

$min(a b)
Evaluates to the smaller of numbers *a* and *b*.

$bit_and(a b ...)
Evaluates to bitwise logical AND of *a*, *b*, etc.

$bit_or(a b ...)
Evaluates to bitwise logical OR of *a*, *b*, etc.

$bit_not(a)
Evaluates to the first 16 bits of the bitwise logical negation of *a*.

$hex2dec(a)
Evaluates to the (decimal) number with hexadecimal value *a*. E.g., $hex2dec(f) is 15.

$dec2hex(a)
Evaluates to the (hexadecimal) number with decimal value *a*. E.g., $dec2hex(15) is "f".

$random(max)
: Evaluates to a random number from 0 to *max*-1, inclusive.

CML Reference Guide

Chapter 4.7: String Manipulation

[TOP] [UP] [PREV] [NEXT]

$upper(words)
Converts all the text in *words* to upper case.

$upper1(words)
Converts the first letter of each word in *words* to upper case.

$lower(words)
Converts all the text in *words* to lower case.

$newline()
Evaluates to a newline character. Useful inside arguments to functions such as $t2hbr(), $ad_item(), etc.

$word(n str)
Evaluates to word number *n* of string *str*. Words are separated by one or more spaces. The first word is word number 1.

$rest(n str)
Evaluates to the "rest" of the words in a string, i.e. word number *n* through the end of *str*, inclusive.

$tablefind(word str)
Look for *word* in *str*. If it is identical to a single word, evaluate to the number of that word in *str*. Otherwise '0'.

$sizeof(str)
Evaluates to the number of words in string *str*.

$width(str)
Evaluates to the width (equivalent number of single-width characters) of *str*. Double-wide kanji have a width of 2.

$pad(num)
Evaluates to *num* blanks. Generally only useful inside <PRE> text.

$replace(a b c)
Each of the strings *a* and *b* must either be single characters, or else the (two or more digit) base-ten numeric representation of a single character. $replace() evaluates to string *c*, but with each instance of character *a* replaced by character *b*.

$asc2dec(text)
Converts *text* to a space-separated string of decimal numbers, representing the value of each character in *text*.

$str_index(what text)
Find the first occurrence of the (one-word) string *what* in string *text*. Evaluates to position number of *what* in *text*. (The first position is 0.) Evaluates to "-1" if not found.

$str_revdex(what text))
Find the **last** occurrence of the (one-word) string *what* in *text*. Evaluates to position number of *what* in

text. (The first position is 0.) Evaluates to "-1" if not found.

$str_sub(pos len text)
Evaluates to a substring of *text*, starting at position *pos*, *len* characters long.

$epoch(date)
Date must be a date of form DD-MMM-YY[YY]. The function evaluates to its equivalent "epoch" time, i.e. the number of seconds since Jan 1, 1970. (See $time().)

$dateof(time)
Evaluates to the date form (DD-MMM-YYYY HH:MM) of an "epoch" *time* in seconds.

$quote(words)
Treats all of the *words* as a single word. (It evaluates to the string *words*, with all of the blanks replaced with a non-printable control character.) This function is useful in several different circumstances:

1. When comparing two multi-word strings, as in:
   $equal ($quote (string1) $quote (string2))

2. To combine several words as one argument to the include directive, as in:
   include file.i $(cnum) $quote(conference description)

3. To supply a multi-word string to a function that specifically expects a $quote'd string, such as $add_resp().

$unquote(words)
Undoes the effect of $quote() when explicitly necessary. I.e. $unquote ($quote (words)) evaluates to the original, unmodified *words*.

CML Reference Guide

Chapter 4.8: Conference List Information

[TOP] [UP] [PREV] [NEXT]

There is a family of functions that provide basic information about conferences. All of these functions begin with "$cl_", to indicate that they refer to information about a list of conferences ("cl", as in **c**onference **l**ist).

$cl_list(names)

Evaluates to a list of conference numbers. If *names* is empty (i.e., nothing), $cl_list() evaluates to the list of all conferences on the host. If *names* contains one or more words, $cl_list() evaluates to the list of conferences that match any of the words in *names*. Example:

```
for cnum in $cl_list(web x)
```

$cl_list() becomes the list of all conferences whose names start with "web" or with "x". (The "for" loop thus sets `cnum` to each such conference number in turn.)

Notes:

- The list of conference numbers is sorted, not by number, but by the name of each conference, regardless of the order of the arguments to $cl_list().
- The rest of the $cl_ functions require that $cl_list() have been called at some time during the session with the relevant conference name (or no names at all).

$cl_num(name)

Evaluates to the number of the conference whose name matches *name*. (An abbreviation is a match). *Name* must have been in the list of conferences generated by any use of $cl_list().

$cl_name(num)

Evaluates to the name of conference number *num*. The name will always be in lower-case. $cl_list() must be called before $cl_name() can be used.

$cl_access(num)

Evaluates to the user's access level to conference *num*. 0 means the user is excluded from the conference, 1 means read-only access, 2 means full "include" access, and 3 means organizer access.

$cl_visible(num)

Evaluates to 1 if the conference is visible to **this** user, else 0. (Does not include effect of manager permission bits). Note the distinction from $co_visible(), the absolute visibility.

CML Reference Guide

Chapter 4.9: Conference Organizer Information

[TOP] [UP] [PREV] [NEXT]

Another family of functions relates to information about a conference that gets set (or changed) by the **c**onference **o**rganizer. In all of these functions, *num* is the conference number.

$co_org(num)
Evaluates to the userid of the primary organizer of the conference.

$set_co_org(num id)
Change the primary organizer of conference *num* to userid *id*. The caller must already be the primary organizer, or else have the MGR_BEORG manager bit.

$co_greet(num)
Evaluates to the text of the "greeting" for the conference. In the original (text) Caucus interface, the greeting was displayed every time a person entered a conference.

$co_intro(num)
Evaluates to the text of the "introduction" for the conference. In the original Caucus, the introduction was displayed when a person tried to join a conference for the very first time.

The introduction offers more information about the conference, to help a person decide if they really wish to join the conference.

$co_add(num)
Evaluates to "1" if ordinary users can add an item, or "0" otherwise.

$set_co_add(num add)
If *add* is non-zero, ordinary users may add new items. If *add* is "0", they may not.

$co_change(num)
Evalutes to "1" if ordinary users can change their own responses. Otherwise "0".

$set_co_change(num chg)
If *chg* is non-zero, ordinary users may change their responses. If *chg* is "0" they may not.

$co_visible(num)
Evaluates to "1" if conference name is visible to non-members in conference lists. Otherwise "0".

$set_co_visible(num vis)
If *vis* is non-zero, the conference name is visible. If *vis* is "0", the conference is invisible to non-members.

$co_userlist(num)
Evaluates to the text of the conference "userlist". The text of a simple userlist might look like:

```
:include
*
:organizer
smith
```

$set_co_userlist(num list)
Set the text of conference *num*'s "userlist" to *list*.

$co_remove(num)

Completely and permanently remove (delete) conference *num*. Will only work for managers with permission bit MGR_RMCONF.

$co_makeorg(num userid)

Immediately add *userid* as an organizer to conference *num*. Will only work for managers with permission bit MGR_BEORG.

$create_conf(name priorg org1 ...)

Create a new conference *name*, with primary organizer *priorg*, and secondary organizers *org1* etc. Will only work for managers with permission bit MGR_CRCONF. Evaluates to:

new conference number on success
0 if user does not have MGR_CRCONF permission
-1 if *name* already exists
-2 *priorg* not supplied
-3 internal database error
-4 if *name* is bad

$co_rename(cnum newname) *(4.06)*

Renames conference *cnum* to have the name *newname*. Will only work for managers with permission bits MGR_CRCONF or MGR_RMCONF. Evaluates to:

1 on success
0 if the user doesn't have manager permission
-1 if *newname* is bad
-2 if *cnum* doesn't exist
-3 if *newname* already exists!

CML Reference Guide

Chapter 4.10: Person Information

[TOP] [UP] [PREV] [NEXT]

Another family of functions provides information about a particular person who is registered with Caucus. All of these functions begin with "$per_".

$per_name(id)
: Evaluates to the full name of the person with userid *id*.

$per_intro(id)
: Evaluates to the text of the "brief introduction" of userid *id*.

$per_phone(id)
: Evaluates to the telephone number of userid *id*.

$per_laston(id)
: Evaluates to the date and time that userid *id* was last on (last using) Caucus.

$per_lastin(id cnum)
: Evaluates to the date and time that userid *id* was last in conference *cnum*. Evaluates to an empty string if *id* is not a member of the conference.

$per_real(id)
: Evaluates to the "real name" (as registered in the server host system password file) of user *id*. If there is no "real name", it evaluates to the empty string. (Note: this function is only meaningful when the Web userids are derived from the system password file userids. See the Caucus Installation Guide for more information.)

$per_delete(id)
: Delete the Caucus userid *id*. (Requires MGR_RMID manager bit). Evaluates to 0 on success, -1 if not allowed, and 1 on other error (such as userid does not exist).

  Note that this only deletes the information **about** user *id*. To delete the actual userid & password entry, see $pw_delete().

CML Reference Guide

Chapter 4.11: "My" Information

[TOP] [UP] [PREV] [NEXT]

The "$my_" and "$set_my_" functions relate to registration information about the current user.

$my_exist()
: Evaluates to "1" if the current user is registered with Caucus, and "0" otherwise.

$my_name()
: Evaluates to the current user's full name.

$my_phone()
: Evaluates to the current user's telephone number.

$my_intro()
: Evaluates to the text of the current user's "brief introduction".

$my_laston()
: Evaluates to the time and date the current user was "last on" Caucus.

$my_text()
: Evaluates to a number which represents when a person's items or responses should appear as "new". 0 means "later" (only after someone else adds a response), 1 means now (immediately becomes new), and 2 means never (it is immediately treated as "seen").

$set_my_text(n)
: Sets the value of my_text, as defined above, to *n*.

$set_my_name(name)
: Sets the current user's full name to *name*. (Will not change the user's name if *name* is empty.) Evaluates to '1' on success, '0' on failure. (Fails if attempting to create a new user, and the maximum total number of users for this license has been reached.)

$set_my_phone(number)
: Sets the current user's telephone number to *number*. (May be set to nothing). Evaluates to nothing.

$set_my_intro(text)
: Sets the current user's brief introduction to *text*. (May be set to nothing). *Text* may contain newlines. Evaluates to nothing.

$passwd(id newpw oldpw)
: **Obsolete**. See the new password functions. Change the password for user *id*, to *newpw* (from *oldpw*). Evaluates to a success code: 0 means success; non-zero is an error code.

$passcheck(id pw)
: **Obsolete**. See the new password functions. Evaluates to '1' if *id* and *pw* are a valid password pair, and '0' otherwise. If $passcheck() fails, most CML functions that reference actual Caucus data are disabled. (If $passcheck() is called again and succeeds, the functions are enabled. Functions are enabled by default.)

CML Reference Guide

Chapter 4.12: Groups of People

[TOP] [UP] [PREV] [NEXT]

$peo_members(cnum)

Evaluates to a list of userids that are members of conference *cnum*. The userids are sorted by "last name" of the actual users.

$peo_names(cnum names)

Evaluates to a list of userids of people who match *names*. A person matches if every word in *names* is an initial substring of some part of their name. If *cnum* is non-zero, matching people must also be a member of conference *cnum*.

$all_users(match)

Evaluates to a list of all of the userids registered with Caucus that begin with the initial substring *match*. If *match* is empty, evaluates to list of **all** of the Caucus userids. (Even on a site with thousands of users, this function evaluates quickly.)

CML Reference Guide

Chapter 4.13: Item Information

[TOP] [UP] [PREV] [NEXT]

The "it_" and "set_it_" functions provide or manipulate information about an item, or items, in a conference, or the user's participation in a conference. *Cnum* always refers to the conference number. *Inum* is a particular item number. *Rnum* is a particular response number.

$it_member(cnum)
: Evaluates to "1" if the current user is a member of the conference.

$it_join(cnum)
: Make the current user a member of the conference. Evaluates to "1" if joining is successful, and "0" otherwise.

$it_resign(cnum userid)
: Resign (remove) user *userid* from the conference. Evaluates to nothing. (If *userid* is not supplied, assumes the current user. Only organizers may successfully "resign" other users.)

$it_last(cnum)
: Evaluates to the number of the last item in a conference, i.e., the number of items.

$it_icount(cnum)
: Evaluates to the actual number of (non-deleted) items in a conference.

$it_inew(cnum)
: Evaluates to the number of new (and undeleted) items in a conference.

$it_rnew(cnum)
: Evaluates to the **total** number of new responses in a conference.

$it_iforgot(cnum)
: Evaluates to the number of forgotten items in a conference.

$it_wnew(cnum)
: Evaluates to the number of items that have 1 or more new responses.

$it_iunseen(cnum)
: Evaluates to the number of unseen items.

$it_listinew(cnum)
: Evaluates to a space-separated list of the new items in a conference. This list appears in "triplet" form. This means that each item is represented by three numbers: a conference number, an item number, and the number of the first relevant response. For example, if conference 17 has two new items, 5 and 6, $it_listinew() would produce the string "17 5 0 17 6 0". To parse triplet lists, use the functions $word() and $rest().

$it_listrnew(cnum)
: Evaluates to a "triplet" list of the new responses in a conference. The response number in a triplet is the first new response in the relevant item.

$it_listiunseen(cnum)

Evaluates to a "triplet" list of the unseen items in a conference. The response number is always 0.

$it_exists(cnum inum)
Evaluates to "1" if the item exists and the user is a member of the conference, and "0" otherwise.

$it_visib(cnum inum)
Evaluates to "1" if the item is visible to the user, i.e. has not been deleted or "forgotten". Otherwise "0".

$it_new(cnum inum)
Evaluates to "1" if the item is new to this user, i.e. it has a higher number than the highest item the user has seen. Otherwise "0".

$it_unseen(cnum inum)
Evalutates to "1" if this item is not new but has not been seen by the user. Otherwise "0".

$it_resps(cnum inum)
Evaluates to the number of responses. If the item does not exist (or was deleted), evaluates to -1. An item without any responses evaluates to "0".

$it_newr(cnum inum)
Evaluates to the number of the first response on this item that is *new* to this user. If no responses are new, evaluates to the number of responses + 1.

$set_it_seen(cnum inum rnum)
Marks all responses through *rnum* as "seen" by this user. To mark an item as "unseen", use an *rnum* of -1. To mark an item as "forgotten", use an *rnum* of -2.

$it_frozen(cnum inum)
Evaluates to "1" if the item is frozen, and "0" otherwise.

$set_it_frozen(cnum inum value)
A *value* of 1 freezes the item. A *value* of 0 "thaws" it.

$it_howmuch(cnum inum userid)
Evaluates to the number of responses seen by user *userid* to item *inum* in conference *cnum*. A value of -1 means the item is new to that user; -2 means the user has forgotten that item.

$it_parse(cnum text)
Evaluates to a triplet list of item/response numbers in conference *cnum* that match, in some form, *text*. *Text* may contain a variety of forms of listing items, separated by commas, including the example shown below. The result is an "or" match, i.e. all the items that match all the entries in *text*.

```
5
7-20
key words in an item title
"words in one item title", "words in another item title"
author "charles roth"
since 5/23/95
since -4
```

Note: to ease the writing (and reading) of CML pages, all of the $it_ functions that take two arguments (such as it_visib(), it_resps(), and it_newr()) may be written with *no* arguments. This means "re-use the exact same arguments as in a previous instance of one of these functions". **Warning**: results may be unpredictable if other $it_...() functions (those with more than two arguments) are called in between.

