Web Account Registration

By Tom Fitch, Technical Services Team Member, 4D Inc.

Technical Note 09-40

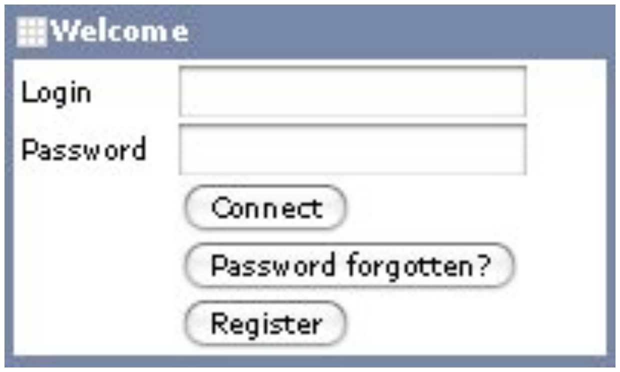
Table of Contents

Abstract

This Technical Note shows how to implement a registration system for Web Accounts. It includes a built component that can automatically handle registration and the source of the component so that it can be customized as necessary. The component includes an Administration GUI and preferences so that email can be sent to new account registrants. All data is stored outside the data file in easily imported XML files.

Introduction

Adding a widget to your website that allows for account registration is a common practice for many websites. A sample widget follows:



The Web Account Component included with this Tech Note offers methods which allow you to create new web accounts, validate existing users, and change user passwords. The component methods include:

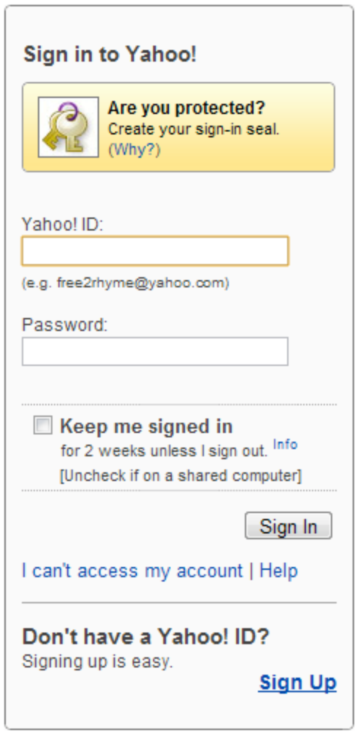
- A method which validates user login and password (login name is based on the email address given).
- A method which handles Web Account Component (WAC) JavaScript functions calling back to 4D for registration of new accounts and changing passwords.
- The necessary setup methods to initialize, define settings, and save settings of the WAC.

Further implementation of the actual login process and the login widget is up to the developer to implement for their own application.

Web Account Registration

The basic functionality of a Web Account Registration system serves many purposes. This tool can be expanded on your specific website to account for security, configurability for specific users, or a contact system for reaching new potential customers. All of this is dependant on what information you require from your users and what level of implementation you wish to use on your site.

One good example of a Web Account Registration system is used by Yahoo! for user accounts. A sample can be seen here:


Sign in to Yahoo!
Are you protected?
Create your sign-in seal.
(Why?)
Yahoo! ID:
(e.g. free2rhyme@yahoo.com)
Password:
Keep me signed in
for 2 weeks unless I sign out. Info
[Uncheck if on a shared computer]
Sign In
I can't access my account | Help
Don't have a Yahoo! ID?
Signing up is easy.
Sign Up

Shown here is the Yahoo! sign in widget, with a "I can't access my account" link as opposed to "Forgot Password". With this widget, Yahoo! tracks their account holders. Yahoo! uses this account registration to limit some functions of their website (fantasy sports and Yahoo! games for example) to registered users. In this way there is a security system involved with the Web Account system. Yahoo! also uses this to save customer data. Each user creates their own profile which can save anything from their mailing address and website preferences to their fantasy sport and gaming histories from the site. In this way, each user's experience is customized and remembered every time they return to the site. Finally, Yahoo! can use the data generated here-in as an email list. Because it is necessary to have an email address to sign up for a Web Account, this same email address can be used to send out sales emails (including offers only available to registered users) or functions specific to different user accounts. For example, regular fantasy sports emails are sent to users who access that aspect of the website.

