WxCoder II

A Web-based data entry system for observers

Program & Installation Guide

Version 1.0

September 30, 2003

Table of Contents

Introduction

Cooperative observer data has traditionally been collected and transmitted to the National Weather Service by telephone. While the method has been effective, it not only requires significant time but also fails to take advantage of more sophisticated methods of data entry and transmission generally available through use of the Internet.

In 1999 WFO Chicago took a significant step in opening the use of the Internet to cooperative observers across the central United States. The result of the effort was the introduction of an Internet-enabled data entry system called WxCoder (***W***eb ***X***mitted ***C***ooperative ***O***bserver ***D***ata ***E***ncoded ***R***eport). Though an advancement, WxCoder only made the first step. Entry forms were cumbersome, error feedback was confusing, and data retrieval was non-existent..

WxCoder II is a major and substantial enhancement to previous methods of data entry for observers. While addressing the shortcomings of its predecessors, WxCoder II breaks new ground with the introduction of a multi-function database. By combining the power of relational database technology with the Internet, WxCoder II provides information services that neither can deliver alone. Major improvements are provided for both the observer and the NWS; significant new features for both are added.

For the coop observer, WxCoder II provides substantial improvements in data entry. The entry form is clean and concise. Through the use of the WxCoder database and dynamic content technology, data entry now occurs through a one-page, customized form – no need for the observer to muddle through pages of non-pertinent entry blanks. Persistent, site-specific data is automatically presented on the form; no need for repetitious, daily entry.

For the NWS, the introduction of the WxCoder database is a major enhancement to coop observer data collection, processing, analysis, and presentation. The collected data no longer serves one purpose. It is available for multiple uses – automatically generating summaries and reports, direct forwarding to NCDC.

WxCoder II was developed and coded by Bob Somrek, WFO Chicago. Please send comments, suggestions, and bug reports to

Robert Somrek
National Weather Service
333 W. University Drive
Romeoville, IL 60446

815.834.0600 x364
robert.somrek@noaa.gov

Application Architecture

WxCoder is a Web-based data entry system for National Weather Service observers. Its architecture and implementation are driven by two design considerations: a friendly, multi-featured interface and the need to maintain 'state' through a user's session.

The requirement for a friendly user interface is easily met by implementing the interface using HTML/ DHTML forms, CSS style sheets and JavaScript. The need for features such as customized forms and historic data retrieval requires the use of a database to store information relevant to each user.

A more difficult feature to incorporate is the need to maintain 'state.' It is not uncommon for an application such as WxCoder to span several pages that are logically grouped and considered part of a larger unit, defined as a *session*. Unfortunately, the HTTP protocol used in Web-based applications is not well suited for session management; it's a stateless protocol that makes no provision for allowing a Web server to associate one request with any other request. WxCoder requires maintaining state to help identify the user as he proceeds from page to page and to 'remember' data entered or actions requested. Of the methods available for maintaining state, a backend database is the best approach for WxCoder. In this scheme, the client announces itself to the server on each request using an identifier (session ID). The server associates the identifier with the appropriate state information that it maintains for each client in the database.

The architecture solution for the WxCoder application is built using a three tier approach: a client browser at the front, a middle tier web server(s), and a backend database server (Figure 1).

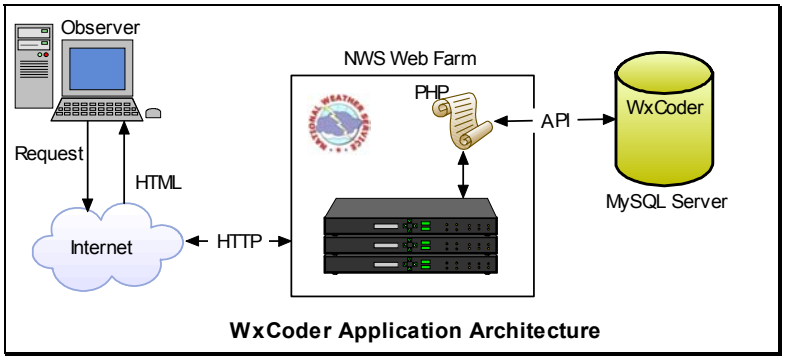

Figure 1

The front end comprises the human interface to WxCoder – normal Web pages. It is implemented through HTML/DHTML forms using CSS style sheets and JavaScript support.