CML Reference Guide

Chapter 4.14: Response Information

[TOP] [UP] [PREV] [NEXT]

The "re_" functions provide information about a particular response. As in the previous section, *cnum* refers to a conference number, *inum* to an item number, and *rnum* to a response number.

$re_exists(cnum inum rnum)
: Evaluates to "1" if the response exists, and "0" if the response does not exist or was deleted.

$re_author(cnum inum rnum)
: Evaluates to the full name of the author (at the time the response was written).

$re_owner(cnum inum rnum)
: Evaluates to the userid of the author (owner) of the response.

$re_time(cnum inum rnum)
: Evaluates to the time and date the response was written.

$re_text(cnum inum rnum)
: Evaluates to the text of the response.

$re_prop(cnum inum rnum)
: Evaluates to the text property number of the response. (The property numbers are, for the moment, arbitrary, but are being used to distinguish how the user meant a response to be displayed -- e.g., as literal text with explicit line breaks, as plain text to be reformatted as simple HTML, or as explicit HTML as written by the responder.)

$re_title(cnum inum rnum)
: Evaluates to the title of the response. Only response 0 has a title, which is the title of the item.

$re_delete(cnum inum rnum)
: Deletes specified response. If *rnum* is 0, deletes entire item. Only the owner of the item or response (or an organizer) can successfully delete an item or response.

$re_epoch(cnum inum rnum)
: Evaluates to "epoch" timing (number of seconds since Jan 1 1970) when this response was written.

$re_bits(cnum inum rnum)
: Evaluates to any special properties possesed by this particular response. A bit value of 0x10 means the response was originally written by an organizer.

$re_copier(cnum inum rnum)
: Evaluates to the userid of the person who **copied** this response to its current location. (If the response was not copied, this evaluates to nothing, i.e. an empty string.)

$re_copied(cnum inum rnum)
: Evaluates to the "copied" string stored by add_resp(). It contains the data "*conference_name item response date time*". (If the response was not copied, this evaluates to nothing, i.e. an empty string.)

Note: as in chapter 4.13, all of these functions may be written with *no* arguments. In that case, the arguments from the previous use of any of these functions (in the same CML page) that *did* have arguments, are re-used.

http://screenporch.com/sp/product/cmlref40/ref415.html Go
2 captures
9 Jul 98 - 21 Aug 99
JUN JUL AUG
9
1997 1998 1999

CML Reference Guide

Chapter 4.15: Adding Items & Responses

[TOP] [UP] [PREV] [NEXT]

$add_resp(cnum prop inum owner author copied text)

Adds *text* as a response to conference *cnum*, item *inum*. The response is attributed to userid *owner*, and the name *author* is recorded with the response. (Typically *author* should be the current name of *owner*. If *author* is more than one word, it must be quoted, as in $quote(*author*).)

Copied is a string that designates where this response was copied from. It should be either **0**, to indicate that this is an original response, or a string of the form "*conference_name item response original_date original_time*", quoted (with $quote) as one word. (The current userid is marked as the *copier*, see re_copier().)

Evaluates to "1" if the adding succeeded, "0" if it failed. (Adding a response can fail if the user has read-only permission in the conference, or if the item is frozen.) Records *prop* as the text property number.

$ad_resp(cnum prop inum text)

Obsolete form of $add_resp(). Current user's userid and name are automatically used as *owner* and *author*.

$add_item(cnum prop owner author title copied text)

Adds *text* as a new item. The item is attributed to userid *owner*, and the name *author* is recorded with the item. (Typically *author* should be the current name of *owner*. If *author* is more than one word, it must be quoted, as in $quote(*author*).)

Title is the title of the new item. (Again, if *title* is more than one word, it must be quoted.)

Copied is a string that designates where this item was copied from. It should be either **0**, to indicate that this is an original item, or a string of the form "*conference_name item 0 original_date original_time*", quoted (with $quote) as one word. (The current userid is marked as the *copier*, see re_copier().)

Evaluates to the new item number if the adding succeeded, "0" if it failed. (Adding an item can fail if the organizer has turned off adding new items.) Records *prop* as the text property number.

$ad_item(cnum prop title text)

Obsolete form of $add_item(). Current user's userid and name are automatically used as *owner* and *author*. **Note** that (for $ad_item() only) there must be a newline between *title* and *text*. (See $newline().)

$ad_author(name)

Sets the author of the next item or response to be added, to the psuedonymn *name*. This name will be used in place of the normal author name, in the next (and only the next) call to $ad_resp() or $ad_item().

$chg_resp(cnum prop inum rnum text)

Replaces the response (*cnum*, *inum*, *rnum*) with *text* and the new *prop* property value.

$chg_title(cnum prop inum title)

Changes the title of item (*cnum*, *inum*) to title. *Prop* is required, but ignored.

$set_wrap(width)

When text is added as an item or response, it is automatically column-wrapped before it is stored in the

Caucus data files. Set_wrap sets the wrapping position to *width* single-width characters. A value of 0 turns column-wrapping off altogether.

CML Reference Guide

Chapter 4.16: Text Filters

[TOP] [UP] [PREV] [NEXT]

$t2hbr(stuff)

Turns plain text *stuff* (which may contain newlines) into HTML. It turns each newline into a <BR>. It also turns each of the special characters <, ", and > into their HTML special codes (unless escaped by a "\"). Example:

```
" $t2hbr( $shell(cat mytext) )
```

displays the text of an ordinary file *mytext* as HTML.

$cleanhtml(prohibit text)

"Clean HTML" filter. Filters HTML fragment in *text*, according to the rules in the string named *prohibit*. Provides a way to filter out certain HTML tags that you may not wish to be displayed in a response, such as applets, javascript, or even annoying tags such as <BLINK>.

Here are the sample contents of a *prohibit* string:

applet,prohibit,all   script,allow,all   blink,prohibit,tag

This means that everything between <APPLET> and </APPLET> is ignored; that the <SCRIPT> tag is allowed, and the <BLINK> tag (but only the tag, not the text that follows it) is ignored. Normally if something is allowed it does not need to be in the list, but advanced uses of this feature can support lists of tags that can be individually allowed or prohibited at run time.

$cleanhtml() includes all of the safety features of $safehtml(), such as automatic tag closing and mismatched quote correction.

$safehtml(prop stuff)

"Safe HTML" filter. Obsolete form of $cleanhtml(). Filters HTML fragment in text of *stuff*, making it "safe" to include in an existing HTML page. Specifically, it removes the tags <HTML>, </HTML>, <HEAD>, </HEAD>, <BODY>, and </BODY>. It "closes" any open tags (such as <B>) that don't have a matching closing tag (such as </B>). It looks for mismatched quotes inside a tag, and adds an extra quote if necessary. (For example, <A HREF="junk> becomes <A HREF="junk">.)

Prop is a number that controls certain properties of $safehtml(). It is the sum of a set of bitmasks (powers of 2); each bit controls a particular property. The properties are:

1 allow <FORM>s. Otherwise <FORM> tags are removed, like <BODY>.

$rhtml(stuff)

Obsolete form of $safehtml(), without the Prop argument. $rhtml(stuff) is equivalent to $safehtml(0 stuff).

$t2html(stuff)

Attempts an "intelligent" filtering of plain text stuff into HTML. Blank lines become <P>'s. Parses and translates URL's into anchored links with the same names. (see $t2url().)

$t2url(stuff)

Translates URLs in *stuff* into anchored links (that pop up a new window) with the same names. Both this function and $t2html() translate URLs that begin with any of the schemes http:/, gopher:/, telnet:/, ftp:/, or mailto:.

$wrap2html()

A more intelligent (than $t2html) filtering of plain text into HTML. Acts as much as possible like a typical word-processor. Each single "hard" RETURN in the original text translates into a <BR>; multiple RETURNs become sequences of "&nbsp;<P>". Groups of N spaces become N-1 "&nbsp;"s plus a regular space. A tab is treated as a group of 5 spaces. Parses and translates URL's into anchored links.

Special note: All 3 functions also recognize and translate special "caucus" URLs of the form "http:/caucus...", into a reference to a particular Caucus CML page on the current host (and with the current swebs subserver). For example, "http:/caucus" becomes a reference to the Caucus Center page, i.e. center.cml, and "http:/caucus/conf_name" becomes a reference to confhome.cml for conference *conf_name*. This is one of the **very** few instances in which the CML interpreter assumes knowledge of the names and arguments of the actual CML files. (Normally this would be a **bad** idea, but in this case the feature is so powerful and useful as to allow the exception.)

$t2amp(stuff)

Translates all "&"s in *stuff* into "&amp;". Useful to "pre-escape" HTML code that is going to be "unescaped" when displayed by a browser. (This pre-escaping is essential when using Caucus to edit a response containing HTML code. Without it, any escaped HTML special sequences like "&gt;" would lose their meaning after one edit.)

$t2esc(stuff)

Translates all instances of "&", "<", and ">" in *stuff* into their HTML code equivalents (**&amp; &lt;** and **&gt;**). Useful to "pre-escape" HTML code that is going to be "unescaped" when displayed by a browser.

$escquote(text)

Translates all double-quotes in *text* to the HTML special sequence "&quot;". This is primarily useful for placing text (that contains double-quotes) inside a double-quote-delimited field inside an HTML <INPUT> tag.

$t2mail(address)

Attempts to translate *address* into a "mailto:" URL. (For example, if *address* is "joe@xyz.com", $t2mail() generates "<a href="mailto:joe@xyz.com">joe@xyz.com</A>".) If *address* does not appear to be an e-mail address, it is passed through unchanged.

$wraptext(width text)

Word-wraps *text* to *width* (single-width-character) columns by inserting newlines in the appropriate places.

$mac_define(name text)

Defines a CML macro *name* that expands to *text*. See the CML macros chapter for more information. If *name* is already defined, the original definition is erased and replaced by the new one.

$mac_define() is an "protected" function, i.e. it is a no-op when called from within $protect().

$mac_expand(text)

Expands any macro invocations in *text*. Evaluates to *text*, with the macro invocations replaced by the expansion of the macros. See the CML macros chapter for more information.

CML Reference Guide

Chapter 4.17: User, Conference, Item, Site Variables

[TOP] [UP] [PREV] [NEXT]

The "regular" CML variables (e.g., "set var xyz" or "$(var)") are ephemeral: once the dedicated swebs server has exited, the values of those variables are lost.

CML also provides a set of variables that are persistent across sessions, and tied to a particular user, conference, item, or to the site as a whole. Such variables may contain any amount of text, including newlines. They provide a convenient way to extend a Caucus interface, and to customize how the interface appears to a particular user or in a particular conference or item.

Note that evaluating a variable is a fairly fast process. (All variables for a particular user, conference, item or site, are loaded when needed, and cached.) Setting a variable is much slower, since the values must be written to disk.

$user_var(user vname)
: Evaluates to the value of userid *user*'s variable called *vname*.

$set_user_var(user vname value)
: Sets userid *user*'s variable *vname* to *value*.

$clear_user_var()
: Clears the user variable cache.

$conf_var(cnum vname)
: Evaluates to the value of conference *cnum*'s variable called *vname*.

$set_conf_var(cnum vname value)
: Sets conference *cnum*'s variable *vname* to *value*.

$clear_conf_var()
: Clears the conference variable cache.

$item_var(cnum inum vname)
: Evaluates to the value of conference *cnum*, item *inum*'s variable called *vname*.

$set_item_var(cnum inum vname value)
: Sets conference *cnum*, item *inum*'s variable *vname* to *value*.

$clear_item_var()
: Clears the item variable cache.

$site_var(vname)
: Evaluates to the value of the site variable called *vname*. Site variables are "global" across an entire

installation of Caucus and are available to all users.

$set_site_var(vname value)
Sets site variable *vname* to *value*.

$clear_site_var()
Clears the site variable cache.

It is important to remember that all of these variables are **cached** for each user. This makes getting the value of a variable very fast; but it also means that if someone else changes a variable, your copy of the variable may be out of date.

In most cases this doesn't matter. But if you need to be sure that the value of a variable is up-to-date, call the appropriate $clear_... function immediately before using the variable. For example, the code below:

```
$clear_conf_var()
set quote $conf_var($(cnum) daily_quote)
```

ensures that the user has the most up-to-date value of the conference variable "daily_quote".

CML Reference Guide

Chapter 4.18: Searching Conference Text

[TOP] [UP] [PREV] [NEXT]

Several very specialized functions provide the capability to search for and display text in the conference items and responses.

$find_it(cnum inum r0 r1 any inword text)

Search conference *cnum*, item *inum*, responses *r0* through *r1*. (If *r1* is -1, search through the last response). Look for the word (or words) in *text*.

The *any* and *inword* arguments modify exactly how and when the search succeeds. If *any* is 1, the search is successful if any of the words in text are found in a response. If *any* is 0, the search succeeds only if all of the words in text are found in the same response. If *inword* is 1, the words in text match no matter where they are found in the response -- including in the middle of a word in the response.

(For example, "the" will match "o**the**r".) If inword is 0, matches must occur at the beginning of a word. (In that case, "the" will not match "other", but it will match "**the**sis".)

Find_it() evaluates to a triplet list of responses that had successful matches. (E.g., "17 2 5 17 2 8" means that responses 5 and 8 in item 2 in conference 17 had successful matches.)

$search_it(cnum inum r0 r1 any text)

This is an obsolete form of $find_it(). It is equivalent to $find_it() with an *inword* of 0.

$find_filter(size words... inword text)

Find_filter is really a text filter. It is meant to be used to display just the "hits" in a response that contains a word or words searched for via $search_it(). It boldfaces the searched-for words, and displays 3 lines of *text* around each hit.

Text is typically the entire text of a response. *Words* contains the word or words that were searched for. *Size* is the number of distinct words in *words*. *Inword* should have the same value it did in $find_it() -- it controls whether matches may be found in the middle of a word (*inword* = 1), or only at the beginning of a word (*inword* = 0).

$search_filter(size words... text)

This is an obsolete form of $find_filter(). It is equivalent to $find_filter() with an *inword* of 0.

CML Reference Guide

Chapter 4.19: CML Page Functions

[TOP] [UP] [PREV] [NEXT]

One of the most challenging tasks in creating sophisticated interfaces in CML is keeping track of where the user has been. For example, a user may start at page A, go to page B to fill out a form, which in turn is processed by page C... which should return the user to page A. If page B may be invoked from many different places, this task (remembering where to return to after page C) can get quite complicated.

This issue is dealt with more fully in the forthcoming "Caucus 3.1 Programmer's Guide". This section details four CML functions which make this capability possible.