This example shows a highly sophisticated Web Account Registration system. The component included offers the necessary tools to add web accounts to your website; you can create accounts, reset passwords, and validate users. How to use the accounts is up to the developer.

How to Use the Web Account Component

There are four main sections regarding using the WAC:

- Installing the component and the necessary support files.
- Setting up the WAC.
- Component method documentation.
- Sample WAC web pages.

It is important to make sure you correctly install and setup the WAC before trying to implement Web Accounts in your application.

Component Installation

The files needed to install the WAC are:

- The built component: WebAccount_1.0.4dbase
- The 4D Internet Commands plug-in: 4D InternetCommands.bundle
- The 4D Pack plug-in: 4D Pack.bundle
- The WAC HTML pages: changepwd.html and signup.html

The steps to install the component are as follows:

1. Copy WebAccount_1.0.4dbase into the Components folder next to your database structure. This is inside the .4dbase package on Mac OS.

2. Copy the plug-ins 4D InternetCommands.bundle and 4D Pack.bundle into the Plugins folder next to your database structure. This is inside the .4dbase package on Mac OS.

3. Copy the HTML pages changepwd.html and signup.html into the Root Web Folder of your database.

4. Open the database. Add the following line of code to the On Startup and On Server Startup database methods:

   ```
   WAC_Initialize
   ```

   Restart the database to initialize the WAC.

5. Add the following the following line of code to the On Exit and On Server Shutdown database methods:

   ```
   WAC_Shutdown
   ```

6. Add the following code to the On Web Connection database method. Note, this assumes that the parameters ($1, $2, $3, $4, $5, and $6) have not already been declared and assigned to variables. If that is not the case, use the variables that have already been set up.

```
C_TEXT($1;$2;$3;$4;$5;$6)
C_TEXT($url_t;$header_t;$clientIP_t;$serverIP_t;$user_t;$pass_t)
$url_t:=$1
$header_t:=$2
$clientIP_t:=$3
$serverIP_t:=$4
$user_t:=$5
$pass_t:=$6
WAC_IsOnWebConn ($url_t;"")
```

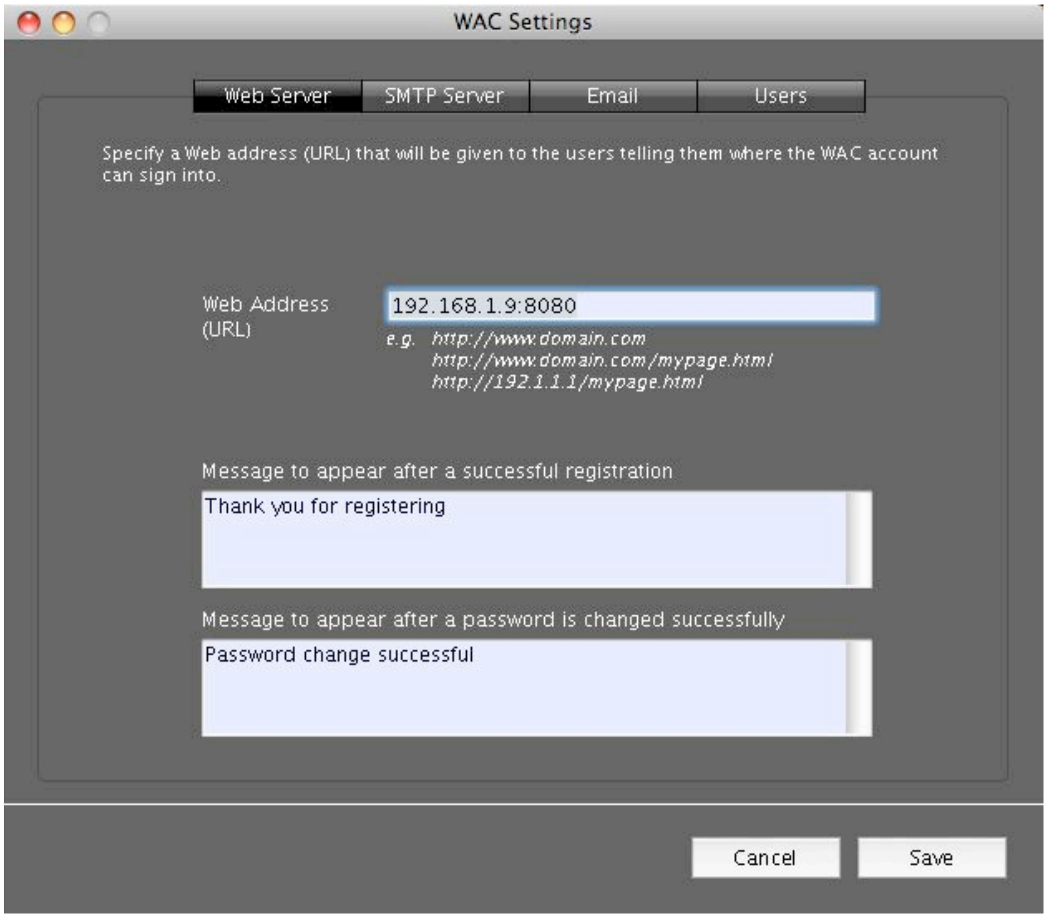
This is the basic component installation. After finishing the setup, the WAC web pages can be linked to for creating new accounts and changing passwords and the WAC_ValidateAccount method will function for validating existing accounts.