The middle tier consists of Apache running on a web server or in a web farm. A PHP module running as a plug-in performs the processing done in the middle tier and includes support for interacting with the MySQL database backend (additional plug-ins provide support for generating PDF files and graphics). PHP scripting provides the flexibility needed for page customization; pages are generated as requested and to specifications required for the specific client being serviced. PHP scripts process form input, generate SQL queries, process query results, and generate HTML documents for return to the client via the Internet.

The backend tier consists of the database server, not necessarily running on the same server as Apache or even in the same web farm. MySQL is used for this application providing a high performance, flexible database.

All software used to generate and support the WxCoder application is open source with one exception -- a commercially licensed PDF-generating software package.

Application Flow

Though the user's initial encounter with WxCoder is generally considered to be the appearance of the login page, WxCoder begins its internal housekeeping before it generates the page. Upon receiving a request for the login page, WxCoder generates initial state information consisting of a session ID (a 32-character string of alphanumeric characters), a session expiration time (5 minutes into the future), and an initial set of session data (the preferred language, determined by the URL used to get to the login page) and stores it in its MySQL database in table *wxsamSessions*. The HTML page is then generated and returned along with the session ID as an in-memory cookie.

Upon presentation of the login page (**Error! Reference source not found.**, Appendix C), the user is asked to enter his username and user identifier. The page is returned to the web server with the data attached as an HTTP POST request. The receiving script reads the username and user identifier and constructs a SQL query used to determine if a user exists that corresponds to the information passed. If no user exists who matches the unique combination of username and user identifier, an error message is returned along with the login page allowing another attempt at successful login. If the username exists but the user identifier does not match the one stored in the database for the user, the login page is returned with an error messages noting the problem; two more attempts at successful login are allowed before the user account corresponding to username is locked and unavailable.

WxCoder grants access to its pages based on a system of user rights – rights that are dependent on the class of user and are assigned when a new user account is established. WxCoder currently defines nine classes of users: guest, cooperative observer, marine observer, WFO-lite user, WFO user, NWS regional user, NWS national user, support/maintenance user, and superuser. Based on the username and user identifier, WxCoder 'knows' which class of user has logged in and generates an appropriate page to return to the user. If a user attempts to access a page to which he does not have rights, he will be denied access and a *no-rights* page will be returned.

The nine classes of users noted above can be divided into two general classes – observers and administrative users. These two classes define the two broad interfaces presented by WxCoder after login – the user interface and the administrative interface. Further determination on the pages presented is made based on the actual class of user.

The discussion below describes the application flow (Figure 2) for an observer user. The same principles outlined below apply to administrative users though the actual pages displayed and functions performed are different.

Application Flow: User Interface

Having successfully logged in, an observer class user is presented with a data entry page (**Error! Reference source not found.**, Appendix C) – the default page presented on login for an observer. Because WxCoder knows both who the observer is and what site is associated with him through entries in the database, it customizes the page presented through those and other database entries. Customizations include

pre-filled entries for site name and ID, date and time of observation, type of observation. The form is generated to include only those data elements (with information retrieved from the 'forms' series of tables in the database*)* observed by the user.