$page_save(refresh cmlfile arglist fragment description)

> This function "saves" a CML page reference in a table inside the CML interpreter. It evaluates to (i.e., returns) a slot number in that table, which may be used by the other $page_... functions to access the saved page. The arguments to $page_save() define a page reference in such a way that the reference can be used later to easily "return to" that page later.
>
> *Cmlfile* is the name of the CML file. *Arglist* is the list of arguments to that file that should be remembered. (*Arglist* must be one word, so typically the arguments are specified in their URL form, i.e. with plus signs separating the individual arguments.) *Fragment* is the anchor point where that document should be re-entered, e.g. "#here". (If there is no such anchor point, *fragment* should just be "#".) *Description* is just ordinary text that describes that page; it may be any number of words, including none.
>
> The "Caucus Center" page shown in the example CML file in section 3 uses $page_save() to save the current location in a table slot:

```
set nxt $page_save (1 center.cml \
        $arg(2)+$arg(3)+$arg(4)+$arg(5)+$arg(6)+$arg(7)+$arg(8) \
                # $(center_name) )
```

> This CML code fragment saves the current page (center.cml), with its list of arguments ($arg(2)+...), no fragment ("#"), and a text description (contained inside the variable center_name). The saved page reference is stored in a slot, and the slot number is stored (by the "set" statement) in variable **nxt**.
>
> The *refresh* argument is somewhat complicated. The slot table in the CML interpreter has a fixed size... which means that slots that haven't been touched in a while will get automatically recycled. *Refresh* is a number that refers to the arguments in *arglist*. If *refresh* has a value of N, then the N'th argument in *arglist* is assumed to be a slot number, and that slot is refreshed -- that is, protected from being recycled until the rest of the slots in the table have been recycled.   See the previously mentioned Programmer's Guide for more information.

$page_get(slot)

> Evaluates to the entire string saved in *slot* (by $page_save()). The first word of the result is the page name, the second word is the argument list, the third word is the fragment (anchor name, with "#"), and the fourth through last words are the page description.

$page_return(slot #override empty)

> Evaluates to a string that can be used in an HTTP "Location:" directive to "return to" a page saved in *slot*. *#override* is a fragment anchor that may be used to override the anchor that was saved (with $page_save()). If it is just "#", the original (saved) anchor is used, otherwise *#override* is used. *Empty* should be a full CML page reference, to be used only if there is no page saved at slot.

Here is an example from the Caucus 3.1 additemf.cml file, which processes adding a new item to a conference, and then returns to the page which invoked "create a new item":

```
   "Location: $(href)/$page_return($arg(2) # \
                            center.cml?$(nch)+0+x+x+x+x+x+x)
```

In this case, $arg(2) is the slot number of the page that originally invoked "create a new item". There is no override on the saved fragment anchor, and the default page (in case there was no saved "calling" page) is center.cml, the "Caucus Center" page.

$page_caller(which slot)

Evaluates to the slot number of the page which "called" the page saved at *slot*. Assumes that the caller of a page is stored in the argument list to that page, in argument number *which*.

CML Reference Guide

Chapter 4.20: Manager Functions

[TOP] [UP] [PREV] [NEXT]

Caucus version 3.2 adds the notion of "managers", specific users who have managerial capabilities not granted to the regular Caucus user. When Caucus is installed, one such userid is designated the "primary manager"; this person has all capabilities at all times, and cannot be removed.

The primary manager may then add other managers with more restricted capabilities. Each capability corresponds to a single bit in a bit mask: the capabilities, and their values and names are shown below:

```
MGR_SET    0x001  modify list of managers
MGR_CRCONF 0x002  create new conferences
MGR_RMCONF 0x004  remove conferences
MGR_BEORG  0x008  become an organizer of any conference
```

There are two CML functions that work with the list of managers and their capabilities:

$mgr_list()
: Evaluates to the complete list of managers, in the form "manager1 mask1 manager2 mask2 ...". Each mask is the numeric sum of that manager's capabilities.

$mgr_list(userid)
: Evaluates to "userid mask", where mask is the sum of that specific manager's capabilities.

$set_mgr_list(mgr1 mask1 mgr2 mask2...)
: Replace entire manager list with the given arguments. (Requires that the current user have MGR_SET capability!)

$admin_mail()
: Evaluates to the e-mail address of the Caucus administrator (as defined in **swebd.conf**).

CML Reference Guide

Chapter 4.21: Password Functions

[TOP] [UP] [PREV]

Caucus version 3.2 adds a much more complete set of userid & password manipulation functions. See the swebd configuration file **swebd.conf** for more information about setting up the initial userid & password management.

$pw_can_add()
: Evaluates to 1 if userids may be added to the userid & password database, and 0 otherwise.

$pw_can_change()
: Evaluates to 1 if passwords may be changed in the userid & password database, and 0 otherwise.

$pw_can_delete()
: Evaluates to 1 if userids may be deleted from the userid & password database, and 0 otherwise.

$pw_can_verify()
: Evaluates to 1 if userids and passwords may be verified from the userid & password database, and 0 otherwise.

The values of the above 4 functions are taken directly from the parameters of the same name in **swebd.conf**.

$pw_add(id pw override)
: Add the userid *id* with password *pw* to the password database. If *override* is 1, anyone can add a userid. Otherwise, the user must have the MGR_MKID permission bit. Evaluates to 0 on success, or one of the error codes listed below.

$pw_change(id pw)
: Change userid *id*'s password to *pw*. A user may change their own password, or a manager with the MGR_CHGPASS permission bit may change anyone's password. Evaluates to 0 on success, or one of the error codes listed below.

$pw_delete(id)
: Delete userid *id* from the password database. Requires that the user have the MGR_RMID permission bit. Evaluates to 0 on success, or one of the error codes listed below.

(Remember to delete the user information with $per_delete() before deleting the userid!)

$pw_verify(id pw)
: Verifies that userid *id* has password *pw*. Evaluates to 0 on success, or one of the error codes listed below.

The error codes for the previous 4 pw_ functions are:

-1	Action not allowed
1	Program error
2	ID already exists
3	ID does not exist
4	Password is incorrect

6	Password is too long
7	Password has bad characters
8	Password database lock failed
9	Can't read password database
10	Can't write to password database
11	System error, disk may be full

CML Reference Guide

Chapter 5: CML Directives

[TOP] [PREV] [NEXT]

CML pages are like mini-programs. They contain directives which control which lines of HTML code will actually get sent to the browser, or control how many times a set of HTML lines will be evaluated.

The directives are:

1. For
2. Count
3. While
4. If
5. Elif
6. Else
7. Set
8. Include
9. Return
10. Quit
11. Break
12. Eval

Plus an "**end**" directive, that closes the "for", "count", "while", "if", "elif", and "else" directives.

5.1 For

The CML "for" loop evaluates a set of lines multiple times. It looks like:

```
for variable1 [variable2 ... ] in list
     (HTML code or other CML directive code)
     ...
end
```

where *variable1, variable2* etc. are names, and *list* is a list of words or values. Typically *list* may be the result of a CML function. The for loop evaluates the lines between "for" and "end", substituting the words in *list* as the values of *variable1*, *variable2*, etc. (The brackets simply mean that *variable2*, etc. are optional. The brackets would not actually appear in the syntax of the for loop.)

For example, the loop:

```
for x in abc qrs xy
     ...
end
```

will evaluate the lines between "for" and "end" three times, using each word in *list*. (If there are no words, the lines will be skipped.) The first time through the loop, *x* will have as its value "abc". The second time it will have the value "qrs", and so on.

A different example shows the use of multiple *variables*:

```
for one two in alpha beta delta gamma
    ...
end
```

The first time through the loop, *one* will have the value "alpha" and *two* will have the value "beta". The second time, *one* will have the value "delta", and so on.

The indenting of each line as shown above is not necessary, but it is a good idea. It helps make the CML code much more readable.

5.2 Count

The CML "count" loop is similar to the "for" loop. It looks like:

```
     count variable x y
          (HTML code or other CML directive code)
          ...
     end
```

where *variable* is a name, and *x* and *y* are numeric values or expressions. The count loop will evaluate the lines between "count" and "end" one time for each integer value between *x* and *y*, inclusive. The first time, variable will have the value *x*. Then *x*+1, and so on, up to and including *y*. If *y* is less than *x*, the lines will be skipped entirely.

5.3 While

The CML "while" loop is perhaps the simplest loop control directive. It has the form:

```
     while expression
          (HTML code or other CML directive code)
          ...
     end
```

The "while" loop evaluates *expression*, and examines the first word of the result. If it is a number, not equal to 0, all of the lines between "while" and "end" are evaluated. The loop then repeats, re-evaluating *expression*, and so on. The "while" loop will continue to execute as long as *expression* is non-zero, so be careful!

5.4 If

The CML "if" statement evaluates a set of lines if a certain condition is true. It looks like:

```
     if condition
          (HTML code or other CML directive code)
          ...
     end
```

where *condition* is some expression. If there is at least one word in *condition*, and the first word is a non-zero number, then the enclosed set of lines will be evaluated once. Otherwise they will be skipped. (Also see the related function $if().)

5.5 Elif

The "if" statement may be extended to handle multiple exclusive cases with the "elif" directive. It looks like:

```
     if condition1
          (HTML code or other CML directive code)
          ...
     end

     elif condition2
          ...
     end
```

The lines between "elif" and "end" are evaluated when the previous "if" *condition1* failed (was 0 or did not exist) and the first word of *condition2* is a non-zero number.

Multiple "elif"s may be strung together, one after another. Only one of the blocks of CML code between the if/end and elif/end pairs will be executed.

5.6 Else

There is an (optional) matching "else" to the CML "if" and "elif" statements. It looks like:

```
if condition
     (HTML code or other CML directive code)
     ...
end

else
     ...
end
```

The lines between "else" and "end" are evaluated if *condition* is 0, or does not exist at all. *Note*: *the "if" must have its own "end"!* "Else" may be used with just an "if", or a series of "if"s and "elifs". If the latter, it must be the last of the series.

5.7 Set

The "for" and "count" directives define the value of a variable during iterations of the lines between the "for" or "count", and the matching "end" directive. Outside of those loops, the variable is undefined.

A variable may also be defined across the evaluation of all CML pages, using the "set" directive. It looks like:

```
set variable x
```

where *variable* is a name, and *x* is some expression. For the rest of this session, *variable* has the value *x* (unless changed by another "set" directive). Variables defined by "set" are considered "global" in scope, i.e. the variables are available in all subsequently evaluated CML pages.

5.8 Include

The "include" directive includes the text of a CML file at the current point. It has the syntax:

```
include filename  [ arg1 [ arg2 ... ] ]
```

where *filename* is the name of a file, or a set of CML functions that evaluate to the name of a file. *Filename* is relative to the CML_Path directory specified in the swebd.conf file. (See the Caucus installation guide for details.) The brackets indicate that arguments *arg1*, *arg2*, and so on are optional (they are not actually part of the syntax). If the arguments are present, they are available inside the included file via the $inc(n) function.

Include directives are evaluated according to the current context. For example:

```
count x 1 3
     include file.$(x)
end
```

would include the contents of the files **file.1**, **file.2**, and **file.3**.

Include understands the "quoting" of multiple words as one *arg*. See $quote() for details.

5.9 Return

The "return" directive immediately "returns" from the current CML file or CML include file. When encountered in a CML file, it terminates processing of that file, as if it had "run off the end" of the file. If encountered in an included file, it immediately "returns" to the includ**ing** file, and continues with the line after the include directive.

Return is particularly useful in CML include files that handle multiple conditions. Return can be used to say "I'm done with this section", rather than use if/else if combinations to exclusively handle different conditions. For example:

```
if condition 1
   do something
   return
end

if condition 2
   do something else
   return
end

"etc...
```

A return in a main CML file has the same effect as a quit.

5.10 Quit

The "quit" directive immediately ceases processing of the current page, whether in a main CML **or** an include file. No more processing is done of that file, or any file that may have "included" the file.

It is particularly useful in CML pages that need to handle special case or "error" conditions. For example:

```
if some "error" condition
     "Location: http://www.xyz.com/errorpage.html
     "
     quit
end

#---OK, go on with the main case here...
"Content-type: text/html
"

"etc...
```

5.11 Break

The "break" directive immediately exits the innermost "for", "count", or "while" loop, and continues execution of the CML script after the closing "end" of that loop.

5.12 Eval

The "eval" directive simply evaluates any CML functions on the rest of the line. Any results from the evaluation are ignored (thrown away).

Eval is really just a more straight-forward way of saying "set ignore some_function".

CML Reference Guide

Chapter 6: Index to CML Functions

[TOP] [PREV]

This alphabetical index to the complete list of CML functions includes a very brief description of the purpose of each function.

A B C D E F G H I J L M N O P Q R S T U V W X

$ad_author()	set psuedonymn for response
$ad_item()	Add an item (obsolete)
$ad_resp()	Add a response (obsolete)
$add_item()	Add an item
$add_resp()	Add a response
$admin_mail()	Caucus Administrator e-mail
$all_users()	list caucus userids
$and()	logical 'and'
$append()	append text to a file
$arg()	argument to CML page
$asc2dec()	decimal values of characters in string
$asynch()	run shell asynchronously

$between()	a <= x <= b?
$bit_and()	bitwise and
$bit_not()	bitwise negation
$bit_or()	bitwise or
$browser_format()	browser language code

$caucus_id()	Caucus userid
$caucus_lib()	caucus library directory
$caucus_path()	caucus home directory
$chg_resp()	Change text of response
$chg_title()	Change title of item
$cl_access()	user's access level to conference
$cl_list()	get list of conference numbers
$cl_name()	get name of a conference
$cl_num()	get number of a conference name
$cl_visible()	conference visible to this user?
$cleanhtml()	clean HTML filter
$clear_conf_var()	Clear conf variable cache
$clear_item_var()	Clear item variable cache
$clear_site_var()	Clear site variable cache
$clear_user_var()	Clear user variable cache
$close()	close open file
$cml_dir()	CML directory in URL
$cml_path()	cml directory path
$co_add()	can users add new items?
$co_change()	can users change their responses?
$co_greet()	conference greeting text
$co_intro()	conference introduction text
$co_makeorg()	add an organizer

$co_org()	userid of organizer
$co_remove()	delete conference
$co_rename()	rename conference
$co_userlist()	conference 'userlist'
$co_visible()	conf visible to non-members?
$conf_var()	value of a conference variable
$copy2lib()	copy file to file library
$create_conf()	create conference
$dateof()	convert time in seconds to full date form
$debug()	debugging switch
$dec2hex()	decimal to hexadecimal conversion
$delfile()	delete a file
$dirlist()	list directory
$disk_failure()	disk-write error occurred?
$divide()	integer quotient of two numbers
$dosfile()	truncate to 8 char filename
$empty()	is string empty?
$epoch()	convert date to epoch time
$equal()	test equality of two strings
$escquote()	escape double-quotes
$file()	include contents of file 'name'
$file_data()	check contents of file against a range of values
$find_filter()	Display results of search
$find_it()	Search items for text
$form()	HTML forms data
$gen_sort()	alphabetic sort
$goodbye()	make server exit in one minute
$greater()	A > B?
$gt_equal()	A >= B?
$hex2dec()	hexadecimal conversion
$host()	host name
$http_lib()	URL of Caucus library
$http_user_agent()	browser name
$if()	if/else expansion
$inc()	argument to include file
$is_passwd()	Is there a password changer?
$it_exists()	does item exist?
$it_frozen()	is item frozen?
$it_howmuch()	how much seen by a user?
$it_icount()	actual number of items
$it_iforgot()	number of forgotten items
$it_inew()	# of new items in conference
$it_iunseen()	number of unseen items
$it_join()	make user member of conf
$it_last()	last item in conference
$it_listinew()	list of new items in conference
$it_listiunseen()	list of unseen items
$it_listrnew()	list of new responses in conf.
$it_member()	user member of conference?
$it_new()	Is item new?
$it_newr()	First new response to item
$it_parse()	parse list of items
$it_resign()	remove user from conf.