Component Setup

The WAC includes a GUI built to facilitate setup and handle administration. To activate this GUI execute the WAC_Settings component method. Each Setup page has a "Save" and "Cancel" button to save your changes or to cancel them for the entire Setup session (not just that page).

Web Server Setup

The Web Server Setup page appears as shown here:



Specify a Web address (URL) that will be given to the users telling them where the WAC account can sign into.

Web Address (URL): 192.168.1.9:8080

e.g. http://www.domain.com
http://www.domain.com/mypage.html
http://192.1.1.1/mypage.html

Message to appear after a successful registration: Thank you for registering

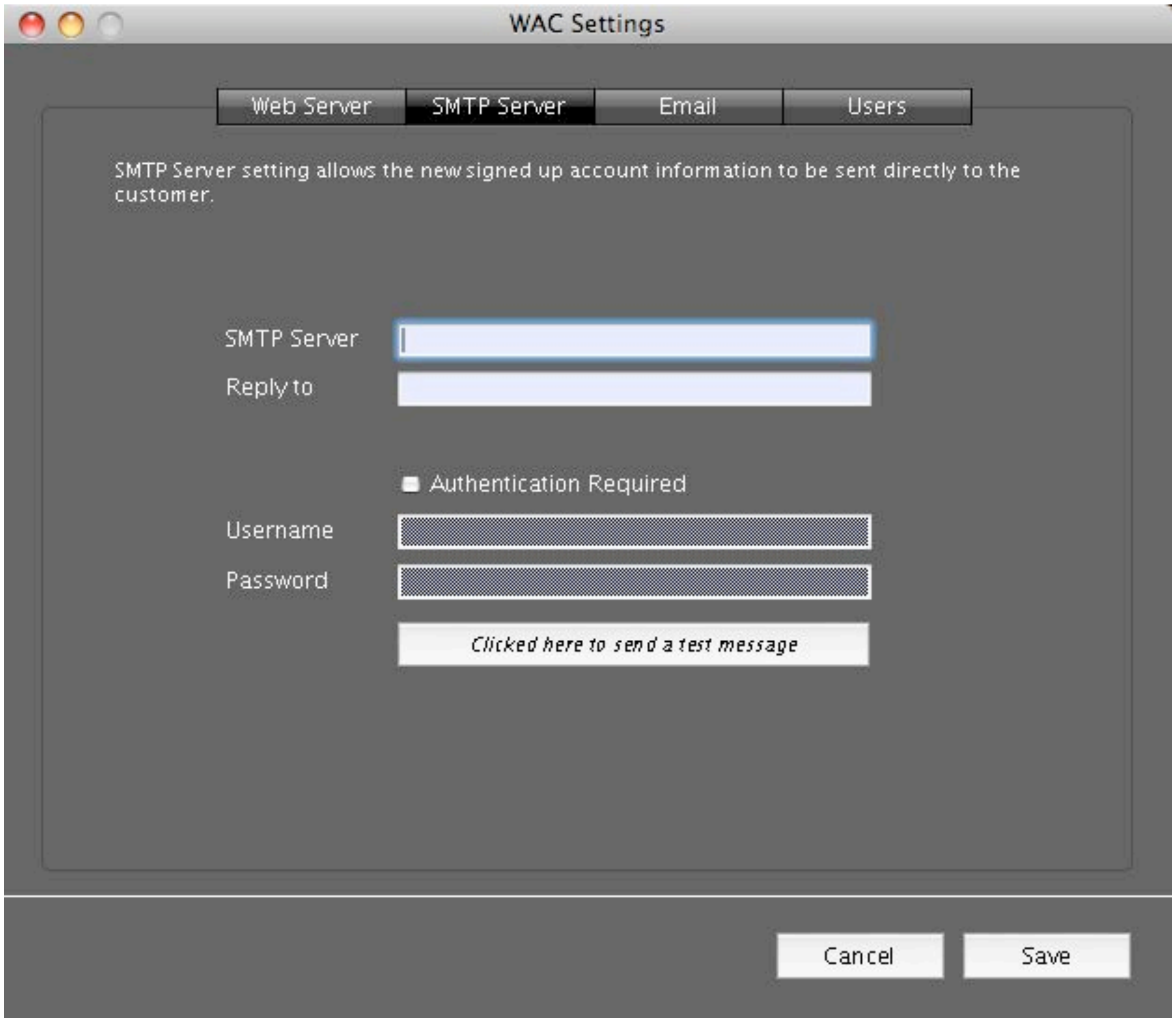
Message to appear after a password is changed successfully: Password change successful

