

Enterprise Web Developer + iWD

Installation Guide

Build 809 (Open Source Version)

Background

Welcome to the Free Open Source version of Enterprise Web Developer (EWD). This document describes the installation of EWD and its web-based management Portal (*ewdMgr*) application.

Build 809 includes the iWD extensions for iPhone Web Application development.

Pre-requisites

In order to use the Open Source version of EWD, you need a Linux system with the Open Source GT.M database installed, configured and working.

For details on GT.M see <http://www.fis-gtm.com>

You'll also need the Apache Web Server to be installed, configured and working and you'll need to have installed and configured the *m_apache* gateway. See the *m_apache* documentation which is included in the source file *m_apache.c*. At the end of this document you'll find a quick summary of how to configure *m_apache* for use with EWD. Note that for best results, *m_apache* should be configured to use the *xinetd* internet superserver.

This document will assume that you've created a GT.M database in the path */usr/local/gtm/ewd* but you can adapt the instructions according to your requirements.

File Contents

The zip file you downloaded contains files in four directory paths:

- routines: Contains the source GT.M Mumps routines for EWD
- ewdMgr: Contains the source EWD pages for the EWD management portal
- m_apache: Contains the source code and pre-built executables for *m_apache*
- iwd: Contains the Javascript and CSS resources for iWD

Installing the EWD Routines

Simply copy the routine files to */usr/local/gtm/ewd* or the directory path you've configured for running EWD routines. They will be automatically compiled and linked on demand by GT.M. Optionally you may decide to force their compilation using the command:

```
mumps *.m
```

Installing the iWD Resource Files

Copy the entire */iwd* directory path contents from the zip file directly under your machine's web server directory. For example, on a Ubuntu Linux machine, you should end up with the path:

```
/var/www/iwd
```

You now have all the Javascript and CSS files needed by iWD correctly installed.

Installing the ewdMgr Application

You'll first need to create your EWD Application Root Path. This is a directory path under which all your EWD source application files will reside. You can use any directory path you want, but these instructions will assume you'll use */usr/ewdapps*. Create this directory path using *mkdir* or a utility such as WinSCP.

Next, create a subdirectory under the Application Root Path named *ewdMgr*, eg */usr/ewdapps/ewdMgr*.

Copy all the files in the *ewdMgr* zip file directory to the */usr/ewdapps/ewdMgr* directory.

You will need to ensure that appropriate permissions are set for the Application Root Path, the *ewdMgr* directory and the files it contains so that the files can be read by the EWD GT.M routines.

In the zip file, under the */ewdMgr* directory path, you'll find a sub-directory named */resourceFiles*. This contains a variety of Javascript, CSS and image files that are used by the *ewdMgr* application. The files in this subdirectory should be copied to a subdirectory named */resources* that you should create under your web server's root directory. For example, on a Ubuntu Linux system you would copy these files to the directory path */var/www/resources/*.

Configuring EWD

EWD needs to be configured so it can find your EWD source files and so that it knows where to create the routines and other files during its compilation process. To

simplify this process, the first time you attempt to compile an application, EWD will ask for and save this information.

Go into the GT.M shell, making sure you are in the directory where you saved the EWD Mumps routines, eg:

```
ewd@mdb:~$ cd /usr/local/gtm/ewd
ewd@mdb:/usr/local/gtm/ewd$ mumps -direct
```

```
GTM>
```

Now start the compiler, specifying the *ewdMgr* application:

```
GTM> d compileAll^%zewdAPI("ewdMgr")
```

[Note that unlike the Caché version of EWD, you don't need to specify the technology to which you want to compile. The Open Source version only generates GT.M / Mumps routines.]

Because this is the first time you've used the compiler, EWD doesn't know where to find the *ewdMgr* source files and doesn't know where to create its output files, so it will ask you to specify this information. It will suggest values for each of the parameters which you can over-ride if you wish. In the example below the default values were accepted simply by pressing Enter at each question:

```
GTM>d compileAll^%zewdAPI("ewdMgr")
```

Installing/Configuring Enterprise Web Developer version 4.0.770

Note: hit Esc to go back at any point

Application Root Path (/usr/ewdapps): /usr/ewdapps

Routine Path (/usr/local/gtm/ewd/): /usr/local/gtm/ewd/

Javascript and CSS File Output Path (/var/www/resources/): /var/www/resources/

Javascript and CSS File URL Path (/resources/): /resources/

Enterprise Web Developer version 4.0.770 is configured and ready for use

The parameters you are asked to provide are as follows:

Parameter	Description
Application Root Path	The directory path under which you create all your EWD application source code. Each application is represented by a subdirectory under the Application Root Path and the page source files for each application reside in their respective application subdirectory. Default: /usr/ewdapps/

Routine Path	<p>The directory path into which EWD will create run-time GT.M Mumps routines. Each EWD page or fragment file will be converted into a corresponding Mumps routine file.</p> <p>Default: /usr/local/gtm/ewd/</p>
Javascript and CSS File Output Path	<p>In addition to the Mumps routines, EWD's compiler generates a number of additional Javascript and CSS files. These need to reside in a directory that can be accessed by the Apache web server.</p> <p>If you are running Apache on the Linux server that you use for EWD compilations, then you should specify a directory that is explicitly or implicitly mapped for use by Apache.</p> <p>If you are compiling EWD applications using a different Linux server, then set this path to any convenient directory, but make sure that you copy all the files generated by EWD from this directory to the server on which you run Apache.</p> <p>Default: /var/www/resources/</p>
Javascript and CSS File URL Path	<p>EWD's compiler has to generate the URLs for the Javascript and CSS files it creates. Therefore it needs to know the relative path that you want to use so that Apache can find and fetch the files for the user.</p> <p>Default: /resources/</p>

Note: make sure that the directories into which EWD will write files have been created and that Read/Write permissions have been set correctly.