$it_resps()	Number of responses to item
$it_rnew()	total # of new responses in conf.
$item_sort()	sort by title, author, date
$it_unseen()	Is item unseen?
$it_visib()	Is item visible to current user?
$it_wnew()	# of items with new responses
$item_var()	value of an item variable
$jshell()	japanese shell command
$lice_act_users()	actual # users
$lice_expires()	epoch time at which license expires
$lice_max_users()	total # users allowed
$less()	A < B?
$lower()	convert to lower case
$mac_define()	macro definition
$mac_expand()	macro expansion
$max()	maximum of A and B
$mgr_list()	get list of managers
$min()	minimum of A and B
$minus()	subtract two numbers
$mult()	product of two numbers
$my_exist()	does this user exist?
$my_intro()	brief introduction of current user
$my_laston()	date user last on caucus
$my_name()	name of current user
$my_phone()	telephone of current user
$my_text()	when does my text appear new?
$new_win()	set size of windows created by $t2url()
$newline()	produce 'newline' character
$not()	logical 'not'
$not_empty()	is string non-empty?
$not_equal()	test equality of two strings
$num_sort()	numerical sort
$open()	open a file
$opsys()	host server operating system
$or()	logical 'or'
$output()	redirect HTML output
$pad()	provide blank padding
$page_caller()	get caller of a page
$page_get()	value of saved page reference
$page_return()	return to a saved page
$page_save()	save page reference
$passcheck()	Check id and password
$passwd()	change user's password
$peo_members()	list of members of conference
$peo_names()	find people by name
$per_delete()	delete person
$per_intro()	person's brief introduction
$per_lastin()	time last in a conf
$per_laston()	date/time person last on caucus
$per_name()	person's name
$per_phone()	person's telephone
$per_real()	real name of userid
$pid()	swebs process id

$plus()	add two numbers
$plusmod()	a + b modulo x
$protect()	Allow only safe CML functions
$pw_add()	Add a userid
$pw_change()	Change a password
$pw_delete()	Delete a userid
$pw_verify()	Verify correct password
$pw_can_add()	Can we add a userid?
$pw_can_change()	Can we change a password?
$pw_can_delete()	Can we delete a userid?
$pw_can_verify()	Can we verify correct password?
$quote()	"Quote" words as one word
$random()	random number
$re_author()	Author of response
$re_copied()	Information about copied response
$re_copier()	Userid that copied this response
$re_bits()	response property bits
$re_delete()	delete item or response
$re_epoch()	resp time in seconds
$re_exists()	Does response exist?
$re_owner()	Owner of response
$re_prop()	Property number of response
$re_text()	Text of response
$re_time()	Date/time response written
$re_title()	Title of item
$readfile()	read contents of file
$readln()	read line from file
$rename()	rename file
$replace()	replace all A's with B's
$rest()	remaining words in a string
$reval()	Recursive CML evaluation
$rhtml()	see $safehtml
$safehtml()	Text -> safe HTML (obs)
$search_filter()	Display results of search
$search_it()	Search items for text
$set_browser_format()	set browser language code
$set_co_add()	control users adding new items
$set_co_change()	allow changing responses
$set_co_org()	set primary organizer
$set_co_userlist()	set text of conference 'userlist'
$set_co_visible()	control conference name visibility
$set_conf_var()	set value of conference variable
$set_it_frozen()	freeze or thaw item
$set_it_seen()	Mark responses seen
$set_item_var()	set value of an item variable
$set_mgr_list()	set list of managers
$set_my_intro()	set user's brief introduction
$set_my_name()	set current user's name
$set_my_phone()	set current user's telephone
$set_my_text()	set when text appears new
$set_user_var()	set value of a user variable
$set_wrap()	control paragraph wrapping
$shell()	run shell command
$silent()	run shell command quietly
$sizeof()	number of words in a string
$str_index()	find string in other string

$str_revdex()	reverse find string
$str_sub()	extract a substring
$t2amp()	escape &'s
$t2esc()	escape & < >
$t2hbr()	Text -> lines with <BR>'s
$t2html()	Text -> formatted HTML
$t2mail()	translate to e-mail URL
$t2url()	Text -> URL's with 'hot' links
$tablefind()	find a word in a string
$time()	current epoch time value
$timezone()	difference between local time and UTC
$triplet_sort()	sort item triplets
$unique()	return a unique number
$unquote()	undoes effect of $quote()
$upper()	convert to upper case
$upper1()	uppercase 1st letter
$user_var()	value of a user variable
$userid()	user's userid
$version()	software version number
$width()	number of characters
$word()	N'th word of a string
$wrap2html()	wordwrap text -> HTML
$wraptext()	wordwrap text
$write()	write text to a file
$writeln()	write text to file
$xshell()	comprehensive shell access

Caucus Architecture Description

Screen Porch

1. Introduction

This document describes the overall architecture of Caucus. This includes the mechanics of how the web interface actually works, and the location, names, and formats of the most important files.

While it is not necessary to read or understand this document in order to *use* or even to *install* Caucus, it is very helpful if you intend to modify the web interface, or to connect other applications or programs to Caucus.

This document assumes a general familiarity with HTML, Web server management, Unix or Windows NT commands and processes, and the use of Caucus.

2. Caucus Design Goals

There were seven main design goals that shaped the architecture of Caucus:

1. Allow the use of *any* Web browser to provide a graphical user interface to Caucus conferences. (In practice, this has come to mean Netscape 2 or Internet Explorer 3.02 or higher.)

2. Provide the tools for Webmasters to build a completely customizable Caucus interface. Caucus uses "CML" (Caucus Mark-up Language) scripts, which are analogous to individual HTML pages.

   Caucus includes a default set of such scripts (pages), but they may be completely customized by the local site. This is in keeping with the long-standing Caucus tradition of complete customizability.

3. The Caucus server was built on top of the existing Caucus API (applications programmer interface) function library, minimizing development time and guaranteeing data compatibility.

4. Caucus works side by side with existing Caucus ("text interface") software. A Caucus user may access conferences through the Web *or* the text interface, without conflict.

5. Caucus works with existing Unix and Windows NT HTTP servers, through the CGI interface. The Caucus server could also be adapted to work with a custom HTTP server to provide for higher efficiency.

6. The Web "access authorization" userid and password scheme is used to provide secure access to Caucus. When a userid has been verified by the Web server, that same userid is used to identify the particular Caucus user. All normal Caucus security (access to specific conferences, etc.) applies.

   An interface to other authorization schemes is also available as part of the Caucus server.

7. Transactions between the browser and the Caucus server must be as efficient as possible. The main effect of this on the design is the creation of a dedicated "sub-server" process for each user's Caucus session.

3. Caucus Web Interface: Transactions

This section describes what actually happens when a person uses a Web browser to access Caucus. In the steps listed below, "swebd" refers to the master Caucus server process. "Swebs" is the dedicated user "subserver" process. "Swebsock" is a light-weight program that passes data to and from swebd. "Httpd" is the standard name for the HTTP server process.

3.1 Initial connection to Caucus

1. The user's browser sends a connection request (over the Internet, or a local intranet) to the host's HTTP server.

2. The HTTP server immediately spawns (or connects to a pre-existing) child httpd process to handle the request.

3. The initial "connection to the Caucus server" is actually an access-authorization (i.e., userid and password) protected URL that runs a CGI program called swebsock. Swebsock opens a socket to Swebd (the master Caucus server).

4. Swebd spawns a child, called the swebs subserver, which gets the userid from the browser. The subserver is now "dedicated" to this userid, and continues running on its own. The subserver constructs the initial HTML page, and passes it (along with its process id and a unique security code) back to swebsock. Swebsock passes everything back through the HTTP server child to the browser.

This process is illustrated in the following diagram:

3.2 Subsequent requests

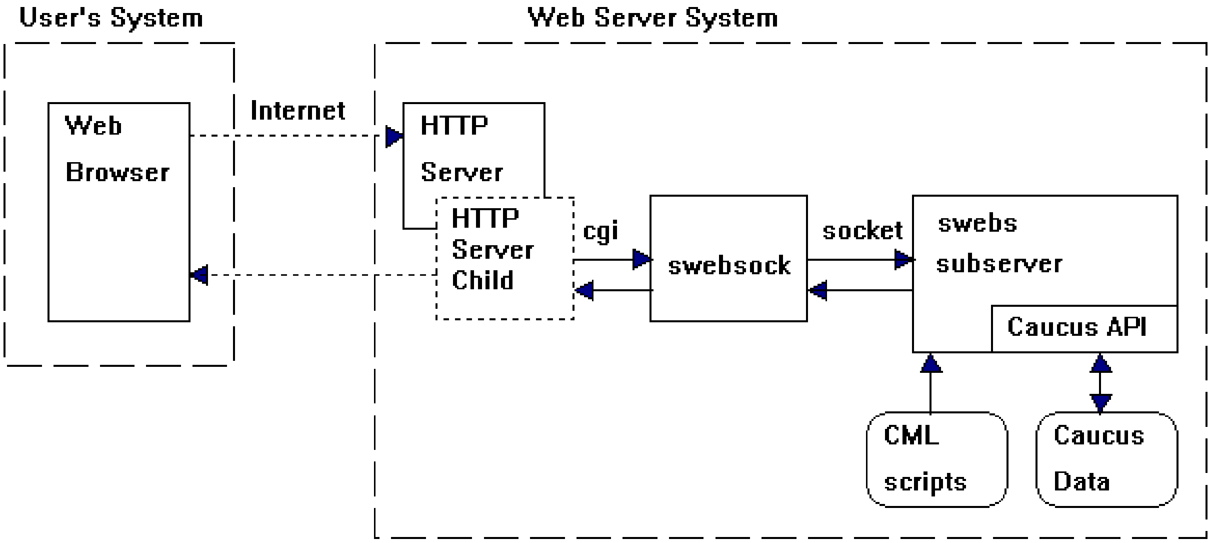
Once the initial connection is made, all subsequent Caucus requests by the browser are passed through to the dedicated swebs subserver. Each such request uses a particular CML script as part of the URL. Such a request will produce the following sequence of events:

1. The browser sends the new request to the HTTP server.

2. The HTTP server immediately spawns (or reuses) a child httpd to handle the request.

3. The httpd child starts a new instance of swebsock, which passes the request on to the dedicated subserver. The subserver reads (or writes) the requested information to the Caucus database, through the Caucus API. The subserver then formats the information according to the codes in the requested CML page, and passes the resulting dynamically created HTML page back through the HTTP server child to the browser.

This process is illustrated in the following diagram:



3.3 Notes

1. In the diagrams, the large dashed boxes are computer systems. The small boxes are processes, and the rounded boxes are disk files. Lines indicate communication paths, either HTTP, CGI (stdin/stdout), sockets, or file reading and writing.

2. Note that *each* browser request involves one or two new processes: the HTTP child, and the CGI swebsock. These processes are kept as lightweight as possible.

3. In contrast, since there is one swebs subserver per user, and each subserver persists across the entire user's browser session, the subservers cache all sorts of information. The subserver also has a timeout period -- i.e., after a certain period with no requests, it exits. Otherwise the system might fill up with inactive subservers.

4. CML: The Caucus Markup Language

4.1 CML Description

The entire Caucus user interface is built out of CML ("Caucus Markup Language") scripts or pages. CML can be thought of as a superset of HTML, with an embedded scripting programming language that is interpreted (by the "swebs" process) *on the server*. Thus, CML pages can not only generate dynamic HTML, but also access the Caucus database on the server, and other files or even programs on the server.

CML as a language contains most of the standard control directives that one would find in any programming language (if/else, loops, etc.), plus a rich set of functions for manipulating web data, Caucus database data, and connections to other programs or files. It is not strictly speaking a superset of HTML (in that it does not understand or parse HTML), but in practice most CML pages contain a large amount of embedded HTML, plus some CML control statements and functions.

CML pages contain 4 kinds of text:

1. **Comments**. In the Unix tradition, all lines beginning with "#" are comments and are ignored. Entirely blank lines are also ignored.

2. **HTML code**. All lines beginning with a single quote (") are parsed for CML functions, but are otherwise passed on (as HTML) to the browser. (The quote is removed.)

3. **CML functions**. Strings of the form "$xyz()", "$xyz(value)", or "$(value)" are parsed and replaced by the appropriate Caucus values. The CML functions are described in the CML Reference Guide.

4. **CML directives**. Directives are like C program code: they describe actions to be taken. Directives include conditional statements ("if" and "else") and loop controls ("for" and "count").

A single logical line may be broken across several physical lines; a "\" as the last character means "continued on next (physical) line". This is generally unneeded, except for HTML <PRE> text that is being built out of mixed text and CML functions.

4.2 CML directives

The CML directives provide some simple control structures recognizable from other programming languages, including:

```
for variable in list

count variable first_val last_val

if condition

else

set variable value
```

For more information, see the CML Reference Guide.

4.3 CML functions

All CML functions evaluate to strings of characters. There is no other data type. The same holds true for CML variables. The CML functions provide access to Caucus data, browser and server control, string manipulation, and logic functions. Again, see the reference guide.

5. Layout of Caucus files

This section describes the layout of the Caucus files -- their location and purpose. All of the files live in or under the Caucus home directory, and (unless explicitly noted elsewhere) should always be owned by the Caucus userid.

Important: If you are editing these files for any purpose, you must do it while logged in as the Caucus userid. In particular, do *not* modify the Caucus files, or run the conference management programs, while logged in as "root" or "administrator".

5.1 CML pages

The CML pages control the precise look and feel of the Caucus web interface. They are all located under the CML directory. As a site may have multiple (distinct) interfaces, each interface gets its own sub-directory under CML. The default set of CML pages is contained in the directory **CML/SP40** (SP for "Screen Porch").

The CML pages are ordinary ascii text files, usually called something.cml, or something.i (for "include" -- files included in other .cml files). Each CML interface (such as **CML/SP40**) also has a special subdirectory called **"Local"**. This contains files that are intended to be changed for your local site, and that will not be touched or replaced the next time you install a Caucus upgrade.

Two particularly important files in the Local subdirectory:

- **switch.i** contains common "switches" that may be set for your site to change how Caucus behaves.
- **l_confs.i** is a list of conference names that will appear under "Popular Conferences" on the Caucus Welcome page.

See the header comments in these files for more information.

In addition to the conferencing interface in **CML/SP40**, there is also a separate (and small) interface in **CML/REG40**. This set of CML pages is entirely dedicated to registering a userid and password for a new user. (It must be a separate interface, because it will be used by people who have not yet gotten or been assigned a userid and password!)

5.2 The SWEB CGI directory

The SWEB directory contains CGI programs and related files that are used to start up the regular Web interface to Caucus.

- **swebd** is the Caucus master server program
- **swebs** is the Caucus "subserver" program
- **swebd.conf** is the configuration file for swebd
- **swebsock** is the CGI program that communicates between httpd and swebd
- **.htaccess** is a file that makes **SWEB** an access-controlled directory (NCSA httpd)
- **cpw1** is a program to modify the httpd password file
- **start.cgi** is a CGI script used to interpret "special" Caucus URLs, such as "http://hostname/caucus/conference_name/item_number.