Three fields can be set up on this page:

- Web Address (URL): This is the homepage that the WAC is creating accounts for.
- Message to appear after a successful registration: This is the message that appears after users register only if you do not redirect them to another page via WAC_IsOnWebConn as per the description under the Component Method Documentation section.
- Message to appear after a password is changed successfully: This is the message that appears after users change their password only if you do not redirect them to another page via WAC_IsOnWebConn as per the description under the Component Method Documentation section.

SMTP Server Setup

The SMTP Server Setup page appears as shown here:



SMTP Server setting allows the new signed up account information to be sent directly to the customer.

SMTP Server

Reply to

Authentication Required

Username

Password

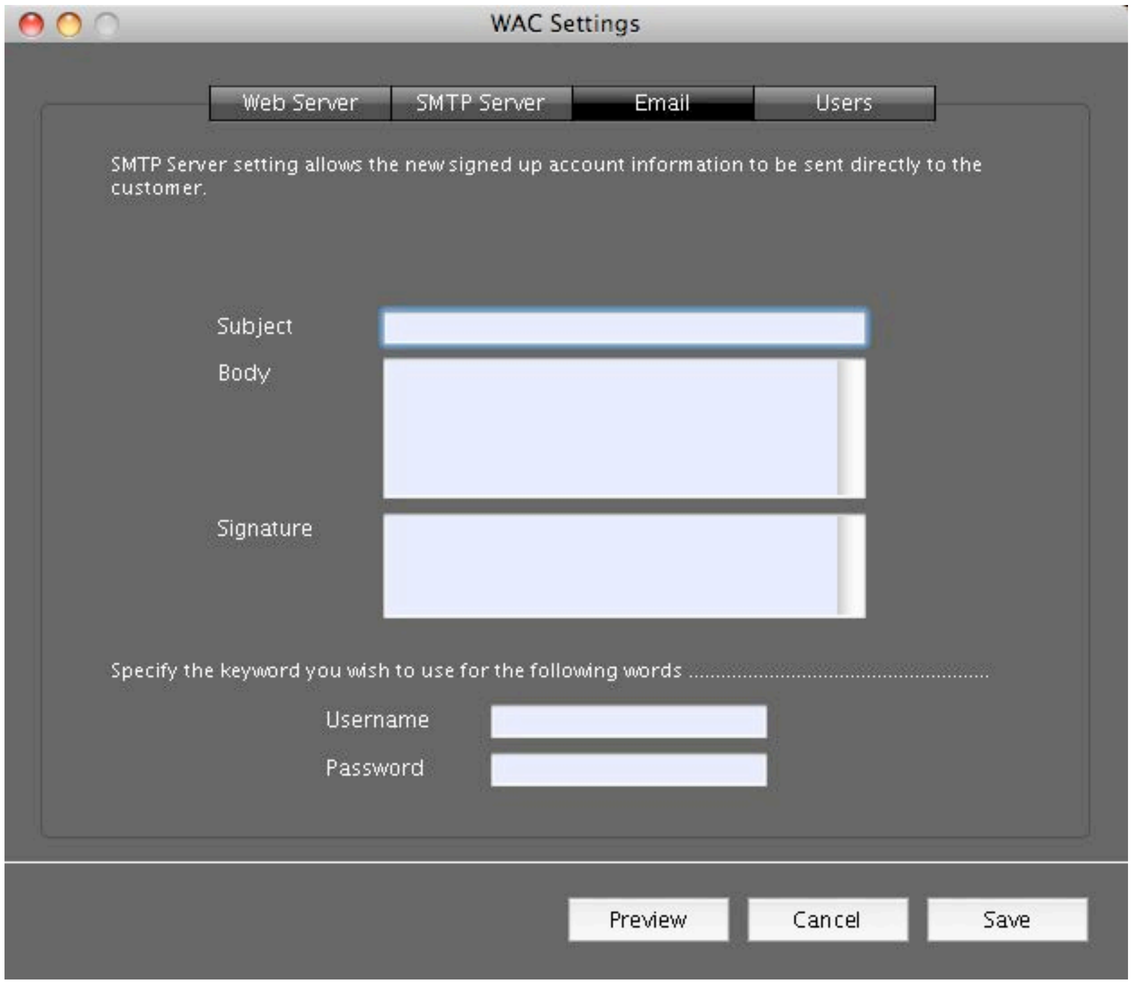
The SMTP Server Setup page has two required fields and two optional fields:

- SMTP Server: This is the SMTP Server that is used to send email messages.
- Reply To: This is the email address that recipients see as the “sender” and who they will reply to.
- Username: If the Authentication Required box is checked this field becomes enterable. It is the username required to log in to the SMTP Server.
- Password: If the Authentication Required box is checked this field becomes enterable. It is the password that corresponds to the above username and is required to log in to the SMTP Server.

After setting up the SMTP Server you can send a test email by clicking the “Clicked here to send a test message” button and filling out the email address, subject, and body fields that are displayed and then clicking “Send”.

Email Settings

The email setup page appears as shown below:



SMTP Server setting allows the new signed up account information to be sent directly to the customer.

Specify the keyword you wish to use for the following words ....

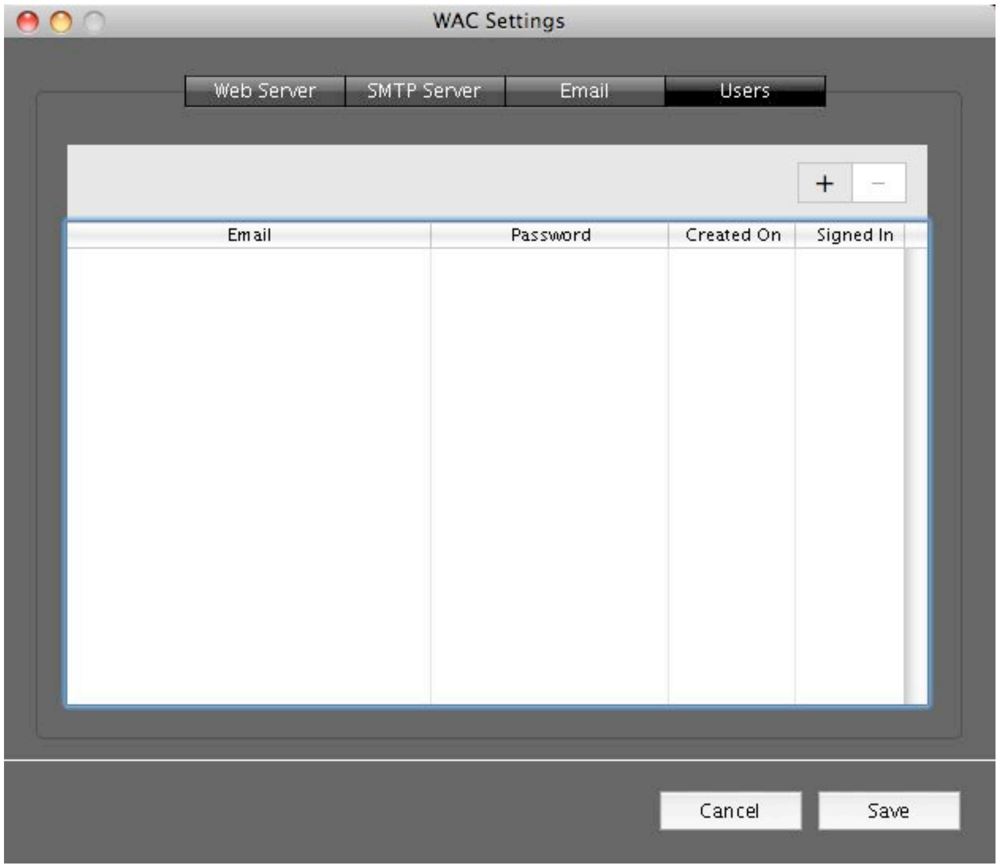
The Email Settings page has five fields that need to be entered.