Having set these parameters, EWD is able to compile the ewdMgr application and you should see it generating the list of pages it processes.

If you've correctly installed and configured the m_apache gateway, you'll now be able to run the ewdMgr application. Start up a browser and point it at the following URL (change the IP address / domain name as appropriate):

<http://192.168.1.111/ewd/ewdMgr/index.mgwsj>

If you've correctly configured everything, you should now be running the EWD management portal application.

Congratulations! EWD is now installed, configured and ready for use.

Invoking EWD pages in a browser

The URL convention for invoking an EWD/m_apache page is as follows:

[http://\[domain name\]/ewd/\[application name\]/\[page name\].ewd](http://[domain name]/ewd/[application name]/[page name].ewd)

If the EWD page is defined as a first page, then it can be invoked via a URL without any associated token name/value pair, and its invocation will initiate a new EWD session. All other EWD pages or fragments can only be accessed via a tokenised URL. This is to prevent unauthorised use of these pages and/or access to the pages.

Notes on using the YUI Custom Tags

Before using the YUI Custom Tags you must first install the YUI Javascript library. You can get this from Yahoo! **Make sure you get version 2.6.0.** This can be downloaded from <http://yuilib.com/downloads/>. By default, EWD expects this to be placed under the `/var/www` path.

If you use the YUI custom tags in an EWD application, you need to be aware that during the first compilation of the application EWD creates copies of certain YUI library files so that they are in the correct path for its “just in time” loader. In order for this to take place, you will need to (at least temporarily during the first compilation) change the permission of the following paths to 777 (ie full read/write/delete access):

`/www/var *`

`/www/var/ewd` (If this path does not exist, you should create it) +

** In fact this path is only required to have 777 permissions once: the very first time you use a YUI tag.*

+ This path is only required to have 777 permissions once for each new application that contains an EWD tag: for the very first compilation of that application.

Notes on Installing and Configuring m_apache for use with EWD

In the EWD download zip file you'll find a directory path named `m_apache` which contains all the Free Open Source `m_apache` components. Full documentation is included in the `m_apache.c` source file. Follow the installation instructions, ensuring that you configure `m_apache` to use the `xinetd` internet superserver. Remember to copy the back-end GT.M Mumps routine files (`_ZMGWSI.m` and `_ZMGWSIS.m`) to an appropriate GT.M routine directory.

The Root Path (`/ewd/`) should be configured in the Apache configuration file to connect to the `m_apache` gateway and to run the EWD page dispatch routine. This is separately described in the `m_apache` documentation, but a suggested Apache configuration is as follows:

```

LoadModule m_apache_module /usr/mgwsibin/m_apache22.so

<Location /ewd>
    SetEnv MGWSI_PORT 7041
    SetEnv MGWSI_M_UCI /usr/local/gtm/ewd/mumps.gld
    SetEnv MGWSI_M_FUNCTION runPage^%zewdGTMRuntime
</Location>

```

This configuration loads the *m_apache* module into Apache and associates the URL path */ewd* with *m_apache* so that it automatically invokes the EWD page dispatch routine (*runPage^%zewdGTMRuntime*) whenever a URL starting with the path */ewd* is received by Apache.

The GTM database path to which *m_apache* will connect is determined by the *MGWSI_M_UCI* parameter, so in the example above, any HTTP requests with a URL starting */ewd* will be directed to the GT.M instance running in */usr/local/gtm/ewd/*.

For the configuration above, your */usr/local/gtm/zmgwsi* script file should look like the following:

```

#!/bin/bash
cd /usr/local/gtm/ewd
source /usr/local/gtm/gtmprofile
mumps -r INETD^%ZMGWSIS

```

and your */etc/xinetd.d/mgwsibin* file should look like:

```

service mgwsibin
{
    disable          = no
    type             = UNLISTED
    port            = 7041
    socket_type     = stream
    wait           = no
    user           = root
    server         = /usr/local/gtm/zmgwsi
}

```

It is also recommended that you define the security settings for the reserved Apache location */mgwsibin/sys*, eg:

```

<Location "/mgwsibin/sys/">
    # This is a reserved location used for the m_apache systems facilities
    invoked through: /mgwsibin/sys/system_functions.mgwsibin
    Order deny,allow
    Allow from all
</Location>

```

Configuring the Encryption Libraries for *m_apache*

Some of EWD's functions make use of OpenSSL encryption and encoding libraries that are interfaced through the *^%ZMGWSIS* routine which calls into *m_apache* for their

invocation. You must make sure that the core OpenSSL libraries are correctly installed in your Linux system. If you've enabled SSL for Apache, you probably already have these files on your system.

The required files (actually symbolic links) are:

- *libcrypto.so*
- *libssl.so*

These are usually located in the */usr/lib/* path. On some systems, eg Ubuntu Linux, you may find the master files after you've installed Apache's SSL extensions, but the symbolic links to them may be missing. If this is the case, do the following:

Look for the files:

- */usr/lib/libcrypto.so.0.9.8*
- */usr/lib/libssl.so.0.9.8*

(This version number at the end of the file name may be different on your system).

Create the symbolic links as follows (you'll need to have root privileges):

```
cd /usr/lib
ln -s /usr/lib/libcrypto.so.0.9.8 libcrypto.so
ln -s /usr/lib/libssl.so.0.9.8 libssl.so
```

Restart Apache and the encryption and encoding functions should be ready to run.