5.3 The REG CGI directory

The REG directory contains CGI programs and related files that are used to start up the "register a userid" interface. This includes:

- **swebsock** is a copy of or link to the SWEB/swebsock program

5.4 The SOCKET directory

The various Caucus programs (swebd, swebs, swebsock) communicate with each other via a data path called "sockets". The sockets must have a name and a location; therefore they are placed in this directory.

- **sweb** is a socket to master swebd server
- sweb**nnnnnn** is a socket for a particular swebs subserver, process number **nnnnnn**
- **debug**: if this file exists, debugging logs are created for **swebd, swebs,** and **swebsock.**

5.4 The public_html directory

A URL of the form "http://yourhost/~caucus/xyz.html" looks for the file **xyz.html** in the **public_html** directory. (Depending on your httpd server, you may have renamed **public_html** to something else.) Caucus keeps some specific files in this directory:

- **caucus.html** is a simple HTML page to link to Caucus interface (via **SWEB/caucus.cgi**) and to the "register a userid" interface (via **REG/register.cgi**).
- **GIF40** is a directory containing gif and jpeg images used by Caucus interface.

5.5 The BIN2 program directory

BIN2 contains all of the programs used by the Caucus text interface.

- **caucus_x** is the main Caucus 2.7 text interface program (run from the "**cv2**" script)
- **cauchk_x** is the Caucus "check" program, run from the **cv2check** script.
- **caumnt_x** is the Caucus maintenance program, run by the various management scripts (**cv2start**, **cv2remov**, **cv2kill**, etc.)

5.6 The DIC2 dictionary directory

DIC2 contains the source files for the Caucus text interface "dictionary". The text interface is completely customizable, and one site may host many different such interfaces. See Customizing the Caucus 2.7 Interface Guide for more information.

5.7 The GROUPS group permissions directory

Users may be given permission to access specific conferences by individual userid, or by groups of userids. These groups are defined in files in the GROUPS directory. For more information, see the Conference Organizer's "How To" guide.

5.8 The Cnnnn conference directories

The conference data for a particular conference is stored in a single directory. Each conference has a unique four digit number; thus, the data for conference number 1 is stored in the directory C0001.

Conference data is always stored in "flat" ascii text files. In theory this means that the Caucus manager may edit these files directory. In practice you should never do this without specific instructions from Screen Porch technical support staff. This information is provided purely for reference; Screen Porch is not responsible for the results of unauthorized tinkering with these files.

Important files:

- **userlist** contains the permissions list of who may or may not access this conference
- **masteres** is the master list of items and number of responses to each item
- **0010000000** is the text of item 1 (and some responses)
- **0050210000** is the text of item 5, response 21 (and some following responses)
- **introduc** is the conference "introduction"
- **greet** is the conference "greeting"
- **membr001** is the list of conference members
- **variable** contains conference variables (from CML $set_conf_var() function)

5.9 The MISC (miscellaneous) Caucus-wide data directory

MISC contains files that relate to the entire Caucus site, not just a specific conference. As in section 5.8, these files should not be tampered with without specific instruction from Screen Porch. Important files:

- **confs001** is a list of conference names and their equivalent three digit numbers
- **dicti000** is the compiled version of text-interface "dictionary" number 0
- names**nnn** is a list of words in names of registered users, with mapping to their userid
- **bugslist** is a log of possible Caucus "bug" conditions encountered on this host

5.10 USER001, Caucus user files

In addition to the conference-specific files, and the Caucus-wide data files, there is also data stored about each user. Data files for a userid **alpha** are stored under **USER001/alpha**. (Some systems enforce so-called "sanity limits" on the number of sub-directories in a directory; if your system is one of them, Caucus may automatically create directories USER002, USER003, and so forth as needed.)

Important files in each user directory:

- **register** contains "registration" information about this person, including their name, telephone number, brief self-description (introduction), and so forth.
- **p000100** is the participation record in conference 1
- **variable** contains user variables (from the CML function $set_user_var().)

5.11 TEXT001, temporary user files

Temporary files created for each user (for example, during the entry or editing of items and responses) are stored here. It has the same structure as **USER001** (one sub-directory per userid). **Note:** The permissions for this directory and its sub-directories should be write-all.

5.12 Files in the Caucus home directory

There are some Caucus files which do not fit in the purposes described for the previously listed sub-directories. These files are kept in top level of the Caucus home directory.

- **caucus_passwd** is the password file used by NCSA httpd
- **cmi_*** are scripts used by the text-interface to integrate e-mail
- **credit** is a full-screen visual editor supplied with the Caucus text interface
- **credit.doc** contains installation instructions for credit
- **csetperm** is a script to set (or correct) file permissions for most Caucus files
- **cv2** is a standard script to run the Caucus text interface
- **cv2cap** is a script to run the Caucus text interface in "captive" mode
- **cv2check** is a script to check for new information in Caucus conferences
- **cv2kill** is a script to delete Caucus users
- **cv2mkmd** is a script to compile Caucus text-interface "dictionaries"
- **cv2pass** is a script to manage "captive" mode users
- **cv2remov** is a script to delete Caucus conferences
- **cv2start** is a script to create a new Caucus conference
- **expuser** is a script to delete "expired" users
- **fixdate** is a script to update date or "SINCE" information about old conferences
- **fixmaster** is a script to automatically corrected corrupted C**nnn**/masteres files

fixnames is a script to rebuild corrupted MISC/names**nnn** files

- **fixtext** is a script to recreate missing TEXT001 sub-directories
- **listuser** is a script to list potentially "expired" users
- **master.opt** is the master options file for Caucus text interface
- **passprog** is a script to run the cpw1 program to modify caucus_passwd or other httpd password file
- **manager_script** contains the Caucus Management Menu
- **register** is a script to pre-register one or more users
- **swebstop** is a script to stop all running Caucus web-interface processes (swebd, swebs, etc.)
- **testconf** is a script to test consistency of conference item & response data
- **vvtermcap** is a file used by "credit" editor
- **webreg** is a script used to register new web interface userids and passwords

---

Caucus

Conference Organizer

"How To"

Last revised: 4 April 1996

1. Introduction

This "How To" guide describes the details of **how** a Caucus conference organizer administers a conference. For more general information about the **whys** of conference management, see the companion Guide for Conference Organizers.

Note that Caucus has both a World Wide Web interface, and a "text" or command line interface. The instructions in this guide usually refer to the Web interface, although instructions for the text interface are shown in parenthesis at the end of each section.

When the Caucus administrator creates a conference, he or she also assigns someone to be the primary organizer -- the person in charge of the conference. Caucus gives this person special abilities.

The primary organizer may in turn give other people these special abilities in order to share the power and responsibility of managing the conference.

2. Starting a Conference

To create a new Caucus conference you (or the Caucus administrator for your site) must run a companion program called **cv2start**. Login to the "caucus" userid on the Caucus server host, and type "cv2start". The program will ask several questions about the new conference. These include:

* what is the name of the conference?

* what is the userid of the primary organizer?

* should this conference be open to everyone?

* permit this conference like another one?

* make this conference LISTED or UNLISTED?

* should this be a CONFERENCE or a LIBRARY?

* who are the other organizers (if any)?

* which groups should be allowed to use this conference (if any)?

Once you have answered all the questions, you also are given the opportunity to edit the conference user list that controls who may join the conference.

Conference names may be up to 20 characters long but cannot contain any blanks. You may use underscores to link words, as in "MY_CONFERENCE". (Conferences may not be named CHECK, HELP, STOP, or LIST, because these are key commands for the text interface.)

The "CONFERENCE or LIBRARY" question determines the type of the conference. A CONFERENCE is a traditional discussion conference with items and responses. A LIBRARY is a conference that is organized as a file library. File libraries are not currently available for use with the Web interface to Caucus.

Cv2start creates an empty conference with no items, no participants, and a default INTRODUCTION and GREETING. (These terms are defined below.) The organizer should join the new conference as soon as is convenient to prepare the conference for its participants.

3. Customizing a Conference

Caucus gives the organizer of a conference special abilities to assist with setting up and maintaining a conference. Many of these abilities, such as controlling who can join a conference, are provided by the **customize** link in the conference home page (or the CUSTOMIZE command in the text interface.) Only the organizer can use this feature.

Pressing the customize link brings up a form (a page with various check boxes and text boxes) that the organizer uses to modify the conference. This form includes boxes for:

* "**Allow users to add new items?**" This lets the organizer control whether or not members can add new items to the conference. Checked (or "yes") is the default value and means anyone can add an item. Cleared (or "no") means only the organizer can add new items. ("Customize ADD" in the text interface.)

* "**Allow users to edit their own responses?**" lets the organizer control whether or not participants can change the text of their own items or responses. Checked (or "yes") is the default value and means anyone can change an item or response that they entered. Cleared (or "no") means only the organizer can change items or responses. ("Customize CHANGE" in the text interface.)

* "**Make the conference visible to non-members?**" controls whether your conference is "listed" or "unlisted". If your conference is **listed**, its name will appear with the other visible conferences on the Caucus Welcome Page. If your conference is **unlisted**, its name will not appear unless the user is already a member of the conference, or can become a member of the conference. ("Customize VISIBILITY" in the text interface.)

* "**Edit the userlist...**" lets the organizer control who may join the conference. A conference is created with an initial user list; typically one that allows anyone to join the conference. The organizer can edit this list to specifically include people, exclude people, permit read-only members, or add other organizers to the conference.

The user list has a special format which must be followed precisely. Each line in the list contains only one word, either a userid, a group file name, or a control word. The control words are **:include**, **:exclude**, **:readonly**, and **:organizer**. The control words affect the userids or group files immediately following them until the next control word or the end of the list is reached. Here is a simple example:

```
        :include
        harpo
        chico
        :readonly
        zeppo
        :organizer
        groucho
```

The userids "harpo", "chico", "zeppo", and "groucho" are included in this conference. This means that they may join the conference. No one else is allowed to join the conference unless the organizer adds their name to the list. Zeppo can only read the material in the conference. Harpo and chico can both read the material and add their own items and responses. Groucho can do anything that the primary organizer can do.

The user list Caucus displays is numbered in paragraphs and subparagraphs. Caucus numbers this list automatically. The subparagraph number is always 0 for "organizer", 1 for "include", 2 for "readonly", 3 is "exclude". For example if you type in the above user list, the next time you click on the customize link you will see:

```
        :1.1 include
        harpo
        chico
        :1.2 readonly
        zeppo
        :2.0 organizer
        groucho
```

A userid in the user list may contain a terminating asterisk(*) as a wild card. A wild card can replace entering a long list of individual userids. For example:

```
        :include
        smith
        csc*
        :exclude
        csc101
```

means that userid "smith" and any userid starting with the letters "csc" may join the conference. The only exception is userid "csc101" who is specifically excluded from joining the conference.

The third kind of word that may be placed in a conference user list is a group file name. A group file is just a file that contains a list of userids. (See section 4 for more information about group files.) To use a group file in a user list, preface the name of the group file with the character "<". For example:

```
        :include
        <faculty
        :readonly
        <students
```

means that all userids in the group file "faculty" may join this conference, but userids in the group file "students" may only read this conference.

(The equivalent command in the text interface is "Customize USERLIST". It starts a text editor with the contents of the userlist.)

* "**Edit the HTML text of the greeting...**" lets the organizer edit the text of the greeting that appears **each** time a person joins the conference. Note that the greeting can include HTML and CML ("Caucus Markup Language") text. ("Customize GREETING" in the text interface.)

* "**Edit the HTML text of the introduction**" lets the organizer edit the text of the conference introduction. This is the text that appears the **first** time a person tries to join the conference. The introduction should briefly describe the purpose and content of the conference, and who should join it. ("Customize INTRODUCTION" in the text interface.)

4. Group Files

When many people are using Caucus on your computer system, you may find that they fall into distinct groups. For example, at a university you will have students, faculty, administrators, support personnel, and so on. These groups may in turn be divided into sub-groups: engineering faculty, liberal arts faculty, law faculty, etc.

Caucus can help you use these groupings to better control who has access to a conference. That is the purpose of the Caucus "group files". A group file is an ordinary text file that contains a list of userids, one per line. Users listed in a group file are members of that group. The name of the group is the name of the group file.

Group files are useful when a specific group of people need access to several conferences. Without group files, the organizer of each conference would have to edit the user list for that conference and add the userid of each member of the group. With group files, each organizer need add only one line to their user list: a "<" followed by the name of the group.

Group files are created and edited by the Caucus administrator (or anyone who can login as the Caucus administrator). All group files must be contained in the directory called **GROUPS** under the main Caucus directory.

Each line of a group file must contain either a single userid, a wildcard match, or a reference to another group file. A wildcard match must end with an asterisk ("*"). The wildcard match "xyz*", for example, means "any userid that begins with the letters 'xyz'." The third case, a reference to another group file, consists of a "<" followed immediately by the name of a group file. The contents of that file are included as though they were part of the original group file.

This last feature means that you can mimic the groupings and sub-groupings of your organization with Caucus group files. To continue the university example, the Caucus administrator might create a group file called **faculty**, which contains the lines:

```
        <faculty.eng
        <faculty.lib
        <faculty.law
```

The group files **faculty.eng**, **faculty.lib**, and **faculty.law** contain the userids for the faculty members in engineering, liberal arts, and law. Or those groups could be subdivided further. For example, **faculty.eng** might contain:

```
        <faculty.mec
        <faculty.ee
```

These in turn would contain the userids for the mechanical and electrical engineering departments.

Group files may reference other group files, "nesting" indefinitely without limit. Be careful to keep your group files arranged in a hierarchy and not allow any loops. That is, if group file **a** contains "<b", then group file **b** must not contain "<a".

5. Other Functions of the Organizer

The primary responsibility of an organizer is to keep the conference running smoothly. The conference participants expect the organizer to answer questions, monitor the progress of the conference, assist in any communications difficulties, and in general help keep the conference well structured.

As organizer, you may want to structure the first few items of the conference. For example, Item 1 could explain the intents and purposes of the conference, Item 2 could be a place to discuss questions about Caucus, and Item 3 could be reserved for special bulletins or other timely announcements, such as "Class registration begins tomorrow, June 17, at 8:30 am".

The organizer also has the ability to change the text of any item or response in the conference, regardless of who entered it. This ability, however, should be used sparingly. A typical example would be helping a user make the text of his or her item more readable. If an interpersonal problem occurs in a discussion on the conference, as organizer you can intervene or even censor parts of the discussion. Fortunately, such problems are rare.

To change the text of an item or response, simply click on the **edit** button next to that text. Normally this button only appears next to responses that you wrote; but since you are an organizer, it appears next to all items and responses. (Text interface users can CHANGE ITEM or CHANGE RESPONSE.)

If your computer system hosts many different conferences with several organizers, you may want to start a conference specifically for organizers. This is a good way to share information and ideas about how to best set up and maintain a conference.

Caucus

Guide for Conference Organizers

By Stuart Karabenick, Ph.D.

Center for Instructional Computing

Eastern Michigan University

Ypsilanti, Michigan 48197

Last revised: 4 April 1996

This guide was written for the Center for the Instructional Computing and University Computing at Eastern Michigan University, Ypsilanti, Michigan. It has been revised and reprinted with Dr. Karabenick's permission. For more details about how to use Caucus features to start and organize a conference, see the companion guide Caucus Conference Organizer "How To".