- Subject: This is the subject of the automatic email that is sent when a user first registers their account.
- Body: This is the main body of the message that is sent when the user first registers their account. There is an additional body portion which includes the username and password they have signed up with and the placeholders described below. Here is a sample:
  UsernamePlaceholder: me@4d.com
  PasswordPlaceholder: 3370Z6d013Y
- Signature: This is the signature that is used in emails sent when users sign up.
- Username: This is the text that takes the place of "UsernamePlaceholder" shown above.
- Password: This is the text that takes the place of "PasswordPlaceholder" shown above.

The Preview button on this page allows you to view a sample of the email that will be sent.

User Administration

The Users administration page appears as shown below:



This page is not necessary for setup. This page is the Administration Page for users. It lists existing users, their passwords, and the dates the account was created and last signed in to. Users can be added or deleted with the "+" and "-" buttons in this page. The fields shown in the output area are enterable to facilitate editing.

Component Method Documentation

There are five methods in the WAC Component that users need to call. Here is a list and short description of each method's purpose:

- WAC_Initialize: This method initializes the Web Account Component's XML files, loading the necessary data.
- WAC_IsOnWebConn: This method tests the requested URL to see if it is part of the WAC Component. If so, it redirects it to the specified redirect URL.
- WAC_Settings: This method displays the settings process.
- WAC_Shutdown: This method saves existing settings and information to disk.
- WAC_ValidateAccount: This method validates the account given the username and password passed.

A more in depth description of each method and its usage follows.

WAC_Initialize

Usage: This method needs to be executed prior to any other subsequent execution of WAC methods. It is recommended that this method is executed in the On Startup database method.

Parameters: None

This method sets up the necessary interprocess variables for WAC and loads settings for accounts, email server, and error codes from disk. It also begins a periodic update process to save those settings to disk.

WAC_IsOnWebConn

Usage: This method determines whether the requested URL belongs to WAC. If so, the web request will be performed. Afterwards, WAC redirects the response to the given URL (provided in $2 as "http://..."). If the value in $2 does not start with "http://", WAC assumes that the given path is relative to the Web Root of the host database. It then sends a redirect to that path.

Parameters:
$1: Text: Requested URL
$2: Text: Redirect URL or Path (If an empty string, a thank you message set in server setting is the returned HTML Text)

Result:
$0: Boolean: True if the URL belongs to WAC, False otherwise

Syntax:

```
WAC_IsOnWebConn ($url_t;"http://www.domain.com") -> Boolean
WAC_IsOnWebConn ($url_t;"/page.html") -> Boolean
WAC_IsOnWebConn ($url_t;"page.html") -> Boolean
WAC_IsOnWebConn ($url_t;"") -> Boolean
```

This method is used to handle the HTTP requests created by the sample HTML pages. The $url_t variable is the incoming HTTP request (usually $1 from the On Web Connection database method). If it is "/wac/signup@" then the signup function is run and input values are retrieved from the form. If $url_t is "/wac/changepwd@" then the change password function is run and the input values are retrieved from the form. In either case, the user is redirected afterwards as follows:

Parameter 2	**Redirect Site**
"http://www.domain.com"	http://www.domain.com
"/page.html"	htmlrootfolder/page.html
"page.html"	htmlrootfolder/page.html
""	Pre-set value from WAC_Settings GUI

WAC_Settings

Usage: This method opens the WAC Settings GUI as described in the Component Setup section above.