This guide is intended for new organizers of Caucus conferences. Prospective organizers may be familiar with conferencing in general, and Caucus in particular; however, there are certain features and issues with which organizers need to be familiar and which deserve special emphasis. These are examined in the following sections. It should be noted that the original guide was written for a large conferencing system in a university setting, and a special section is devoted to that context. Most of the topics, however, are generic, and they apply to a wide variety of settings.

Furthermore, the Organizer's Guide was prepared as a stand-alone document, and it has its own set of organizer-relevant commands (13.9). Thus, Chapter 13, or a version specifically tailored to your hardware, system, and support environment, can be an effective supplement to your conferencing environment.

1. Starting Up

1.1 Types of Conferences

Each conferencing application will have a variety of conferences suited to that environment. In general there are two types.

Open

In an open conference, membership is available to anybody with access to the Caucus conferencing system. Open conferences cover general topics open to all. These can include anything from restaurant reviews and company picnics to office policies, vacation schedules, music, politics, and literature.

Restricted

Restricted conferences impose some limitation on membership. The organizer can specifically designate persons who may become full or read-only members and/or exclude others. Examples of restricted conferences are:

Course-related

In an academic setting one of the major uses is in connection with classes. In the typical course conference, membership is restricted to students and their instructor. Additional non-course participants, such as other faculty members or experts, are sometimes included (see Course Conferencing, section 7).

Special purpose

There are many types of special conferences whose membership may be partially or fully restricted. Examples in an academic setting are: thesis committees, faculty committees, student and faculty organizations and research groups. Business examples are: Boards of Directors, research and development groups, holiday party committees, and sales support groups.

1.2 Obtaining Computer IDs for Participants

Using Caucus requires some kind of computer account or "id". Typically these ids will either be assigned by your organization, or else can be selected by the users themselves. If you as an organizer are running a restricted conference, you will need to know your members id's so that you can identify them.

1.3 Starting a New Conference

In some applications, users are free to start their own conferences. In other settings, a conferencing system coordinator may restrict the number and type of conferences. Restrictions prevent duplication of discussions and conserve system resources. (For information on how to start new conferences, see the Caucus Conference Organizer's "How To".)

1.4 Learning to Use Caucus

It is very important that participants know how to use the system **before** engaging in any "serious" conferencing. Introductory training sessions led by experienced users, practice (fun!) conferences, and provision of quick reference guides to users are suggested.

2. Principles of Conference Organizing

2.1 Creating a General Framework

An organizer's first task is to provide a conference structure, or framework. Considerable time and care at this phase is suggested. The information participants first encounter begins to establish this structure.

Pre-conference Communication

Computer conferencing is often preceded by interaction using non-electronic means. Quite frequently, participants communicate with each other by phone and/or discuss the conference in person prior to any computer communication. These interactions may be augmented by printed material that announces the conference, its goals, topics to be covered, and information about the participants. Such interactions and information are especially important for first-time conference users. The more they know prior to going on-line, the more they can concentrate on mastering the conferencing system and substantive content. Consider the nature and extent of such preliminary communication and how it can help to achieve your conference's objectives.

Conference Interaction

Conferencing users are required to register the first time they use the system. That registration carries over to all conferences of which they become members. Their first interaction with a specific conference consists of an **introduction**, followed by a **greeting**. The organizer customizes them by editing some text inside Caucus.

Introduction

The purpose of the introduction is to describe the conference to prospective participants. It is displayed the first time a participant joins a conference. Its major utility is giving prospective members of open conferences enough information to decide whether they wish to join. For closed conferences the introduction is typically less important since it is presumed that members of a restricted group would already know why they are joining.

Greeting

The greeting is text that is displayed **every** time participants join a conference. The greeting can serve several functions. For example, at the outset, it can serve to orient members by elaborating the conference's goals, purposes and etiquette and rules (see section 5). Later it can be used as a bulletin board for announcements or to direct participants to important new information, items, or responses. Some organizers prefer to keep greetings brief and use a separate item as a bulletin board.

2.2 The First Few Conference Items

The first conference items (discussion topics) have an important bearing on a conference's success. This is especially true when participants are new to conferencing. The following are recommended:

Extended Introductions

The first item gives participants the opportunity to expand upon the information they provided when they first registered in the conferencing system. Consider asking them to describe their background and/or provide other relevant information. This is especially important in larger and open conferences and even in small conferences in instances where participants are relatively unfamiliar with each other.

Purpose(s)

Even if discussed in other forms (e.g., in pre-conference interactions or hard-copy) an item devoted to the conference's purposes is worth including. This is your opportunity to restate the conference's goals and, importantly, to receive feedback from participants. It provides an opportunity for participants to ask questions and to suggest alternatives after having encountered the original introduction and greeting. The item may also be useful in keeping track of changes in objectives as the conference progresses.

Help With the Conferencing System

This item provides a central place to ask questions and serves to reduce the stigma that participants often attach to seeking help. It is especially important for novice conferencers.

Bulletin Board

Even if the greeting is reserved for fast-breaking news, an item devoted to a bulletin board is quite useful. Unlike the conference greeting, past information remains, and there is a record of prior bulletins.

Conference Rules and Norms

Another useful item is one reserved for the discussion of special conference rules or norms. For example, there may be issues of confidentiality, anonymity, adding items or altering previous responses, and rules of conduct (see Etiquette and Rules, section 5) that need to be stated and about which participants may have opinions.

3. Managing Your Conference

3.1 Facilitating Interaction

Starting Discussions

Participants, especially novice conferencers, are understandably reluctant to respond to "blank" items. Thus, a useful technique is for organizers to respond to their own items just to get the ball rolling. For example, an organizer might be the first respondent to the item used for extended introductions or the one used to discuss a conference's purposes.

Respond to Initial Responses

Keep in mind that while you may enjoy discussions and conferencing, some people do not. They may be cautious and embarrassed about stating their own opinions in public, and, quite possibly, intimidated by computers. It helps if organizers respond to participants' contributions either in the conference itself or by sending a private message acknowledging their input.

Developing a Sense of Cohesion

Although physically and temporally separated, regular conference participants can develop a feeling of cohesiveness. This dynamic varies according to the nature of the conference. It is more evident in longer-lasting working groups than in large open conferences, but it is usually present in all conferences to some degree.

Organizers can play an important function in nurturing cohesion. It helps to greet people when they join. Ask them questions. Encourage people who have not contributed to do so rather than just to read. Give participants feedback! Remember, as in face-to-face discussions, participants who are consistently ignored, who feel they are talking to themselves, will cease to contribute.

Lighten Up!

Humor can be an important element of discussions. If not spontaneously generated by participants themselves, consider injecting some in otherwise "serious" conferences. It helps relax people if you break the ice first.

Degree of Organizer Participation

Too many public responses by an organizer can make a conference seem moderator-dominated. Thus, organizers should consider using private e-mail to make constructive comments, to ask a participant why they haven't contributed, or to defuse an argument. Private e-mail does not interfere with conference activity.

Summarizing

Providing summaries is another important organizer function. This helps current participants to quickly understand what has transpired while helping new participants catch up on discussions.

Keeping Things Going

Because computer conferences can extend over long time periods, there are two important maintenance operations. One is to bring in new material to help freshen up conferences. Consider bringing in material from other sources (including from other conferences). The second is to houseclean occasionally by deleting dormant items and keeping subject categories up to date.

3.2 Managing Discussion Topics

How Many Items, How Structured the Conference?

Some conferences have a very well-defined and detailed agenda which should be set by the organizer in advance. For example, a group working on a task (e.g., a new marketing strategy), a course conference, or a committee established to discuss a new program might have specific topics they need to discuss. In these instances the items may be known in advance and the conference structure may be rigid. However, in conferences with more general topics (e.g., office morale, micros or music), it may not be possible, or even desirable, to do this. For open conferences, it is suggested that the initial topic structure consist of a few general items. More specific items typically emerge from those general discussions, and there may be hundreds of items in conferences of a long duration.

Item Drift

An important moderator function deals with what is called "item drift." This occurs when people stray from the topic of an item. You might want to gently (sometimes not so gently) remind "drifters" to return to the topic. Conferences with significant item drift turn out to be "muddy" since the same topic may be discussed in many different items. Some drift is inevitable (do not be too heavy-handed), it is a matter of degree. In fact, participants sometimes signal they are drifting to make a digression (by saying "set drift on" and "set drift off"), indicating that others should not follow their lead. If the drift is significant and raises issues or covers topics not addressed in other items, a new item may be warranted.

Grouping Items Into Subject Categories

Caucus provides the capacity for organizers to group items under subjects. This is an extremely important organizer function, especially in large, open conferences and those of long duration. Like items, these might be thought out in advance and grouped under these headings as they are added. In some conferences with no set agenda, they are likely to develop as the conference progresses. It would also be helpful for you to notify participants of the subjects' existence and explain how to access items by subject categories.

Private E-Mail vs. Public Responses to Items

A conferencing system is designed to facilitate group discussions. Private e-mail would, therefore, seem antithetical to this purpose. Nevertheless, e-mail can serve many useful functions. As in face-to-face discussions, there are some things better said in private. Some communications are simply more appropriate for another individual or subset of the entire group. It is suggested that as much of the communication as possible be conducted in the conference itself (it would not be much of a conference otherwise), while recognizing the need for private communications. The presence of extensive private communication between some people could suggest the need for another conference for those members. For more information on e-mail, see chapter 5.

4. Participant Restrictions

4.1 Adding Items

Typically, conference participants are permitted to add their own items. There are, however, circumstances in which this may be undesirable. This is especially true in newly organized conferences, when it may be beneficial for the organizer to maintain control of the topics and/or the order in which they are discussed. There may even be conferences where the organizer wants to completely control the conference items, such as in computer-mediated business meetings and, in educational settings, course conferences. Once conferences have matured, an organizer may wish to relax this restriction. Note that open conferences would probably not survive this restriction for very long.

4.2 Altering Previously Entered Conference Material

Unless you decide otherwise, participants are permitted to change (i.e., edit, replace, or delete) material they have previously entered (items or responses). This is useful when, in retrospect, they are not content with something they have said. However, there may be circumstances when allowing such changes would be inappropriate. For example, in a group working on a sensitive topic, retrospective changes could significantly alter the context in which subsequent remarks are embedded, changing their meaning entirely. It is suggested that restricting the right to make such changes should be used with caution and only with the consent of the participants. Of course, you can always reverse the restriction. Restricting changes is ordinarily not appropriate for public conferences.

4.3 Names: Real and Pseudo

Except for duplications of names already registered, participants can select any name they desire. Thus, pseudonyms are possible and can be used creatively. For example, names can be used for role-playing, or groups of individuals can select similar names for simulations. However, under some circumstances they may be inappropriate, as in business settings or in course conferences where it is necessary to track participation. Furthermore, pseudonyms should be used responsibly and not to harm or impersonate other conference members. Remember, the identity of the author of any item or response can be discovered despite the use of pseudonyms.

5. Etiquette and Rules

5.1 Remedies for Violations

Organizers of open conferences need to be especially sensitive to objectionable content. After all, participants' comments are available to anyone with access to your Caucus system. The same basic guidelines that apply to free speech using any other medium apply here as well. There are two minimal rules that should be adhered to in all public conferences: no vulgar language and no personal attacks.

There are several ways to handle problems:

* Delete any offensive material (items or responses).

* Ask the person to apologize for offensive remarks.

* Send a private e-mail or speak directly to the participant involved.

* Publicly chastise the participants in the conference.

* Exclude members from the conference and/or system if necessary.

5.2 Confidentiality

Confidentiality is frequently an issue any time people communicate. However, since computer conferencing creates an instant transcript, breaches of confidentiality become markedly simplified: it is relatively simple to print material, copy it to another conference, or publish it in some other fashion. Thus, computer conferencing provides a greater potential for abuse.

It is typically assumed that conference material is intended only for other participants. Reproducing that material for wider distribution would violate that assumption. However, if material in a conference is of a particularly sensitive nature, you might wish to:

* Discuss the issue of confidentially with participants

* Place a notice in the introduction and/or greeting about the confidentiality policy

* Suggest that persons who specifically wish should emphasize confidentiality in their communications

6. Winding Down

6.1 Archiving Your Conferences

Keeping a permanent record of a conference is highly recommended. This is especially true for course and special purpose conferences that you may wish to review after they have ceased to exist. One option is to print a hard copy. Another is to write the conference to a file (see print and file transfer commands, in the Caucus 2.0 User's Guide.)

6.2 Terminating Conferences

Open conferences run continuously but are typically restarted periodically (with much advance notification to participants) to conserve computer resources. Conferences established for specific purposes (e.g., task groups) have a definite life. In educational settings, course conferences normally terminate when the term closes. Other special conferences may be indeterminate. In each case, the organizer should notify the conferencing system coordinator when to terminate the conference.

7. Course Conferencing In Educational Settings

7.1 Why Course Conferences?

In course conferencing, a class is provided with one or more of its own closed conferences. When used with on-campus classes, add communication possibilities beyond those which normally exists. Course conferencing also can be used to teach complete courses by computer-mediated communications (known as virtual classrooms).

By opening up additional communication channels, conferencing can increase access between instructors and students, and among students. Conferencing also has the potential to significantly increase the amount of writing by students, even in courses where writing is neither an essential nor even a minor component. Parenthetically, using conferencing helps satisfy two goals which colleges and universities typically attempt to foster among their students: familiarity with computers and increased written communication.

7.2 Important Benefits for Classes and Students

* Students can become more familiar with their classmates and instructors.

* Instructors can post assignments in the conference - using it as a bulletin board.

* Unresolved class discussions can be continued in the course conference.

* Students can communicate with each other and their instructor outside of class without having to coordinate their schedules. Electronic "office hours" are very efficient.

* Unlike class discussions, a written record is maintained and available for later review by both students and instructors.

* Students have time to think before they respond to what others have said.

* Students can enter and edit their remarks without taking up each other's time.

* Instructors and students can engage in many simultaneous discussions and private conversations without interference with each other.

* Students can get rapid response to their ideas.

* Students who might rarely or never speak in class are more likely to contribute in a conference.

* Aggressive students are less likely to dominate discussions as they might during class.

7.3 Some General Considerations

It is important to recognize that communicating via computer and conferencing are probably new experiences to many, if not most, students. Therefore, an introduction in class to the general principles of both is important prior to any workshops or other hands on experience. Equally important is that students understand why conferencing is being included in the course. Indicate how much you expect them to participate. Also, clarify for students the use of private e-mail.

While this information also may be presented and discussed in the conference itself, creating a context for conferencing can go a long way toward allaying anxieties that accompany this experience. As conferencing becomes more common, such introductions will probably cease to be as necessary.

7.4 Suggestions for Items and Uses of the Conference

Here are some specific ways of using a course conference:

* Follow up unfinished or unresolved class discussions by continuing them in the conference.

* Enter items to discuss major issues and concepts in the course. These need not be in place when the course begins but may be added as the course progresses. The more provocative and challenging the better.

* Use the conference to receive feedback on lectures and assignments.

* Make specific assignments in the conference during the course. Post assignments you are contemplating and ask students for input before assignments are finalized.

* Use the conference to post exam keys and answers and to discuss exams. Considerable time and energy is saved when everyone in the course can partake in these discussions -especially students who may be absent when the exam is discussed.

* Consider inviting "outside experts" to your conference to enrich the course. They could be other faculty and staff or off - campus experts. Note that special arrangements would have to be made for the latter - they can be given special Caucus accounts (i.e captive accounts).