Parameters: None

WAC_Shutdown

Usage: This method saves all necessary information into the disk before shutting down the system. It is recommended that this method is executed in the On Exit database method.

Parameters: None

This method saves the WAC interprocess variables setting accounts, email server, and error codes to disk. It is important not to run the WAC_Shutdown method without having first run WAC_Initialize, otherwise errors will occur.

WAC_ValidateAccount

Usage: Validate the given username and password.

Parameters:
$1: Text: Username
$2: Text: Password

Result:
$0: Text: Unique User ID

This method should be called when trying to validate a web user. An example would be calling it from the On Web Connection database method when the user clicks the "Login" button on your website. (Another way this can be achieved is via an HTML button object and a JavaScript function calling the WAC_Validate method via 4D HTML tags.)

Once it is called, pass the username and password to the function. If the user is validated their unique ID will be returned. If not, a blank string is returned.

WAC Web Pages

There are two sample web pages included with the WAC: changepwd.html and signup.html.

Signup.html

The signup.html page includes two input HTML objects which are used to input the email address to sign up with and the button to sign up.

Signup email address:

```
<input type="text" name="signupemail" value="" />
```

Register Button:

```
<input type="button" onclick="WacSignup();" value="Register Now" />
```

The register button calls the WacSignup JavaScript function when it is clicked. Here is the code for that function:

```
function WacSignup(){
   var email = document.forms.signupform.signupemail.value;
   if(email != '')
          makeCall("/wac/signup?email=" + email, mysignuphandler);
   else
   alert('Please specify your email address');
}
```

This function checks the signupemail input area's value and if it is not blank calls the makeCall JavaScript function with the "/wac/signup…" parameter, where … is the email address being signed up. The makeCall function creates an HTTP Request and tests a handler function to make the HTTP request to the database. This prompts the call to WAC_IsOnWebConn in the database On Web Connection database method. Since the call starts with “wac/signup@” it is treated as a WAC request and the user’s email is signed up.

Changepwd.html

The changepwd.html page uses three input areas for text and one for a button. The three text input areas are the email address an account is listed under, the old password, and the new desired password.

Signup email address:

```
<input type='text' name='signupemail' value='' />
```

Old password value:

```
<input type='password' name='oldpassword' value='' />
```

New password value:

```
<input type='password' name='newpassword' value='' />
```

The button is used for changing the password. Here is the Change Now button:

```
<input type="button" onclick="WacChangePassword();" value="Change Now" />
```

This button calls the WacChangePassword JavaScript function. That function is shown here:

```
function WacChangePassword(){
   var email = document.forms.signupform.signupemail.value;
   var oldpwd = document.forms.signupform.oldpassword.value;
   var newpwd = document.forms.signupform.newpassword.value;
   if(email == '')
          alert('Please specify your email address');
   else if (oldpwd == '')
          alert('Please specify your old password');
   else if (newpwd == '')
          alert('Please specify your new password');
   else
makeCall("/wac/changepwd?email="+email+"&old="+oldpwd+"&new="+newpwd,
mychangepwdhandler);
}
```

Note: The last line of this JavaScript function overflows from the first line onto the second.

This page tests all of the necessary input values to ensure they have been entered, then it calls the same makeCall function used in the signup.html page. In this case the call includes the email address, the old password, and the new password. The makeCall function creates an HTTP Request and tests a handler function to make the HTTP request to the database. This prompts the call to WAC_IsOnWebConn in the database On Web Connection database method. Since the call starts with "wac/changepwd@" it is treated as a WAC request and the user's email is checked against their current password. If this is correct the new password is used to update the password value.