* Consider a simulation. Set up a situation in which class members take certain roles. This can be a major use

of conferencing. For example, business courses might simulate communications between managers and employees. A course could have an additional conference established for the students for this purpose.

* Written work could be entered in a conference and "critiqued" by other class members. Text can be entered and revised by others, and the class would have a record of successive revisions.

* Divide the class and have an extended "debate" on an issue. The conferencing medium gives the debaters more time to consider their responses and perhaps consult resources.

* Conduct polls on specific issues.

7.5 Additional Considerations and Suggestions

Here are some further issues to consider when using course conferences:

* As noted earlier, conferences may be set up so that only the organizer can enter items. It is strongly suggested that this be done at the beginning of the term. This restriction can be relaxed once students have more experience with the course and with conferencing.

* Consider whether conferencing should be a required or voluntary activity. Especially if required, consider further the degree of access that students will have to the timesharing system (i.e. computer terminals or microcomputers with communications capability). For example, commuting students, especially those on campus only in the evening, would have less time to conference than on-campus residents.

* Use the conference for small group projects. It is possible to establish additional closed conferences for this purpose. The instructors can be a consultant to several group projects efficiently if s/he is a member of each project's conference.

* Consider storing text material you plan to use frequently in computer files. This material can then be imported into the conference as needed. For example, a newspaper article which might be the stimulus for a discussion could be typed into a file in your area. The article is then available for the class whenever you desire, e.g., by making it an item. Having several files available gives you an on-line course-pack. Computer files could even be shared among colleagues.

* Students should have introductory hardcopy material about conferencing.

* Workshops can be set up and made available at the beginning of each term to teach new students and faculty how to use Caucus computer conferencing - an instructor who plans on using conferencing extensively may want to arrange to have a workshop for the first scheduled class meeting.

* Ask students who are familiar with conferencing to act as tutors to others.

* Perhaps the most general statement about moderating a course conference is: treat it as you would the class itself. That is, guide but do not dominate. Do not think that you have to answer every question that arises. Give students a chance to "talk", or they will just read. Enter items gradually so as not to overwhelm students.

* Finally, students will only use conferencing if there is a reason for doing so. While you can make conferencing mandatory, that could engender some resistance. Instead, suggest guidelines, such as checking in on the conference at least three or four times per week. Voluntary conferencing can be successful if:

- important information exists on the conference

- conference discussions are truly interesting

- there are some purely entertaining or social items as well as "serious" topics

- conferencing is undertaken in a relaxed atmosphere

- students feel free to ask for help with the system

7.6 Evaluating Conferencing Activity

An important advantage of course conferencing is the capacity to effectively evaluate students' contributions to discussions. This can be done either by scrolling through the items on your terminal or computer screen, or by printing the conference (see print commands).

The way to evaluate contributions depends on your course objectives and expectations for conference participation. Here are some typical conferencing standards:

Have Students Accessed the Conference and Its Items?

At the very least, students might be expected to read the material in the conference. Caucus permits checking on the items and responses that students have displayed. If this is done once per week, a record could be kept and referred to at the end of the term.

Minimal Responding

A second level requires that students join the conference and contribute to some or all of the items. This could be checked by examining the conference transcript. By using the information tagged to each response one could determine whether these contributions occurred during a specific time interval, say once per week.

Beyond the Minimum

Levels of contribution beyond these minima can be gauged only by closer examination of the conference transcript. One criterion is response length. Although as with any other contribution this may not be the best measure, it can be used as a rough index of student participation. One suggestion is to use three levels, something like terse, average, and extensive. A one-line response may indicate the student's participation but not much else. Two or three sentences is usually enough to justify more extensive interaction, while a ten-line response may signify extensive participation.

Content Analysis

How incisive and meaningful students' contributions to discussions are can be only determined by carefully examining an entire course transcript. How one does this depends, again, on one's course objectives. As with any grading scheme the metric can range from global to fine-grain, from acceptable vs. unacceptable to a specific letter grade (complete with + and -). The advantage of having the complete conference transcript over attempting to do this with in-class participation should be obvious. Instructors who repeat courses have the additional advantage of making between-term comparisons.

8. Helpful Hints

Here are some additional suggestions that others have found useful:

Foreign Language Conferencing

It is possible to conduct conferences in languages other than English by adopting text conventions. The creative use of punctuation marks and symbols can substitute for many accents. The greeting or first conference item should be used to establish the conventions.

Indexing With Creative Item Titles

Items can be selected in database fashion with the judicious use of item titles. Suppose there are several working groups producing several versions of documents in one conference. If each group's document was entered as a separate item that carried the group (e.g.,G1) and version number (e.g.,V1) the **list some items** link from the conference home page could be used to show all documents for group 1. Other standard information contained in item titles would be similarly searchable.

9. Important Organizer Features

9.1 Conference Membership

Each conference maintains a userlist which controls who has membership privileges. This list can be edited from the **customize** link on the conference home page (or the "Customize USERLIST" command in the text interface.)

Completely Open Conferences

A completely open conference should have a colon (:) followed by the word "include" as the first line in the file and no other text. The second line should contain an asterisk (*). Therefore, the file would look like this:

```
:include

*
```

Restricted Conferences

To specify a list of permitted participants, the asterisk should be replaced by their user IDs. The following userlist specifies three faculty members and a student (fox).

```
:include

psy_karabeni

csc_remmers

ori_young

fox
```

Use of "Wild Cards"

To specify a group of participants with a common account name, such as a set of course accounts, enter the common element, then an asterisk which is called a "wild card". The following userlist would permit a faculty member and all students in his chemistry class. Student IDs would all begin with the course prefix, then their student number (e.g., chm610566475). The common element is chm610.

```
:include

chm_ramsay

chm610*
```

Excluding or Limiting Participants

Participants are excluded by listing their account names, following a line which reads ":exclude." In addition, participants can be limited to only being able to read (not contribute to) material by listing their account names following a line which reads ":readonly". The following userlist would permit all faculty members in a department to participate, give an invited guest (gendin) permission to read the discussions, and exclude one department member (orloff) from accessing the conference at all.

```
:include

psy_*

:readonly''

phi_gendin

:exclude''

psy_orloff
```

9.2 Customizing Conference Characteristics

Customize the Introduction

The conference introduction may also be edited from the "customize conference" page. Modify it to suit your needs for the particular confernce. For example:

```
Welcome to the Music conference.  Its purpose is to discuss all facets of the musical scene, both
classical and contemporary.  We welcome membership and participants.
```

Customize the Greeting

The greeting appears every time a person sees the conference home page. As with the introduction, text already exists upon the conference's creation. Modify it as you see fit. An example:

```
Note that the items in Music are in organized by categories, according to their title.  To see these
categories and the items under each, click on list some items, and then pull down the menu bar to
"by words in title". See item 26 for a discussion of the next assignment.
```

Allow users to add new items?

The organizer may instruct Caucus to permit or deny conference participants the right to add their own items. The default is to permit adding items.

Allow users to edit their own responses?

Caucus gives the organizer the option of not permitting participants to alter, replace, or delete the text of items and responses they have entered. The default setting permits such changes.

Screen Porch: *Virtual Hosts & Caucus*

This page describes how to use WWW "Virtual Hosting" in combination with Caucus, to supply different Caucus interfaces for different virtual hosts on a single host computer.

"Virtual Hosting" is a feature of the Apache httpd server (and possibly others). It provides a way to make a single host computer look like different, multiple, hosts. Each such "virtual" host, when accessed through a web browser, has its own distinct web document tree.

Each of those virtual hosts can also have its own distinct Caucus interface, as well. And yet the virtual hosts will share the same installation of Caucus, and potentially the same set of conferences.

The rest of this page describes, by example, how to implement this. This discussion assumes basic familiarity with the Apache httpd server, and experience installing Caucus on a single web host.

Section I: Virtual host setup.

Assume two hostnames, "gamgee.host.edu", and "frodo.host.edu", both of which point to the same machine (originally called "original.host.edu"). The httpd server is installed in the directory `/caucus/HTTPD`.

To implement virtual hosting in the httpd server, modify the httpd server configuration file httpd.conf to include these entries for the two virtual hosts:

```
<VirtualHost gamgee.host.edu>
DocumentRoot /caucus/HTTPD/GAMGEE
ServerName   gamgee.host.edu
</VirtualHost>

<VirtualHost frodo.host.edu>
DocumentRoot /caucus/HTTPD/FRODO
ServerName   frodo.host.edu
</VirtualHost>
```

Make sure to remove any other definitions of DocumentRoot and ServerName elsewhere in the configuration file(s).

Note that each virtual host has its own document tree, in `/caucus/HTTPD/GAMGEE`, and `/caucus/HTTPD/FRODO`, respectively.

Section II: HTML files for starting Caucus

Assume that Caucus is already installed in `/caucus`, and that it is just being modified to allow virtual hosting.

Further, assume that the virtual hosts are **only** used for Caucus, so the DocumentRoot index.html files for each virtual host can take the user's browser right into Caucus.

The principle here is really quite simple. Starting at the top (the browser's entry point into each virtual host), replace all links to the generic host, with links to specific URLs **or CGI files** that reference the desired virtual host. Continue following the chain of links and CGI files until the CML files also point to the correct hosts.

Start by copying

```
     /caucus/public_html/caucus.html  to
     /caucus/HTTPD/GAMGEE/index.html,
```

and likewise for FRODO.

Edit the link in the index.html file to point to the appropriate host and unique CGI files. For example, in GAMGEE's index.html file, the lines:

```
   If you already have a userid and password, go to the
   <A HREF="http://original.host.edu/sweb/caucus.cgi">
   Caucus Welcome Page</A>.
```

are changed to reference

```
   <A HREF="http://gamgee.host.edu/sweb/gamgee.cgi">
```

Do the equivalent change for FRODO's index.html file.

If the virtual hosts should have self-registration of Caucus userids, make the equivalent changes for that link as well, in FRODO and GAMGEE's index.html files.

Section III: CGI files for starting Caucus

The HTML files in the previous section referenced (new) unique CGI files for starting Caucus. Now create them. Copy

```
     /caucus/SWEB/caucus.cgi  to
     /caucus/SWEB/gamgee.cgi
```

Edit gamgee.cgi, and change:

```
   echo "Location:
http://original.host.edu/sweb/swebstart.cgi/SP/Local/start.cml"
```

to

```
   echo "Location:
http://gamgee.host.edu/sweb/swebstart.cgi/GAMGEE/Local/start.cml"
```

Note that this assumes the creation of a separate set of Caucus interface files under `/caucus/CML/GAMGEE`, to replace the default `/caucus/CML/SP` directory.

Make the equivalent changes for `/caucus/SWEB/frodo.cgi`, and (if self-registration is allowed), `/caucus/REG/gamgee.cgi` and `/caucus/REG/frodo.cgi`.

Section IV: CML files for each virtual host's interface

Finally, create the actual new interfaces for each virtual host. Copy **all** of the files (and directories) under `/caucus/CML/SP` to a new directory, called `/caucus/CML/GAMGEE`.

Then edit `/caucus/CML/GAMGEE/Local/start.cml`, and change:

```
   set dir    SP
```

to

```
   set dir    GAMGEE
```

and

```
   set href   http://$host()/sweb/swebt.cgi/$pid()/$(dir)
```

to

```
   set href   http://gamgee.host.edu/sweb/swebt.cgi/$pid()/$(dir)
```

Make the equivalent changes for `/caucus/CML/FRODO`.

If self-registration is allowed, create duplicates of the `/caucus/CML/REG` directory, perhaps REGF (for FRODO) and REGG (for GAMGEE). In those directories, make the same changes as shown above, in the respective register.cml file(s).

That's It!

Presumably some other aspect of GAMGEE's or FRODO's CML interface files will also be changed (otherwise, why have different interfaces in the first place?).

Instructions for performing common interface modifications (such as changing the Caucus logo, or the page background) can be found in the Caucus Installation Guide.

---

1 ./MODULES/caucusmail.html

Caucus E-mail Interface Installation and Usage Guide

Copyright (C) 1996 Screen Porch LLC. Last Revised: 11 June 1998

1. INTRODUCTION AND PURPOSE

The Caucus e-mail interface package adds a "listserv" or mailing-list like capability to the existing Caucus conferencing system. (Currently this package only works with Unix platforms, although the CML scripts may be adaptable to Windows/NT platforms, if a command-line mailer client is available to replace the Unix 'mail' program.)

With this optional package, you can extend the use of your Caucus conferencing system to people who have only e-mail access to the Internet.

When this package is installed and enabled, each Caucus conference organizer can specify a list of e-mail addresses that may participate in that conference. New material (items and responses) are automatically sent to those participants, via e-mail, on a regular basis.

Those users may in turn contribute to the conference by simply replying to those messages. The replies are automatically placed in the proper conference and item.

2. INSTALLATION

2.1 E-mail Kit Contents.

The Caucus E-mail link kit is contained in a file called "email.tar" that may be downloaded directly from the Screen Porch web site, at http://screenporch.com/MODULES/email.tar.

The email.tar file contains this README file, the 'einstall' installation script, and a compressed kit file, kit.t.Z.

2.2 Create the Caucus Mailer userid ("caumail").

Create a Unix userid that is dedicated to handling the e-mail for this interface. (The 'root' user or system administrator must do this.) A good name for this userid is "caumail", although any userid will work. (Do not use the regular "caucus" userid for this account. This must be a separate userid that is only used for this purpose.)

This userid must be able to use the Unix 'crontab' utility.

2.3 Install the software.

Login to the id you created in step 2.2. Do NOT install the software as root! Download or copy the email.tar file to the home directory of that id. Type:

```
tar xvf email.tar
   ./einstall
```

Follow the instructions that are displayed.

Initially the e-mail updates will be sent out once per day. This may be changed by examining and modifying the contents of the "crontab" listing for this userid.

2.4 Connect the e-mail link to Caucus.

To finish the installation, you must "connect" the Caucus e-mail link software with your regular Caucus installation.

Login to the "caucus" userid. From this id, run the script called "copysweb" that is located in the Caucus Mailer userid's home directory. (For example, if the Caucus Mailer userid's home directory is /home/caumail, then type "/home/caumail/copysweb".)

Now edit the file CML/SP40/Local/switch.i, and change the definition of the "mail_out" variable to be the Caucus Mailer userid (for example, "caumail").

3. CONFERENCE ORGANIZER INSTRUCTIONS

To allow e-mail users to participate in a conference, the conference organizer must do two things from the "customize" page:

3.1 Include the Caucus Mailer userid in your conference.

Add the Caucus Mailer userid from step 2.2 to the list of users included in your conference. This only needs to be done once.

3.2 Add individual e-mail users.

For each e-mail user that is participating, add their e-mail address to the "Section IV: E-mail participants" box at the bottom of the customize page.

Note that this must be the address that appears on mail sent **from** the user. Caucus uses the entries in the E-mail participants box for two purposes: to determine who to send mail **to**, and to control who mail will be accepted **from**.

This is somewhat subtle point. A person with simple "mail to" address may actually have a longer "from" address. You **must** use the "from" address. (Some people may also have multiple e-mail aliases that all point to their "real" e-mail address. In either case, you must always use the "from" address that appears in their replies.)

To remove an e-mail participant, simply delete their address from the box. (There is no way to "rename" an e-mail participant to another e-mail address.)

4. E-MAIL PARTICIPANT INSTRUCTIONS

When an e-mail participant is added to a conference (in step 3.2), they will receive the entire contents of the conference as e-mail. Each item will appear (with all of its responses) as one message. The subject heading of the message begins with "::Caucus", and then shows the conference name, item number, response numbers, and item title.

Thereafter, as new items and responses are added to the conference, e-mail participants will receive regular updates (typically daily). All new responses to an item will be delivered as one message. Each new item (with its responses so far) will be delivered as one message.

An e-mail participant may add a response by simply replying to the appropriate message. A reply to a particular message will be posted as a response to that item.

E-mail participants may post HTML responses, by making the first word of their response be "<HTML>". (It must be followed by a space or a return.)

E-mail participants may post new items by replying to any message (from the relevant conference), and changing the subject field to remove the item and response numbers. (I.e., the subject field should just contain the "::Caucus" and the conference name.) On most mailers, this can easily be accomplished by simply backspacing over the subject until the conference name is reached.

The first line of the message will be used as the item title. If the first word of the title is "<HTML>", then the entire item text will be treated as HTML.

5. APPEARANCE OF E-MAIL POSTINGS IN A CONFERENCE

Items and responses posted by e-mail participants look just the same as entries made by regular Caucus users.

The only exception is the name of the participant. The name will appear as plain text, typically followed by their e-mail address, shown "blued" as a link. (Since Caucus doesn't know anything else about them, only the e-mail address is active.)

Conference organizers can delete or edit the participants' items or responses in the usual way.

6. POTENTIAL PROBLEMS

There is one known problem, having to do with e-mail replies.

For many mailers, when a user replies to an e-mail message, the content of the original message is made part of the reply, with a "> " before each line (to distinguish it from the reply proper).

The Caucus e-mail package understands this syntax, and strips all such lines from the text before adding it as a response.

However, some mailers use other methods of marking the lines from the "original text". As these methods are identified, those lines should also be stripped out! (Otherwise a potentially exponential growth may apply, as replies to replies to replies etc. get posted in the conference.)

See the section in the file import.cml in the Caucus e-mail package for more information about how to accomplish this stripping.

Caucus E-mail Interface
Installation and Usage Guide

Copyright (C) 1996 Screen Porch LLC.
Last Revised: 11 June 1998

1. INTRODUCTION AND PURPOSE

The Caucus e-mail interface package adds a "listserv" or mailing-list like capability to the existing Caucus conferencing system. (Currently this package only works with Unix platforms, although the CML scripts may be adaptable to Windows/NT platforms, if a command-line mailer client is available to replace the Unix 'mail' program.)

With this optional package, you can extend the use of your Caucus conferencing system to people who have only e-mail access to the Internet.

When this package is installed and enabled, each Caucus conference organizer can specify a list of e-mail addresses that may participate in that conference. New material (items and responses) are automatically sent to those participants, via e-mail, on a regular basis.

Those users may in turn contribute to the conference by simply replying to those messages. The replies are automatically placed in the proper conference and item.

2. INSTALLATION

2.1 E-mail Kit Contents.

The Caucus E-mail link kit is contained in a file called "email.tar" that may be downloaded directly from the Screen Porch web site, at http://screenporch.com/MODULES/email.tar.

The email.tar file contains this README file, the 'einstall' installation script, and a compressed kit file, kit.t.Z.

2.2 Create the Caucus Mailer userid ("caumail").

Create a Unix userid that is dedicated to handling the e-mail for this interface. (The 'root' user or system administrator must do this.) A good name for this userid is "caumail", although any userid will work. (Do not use the regular "caucus" userid for this account. This must be a separate userid that is only used for this purpose.)

This userid must be able to use the Unix 'crontab' utility.

2.3 Install the software.

Login to the id you created in step 2.2. Do NOT install the software as root! Download or copy the email.tar file to the home directory of that id. Type:

```
    tar xvf email.tar
    ./einstall
```

Follow the instructions that are displayed.

Initially the e-mail updates will be sent out once per day. This may be changed by examining and modifying the contents of the "crontab" listing for this userid.

2.4 Connect the e-mail link to Caucus.

To finish the installation, you must "connect" the Caucus e-mail link software with your regular Caucus installation.

Login to the "caucus" userid. From this id, run the script called "copysweb" that is located in the Caucus Mailer userid's home directory. (For example, if the Caucus Mailer userid's home directory is /home/caumail, then type "/home/caumail/copysweb".)

Now edit the file CML/SP40/Local/switch.i, and change the definition of the "mail_out" variable to be the Caucus Mailer userid (for example, "caumail").

3. CONFERENCE ORGANIZER INSTRUCTIONS

To allow e-mail users to participate in a conference, the conference organizer must do two things from the "customize" page:

3.1 Include the Caucus Mailer userid in your conference.

Add the Caucus Mailer userid from step 2.2 to the list of users included in your conference. This only needs to be done once.

3.2 Add individual e-mail users.

For each e-mail user that is participating, add their e-mail address to the "Section IV: E-mail participants" box at the bottom of the customize page.

Note that this must be the address that appears on mail sent **from** the user. Caucus uses the entries in the E-mail participants box for two purposes: to determine who to send mail **to**, and to control who mail will be accepted **from**.

This is somewhat subtle point. A person with simple "mail to" address may actually have a longer "from" address. You **must** use the "from" address. (Some people may also have multiple e-mail aliases that all point to their "real" e-mail address. In either case, you must always use the "from" address that appears in their replies.)

To remove an e-mail participant, simply delete their address from the box. (There is no way to "rename" an e-mail participant to another e-mail address.)

4. E-MAIL PARTICIPANT INSTRUCTIONS

When an e-mail participant is added to a conference (in step 3.2), they will receive the entire contents of the conference as e-mail. Each item will appear (with all of its responses) as one message. The subject heading of the message begins with "::Caucus", and then shows the conference name, item number, response numbers, and item title.

Thereafter, as new items and responses are added to the conference, e-mail participants will receive regular updates (typically daily). All new responses to an item will be delivered as one message. Each new item (with its responses so far) will be delivered as one message.

An e-mail participant may add a response by simply replying to the appropriate message. A reply to a particular message will be posted as a response to that item.

E-mail participants may post HTML responses, by making the first word of their response be "<HTML>". (It must be followed by a space or a return.)

E-mail participants may post new items by replying to any message (from the relevant conference), and changing the subject field to remove the item and response numbers. (I.e., the subject field should just contain the "::Caucus" and the conference name.) On most mailers, this can easily be accomplished by simply backspacing over the subject until the conference name is reached.

The first line of the message will be used as the item title. If the first word of the title is "<HTML>", then the entire item text will be treated as HTML.

5. APPEARANCE OF E-MAIL POSTINGS IN A CONFERENCE

Items and responses posted by e-mail participants look just the same as entries made by regular Caucus users.

The only exception is the name of the participant. The name will appear as plain text, typically followed by their e-mail address, shown "blued" as a link. (Since Caucus doesn't know anything else about them, only the e-mail address is active.)

Conference organizers can delete or edit the participants' items or responses in the usual way.

6. POTENTIAL PROBLEMS

There is one known problem, having to do with e-mail replies.

For many mailers, when a user replies to an e-mail message, the content of the original message is made part of the reply, with a "> " before each line (to distinguish it from the reply proper).

The Caucus e-mail package understands this syntax, and strips all such lines from the text before adding it as a response.

However, some mailers use other methods of marking the lines from the "original text". As these methods are identified, those lines should also be stripped out! (Otherwise a potentially exponential growth may apply, as replies to replies to replies etc. get posted in the conference.)

See the section in the file import.cml in the Caucus e-mail package for more information about how to accomplish this stripping.

Collaborate

The Screen Porch Newsletter on Caucus a nd social computing

Issue #1
February 1998

Happy New Year to you—the Caucus community and friends of Screen Porch. 1998 will be an important year for social computing, and we all have something to contribute.

—Tom Mandel

Stories

1. Caucus 4.0—it'll be here soon
2. Success Stories—Creating the Future with Caucus
3. Tech Tips—understanding and managing Caucus userlists
4. Website Redesign—more information, easier to find
5. Take Caucus for a Test Drive
6. Keeping Your Maintenance Contract Up-To-Date

Instructions for subscribing/unsubscribing may be found at the end of the newsletter. Please send your comments on Collaborate! to editor@screenporch.com.

Announcing Knowledge Ecology Fair 98

Knowledge Ecology Fair 98 showcases companies creating shared work environments on the net. Participants from Intel, Hughes Space & Communication, Sun Microsystems and Skandia will report on experiences in knowledge ecology, the creation of Web communities among employees to develop, share and harvest best practices. This vitual Fair is produced by Metasystems Design Group and Community Intelligence Labs (CoIL).

"This unique web event will give executives first-hand experience of a collaborative on-line culture," comments George Por, founder of CoIL. The month-long on-line conference started February 2. It features keynotes from leading thinkers and authors, teleworkshops, and informal conversations among experts and participants. Full information on the Fair and participating companies can be found at http://www.co-i-l.com/kefair.

The Knowledge Ecology Fair is powered by Caucus!

1. Caucus 4.0—The most powerful & flexible Caucus yet

Caucus 4.0 is coming soon. Expect a host of enhancements and a new look. You'll see a new browser-based module for managing Caucus and organizing and administering conference workspaces. CML is still the most powerful Web-based language for building collaborative spaces online, and now it includes easy ways for you to integrate all kinds of knowledge resources into your Caucus conference workspaces. Your users will enjoy the new Caucus Notebook, where they can manage and organize knowledge created in Caucus. Stay tuned for more information and an announcement of the release date.

Note: special Caucus pricing is available before March 1998. Contact sales@screenporch.com for more info.

2. Success Stories—Creating the Future with Caucus

Look for case studies and success stories in upcoming issues of Collaborate!—you will read about government agencies, Fortune 500 companies, and major educational institutions where Caucus is making a difference and helping you and people like you work together to create the future. If you've got a success story to share, please email our editor@screenporch.com

3. Tech Tips—understanding and managing Caucus userlists

Caucus gives you powerful tools to manage the userlist for a conference space quickly and with a fine level of control. Lets say you are creating a conference to plan your annual offsite marketing meeting in Tahiti and want to include marketing, but exclude the strategy group within marketing. Still, you *do* want Mary Walton (mwalton), of the strategy group, to participate. Here's how to build the userlist:

```
:include
<marketing
:exclude
<strategy
:include
mwalton
```

("<" indicates a Caucus group) With this power comes the need to think carefully to get the result you want. For example:

```
:include
<marketing
mwalton
:exclude
<strategy
```

won't include Mary Walton. Caucus includes her by name, but then excludes her as a member of Strategy! Some thought and planning is required.

4. Website Redesign—more information, easier to find

Major changes are coming to our web site, including a new design and public conferences. There'll also be a conference space devoted to issues of social computing and online collab- oration. We'll invite speakers to join these conferences and discuss their activities and applications. Of course, we will continue to invite all to join our demonstration conference and try out Caucus.

5. Take Caucus for a Test Drive

If you are starting to feel the urgency to support virtual teams, enable communities of practice, or create learning environments online, there is no better time than now to download a trial version of Caucus and find out what all the excitement is about. The trial version is a fully-functional, easy-to-install copy of Caucus. The only difference between trial and purchased versions is that the trial version times out in thirty days.

6. Keeping Your Maintenance Contract Up-To-Date

A current maintenance contract will ensure that you receive the new version of Caucus at no additional charge. Please contact sales@screenporch.com or call directly at 760-751-2142 if you have any questions or need information about the status of your maintenance contract. When you need support under your Caucus maintenance contract, please have one of the two technical contacts you designated contact our support department directly by email or phone. We're looking forward to helping you.

Screen Porch Contacts:

http://screenporch.com

sales@screenporch.com
support@screenporch.com
corporate: 703-243-3001
sales: 760-751-2142
fax: 760-751-4228

Collaborate! will appear approximately once a month. Please email requests to subscribe or unsubscribe to editor@screenporch.com. Thanks.

Open for Business℠

—Caucus Case Studies

Who We Are · Caucus · On the Porch · Screen Porch

Department of Defense Plans Military Health Care using Caucus

Over four hundred experts worldwide create system for the year 2020 in online Caucus conferences.

Educational Institute Selects Caucus for Worldwide Online Education Program

Innovative Virtual Campus goes live using Caucus discussion spaces

Caucus Helps TransCanada Create Scenarios for Successful Future

Senior management team planned global future in Caucus conversation spaces

Caucus Helps Merged Organizations Work As One

Senior managers of Avery Dennison plan unified corporate future using Caucus

Defense Department Schools Use Caucus to Create Online Virtual Campus

Initial program focuses on faculty development.

Consulting Firm Creates Online Community Using Caucus From Screen Porch

Government agencies and commercial firms collaborate.

Who We Are

- **Caucus Success Stories**

Licensing and Sales

Job Opportunities

Contacting Screen Porch

Latest News

http://screenporch.com/sp/company/stories/pr_TIES.html Go
1 captures
9 Jul 98 - 9 Jul 98
JUN JUL AUG
9
1997 1998 1999
Clos
Hel

Join the Conversation℠: A Caucus Success Story

Who We Are
Caucus
On the Porch
Screen Porch

Who We Are

- **Caucus Success Stories**

Licensing and Sales

Job Opportunities

Contacting Screen Porch

Latest News

For information contact:
Jan Searls (searls@screenporch.com)

Educational Institute Selects Caucus for Worldwide Online Education Program

Innovative Virtual Campus goes live using Caucus discussion spaces

Washington, DC, April 7, 1997—Screen Porch LLC and The Institute for Educational Studies (TIES) announced today that TIES has chosen Caucus Web-based groupware from Screen Porch as the platform for its virtual campus in cyberspace. The first group of teachers from around the world has already begun a new masters degree program in Transforming Education. The online program uses Caucus to deliver distance learning to degree candidates. The TIES project has been guided by a registered Screen Porch Partner.

Using Caucus on the World Wide Web, course participants come together in a variety of lecture halls, seminar rooms, and discussions. The distributed educational network allows for guest faculty to provide services from locations anywhere in the world. Participants in the program hail from all over the United States, South America, and Europe.

Screen Porch president Tom Mandel comments: "We are pleased that Caucus has been chosen for this innovative project. Educational institutions around the world see the importance of distance learning and the online environment. We are happy that TIES, like so many other institutions, will be using Caucus to support this critical application."

Phil Gang of University of Vermont said, "We selected Caucus because of its power and flexibility. The pedagogical clarity of the software makes it a rich platform for thoughtful conversation and learning. Caucus allowed us to create custom environments for people working and learning together—a Virtual Campus specific to the needs of our innovative program. Caucus lets us bring together online learners from around the world and help them gain experience and skills to transform their schools and teaching environments."

Screen Porch was founded in 1996 to develop and market Caucus, the most powerful and open way to create Web site workspaces for teams and learning groups online. Caucus enables people to meet, share, learn, and work together in open and flexible Web conversation spaces. All Caucus tools and technologies are completely Web-compatible. Caucus conversation spaces can include and integrate any-and-all Web applications and Web-compatible information. Caucus is distributed both directly and through partnerships with resellers and consultants. Screen Porch is located at 4020 Williamsburg Court, Fairfax, VA 22031, and may be reached at 703-243-3001 (telephone) or 703-385-3209 (fax), and on the Web at http://screenporch.com.