How the Component Works

The component stores all of its data in an XML file outside the data file in the WAC folder next to the component structure. The settings and users are stored here. When the component initializes, interprocess variables are initialized with the given data. The WAC_Initialize method handles this as shown here:

```
C_BOOLEAN(<>wac_quit_b;<>wac_needupdate_account_b;<>wac_needupdate_settings_b)
<>wac_quit_b:=False
<>wac_needupdate_account_b:=False
<>wac_needupdate_settings_b:=False
WAC_LoadAccounts_FromDisk
WAC_LoadServerSettings_FromDisk
WAC_LoadErrorCodes_FromDisk

C_LONGINT($process_l)
$process_l:=New Process("WAC_PeriodicUpdate";1024*1024;"WAC Peridic Update";*)
```

The component methods WAC_LoadAccounts_FromDisk, WAC_LoadServerSettings_FromDisk, and WAC_LoadErrorCodes_FromDisk load the given data from disk. Each of these methods initializes a set of interprocess arrays which are then used to store that data.

At the end of the WAC_Initialize method, a new process is created by calling the WAC_PeriodicUpdate command. This method sets a timer and periodically repeats writing the data from the interprocess arrays to disk.

The WAC_Shutdown method shown here also writes the data to disk:

```
<>wac_quit_b:=True

WAC_SaveAccounts_ToDisk
WAC_SaveServerSettings_ToDisk
WAC_SaveErrors_ToDisk
```

The component methods WAC_SaveAccounts_ToDisk, WAC_SaveServerSettings_ToDisk, WAC_SaveErrors_ToDisk save the data from the appropriate interprocess arrays to disk.

The WAC_Settings method simply runs the project form "Pane" which is used to save the necessary settings for the Web Server and Email Server. Once the setup has been handled, the web accounts can be created. This is generally handled on the front end (the website) via calls to the WAC_IsOnWebConn method. That method includes a case statement as shown here which handles both account creation and changing passwords:

```
Case of
   : ($url_t="/wac/signup@")
         ` code here handles signup
   : ($url_t="/wac/changepwd@")
         ` code here handles changing passwords
End case
```

Note that the case statement depends on the $url_t parameter which is the HTTP request from the On Web Connection database method. More can be read about that under the WAC Web Pages section above.

The code that handles the new accounts is shown here:

```
$email_t:=WAC_GetWebVar ("email")
If ($email_t#"")
   returnHtmlMessage_t:=<>wac_signup_thankmsg_t
   WAC_AddAccount ($email_t)
   WAC_SendHTTPRedirect ($redirectpage_t)
End if

$0:=True
```

First the WAC_GetWebVar method is called to get the email address. This gets a variable that was set by the GET WEB FORM VARIABLES command, retrieving data from the front end. Once the email has been returned, WAC_AddAccount is used to add the account; saving the data to the interprocess variables which have already been set up. WAC_SendHTTPRedirect redirects the front end to the previously specified webpage.

The code that handles changing the user password is shown here:

```
$email_t:=WAC_GetWebVar ("email")
$oldpwd_t:=WAC_GetWebVar ("old")
$newpwd_t:=WAC_GetWebVar ("new")

$error_l:=WAC_ChangePassword ($email_t;$oldpwd_t;$newpwd_t)
If ($error_l=0)
   returnHtmlMessage_t:=<>wac_pwd_changemsg_t
Else
   returnHtmlMessage_t:=WAC_GetErrorMessage ($error_l)
End if

WAC_SendHTTPRedirect ($redirectpage_t)

$0:=True
```

First the WAC_GetWebVar method is called to get the email address, old password, and new password. This gets a variable that was set by the GET WEB FORM VARIABLES command, retrieving data from the front end. Once the data is returned the WAC_ChangePassword command is called with those values as parameters. This command tests the old password value against the user’s existing password and - if it fails - returns an error code. In this case, WAC_GetErrorMessage is called to return the given error to the user. Otherwise the new password value is set as the password and saved in the appropriate interprocess array. The password change message is given and WAC_SendHTTPRedirect redirects the frontend to the previously specified webpage.

Conclusion

This Technical Note includes a component with the necessary tools to implement a Web Account Registration system. It allows developers to implement login, creation of new users, and changing passwords. It also includes an administration client inside 4D for the Web Accounts. Using the source code of the component and the data structures created, extra data can be stored for user profiles and further possibilities. Using the sample HTML pages and the JavaScript functions defined herein can further the interactive frontend functionality as well.