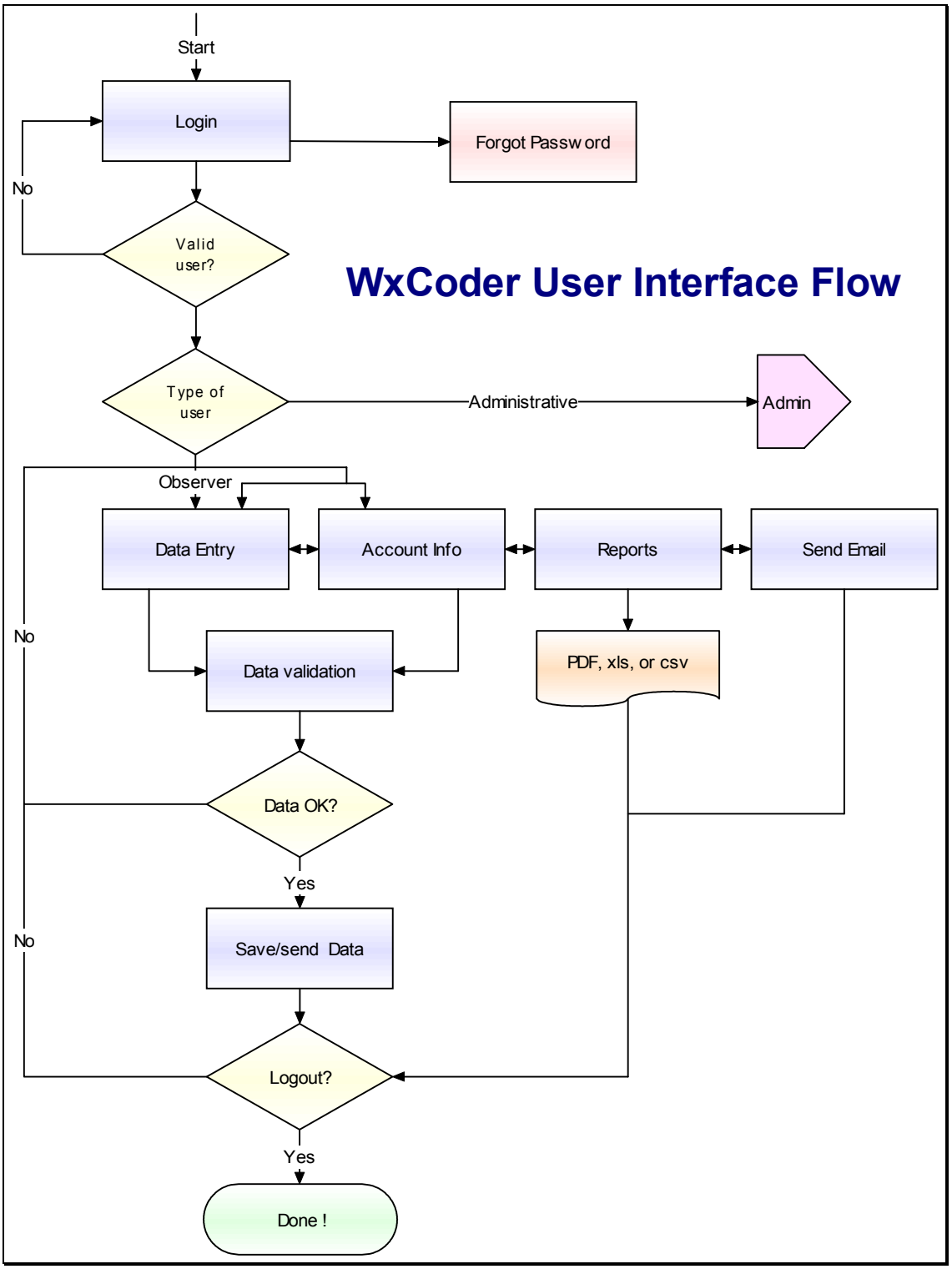

Figure 2

Further customizations include information/links on the page related to the observer's local NWS office/contact. This is accomplished in the PHP files *dataEntryCoop.php* and *dataEntryMarine.php* as appropriate.

With many portions of WxCoder's pages being common (page header/footer, menus, general page layout.), it would be burdensome and time-consuming to maintain these features on every page. Instead WxCoder pages are generated using templates. The use of templates separates the commonly used portions of a page from the rest of the page code. For those portions of a page that only need minor changes, templates also support the use of variables. At runtime, the page's template is retrieved, template variables are replaced with either plain text (HTML) or the output of other templates, and the page is generated. The PHP package **FastTemplate** is used to implement this feature.

When the user submits the data entry page, data validation begins (file *validateDataCoop.php*). The data is first read from the HTTP POST variables returned to the server and stored in an arrayed session variable. Data quality checks (Table 1) are then performed on the data.

- blank form
- ob entry no earlier than 30 minutes before nominal observation time *(regular observation only)*
- invalid characters
- correct number of digits in each field
- data within valid range (WFO settable)
- precipitation entered, if required
- cross checks between data
  - must report precipitation with snowfall *(reg ob only)*
  - must report snow depth with snowfall *(reg ob only)*
  - must report snow depth with snow core *(reg ob only)*
  - don't report snow core with snow depth less than 2 in
  - must report snow core if is on/after last Monday in January
  - must report precipitation type, if reporting, with precipitation
  - must report frozen precipitation type, if reporting precipitation type, with snowfall
  - min temperature must be ≤ max temperature (air and soil temperatures)
  - temperature at observation must be ≤ max temperature
  - max temperature must be ≥ yesterday's temperature at observation and min temperature must be ≤ yesterday's temperature at observation

Table 1

If no errors are encountered, the data are encoded in SHEF and stored in the WxCoder database. A 'success' message (**Error! Reference source not found.**, Appendix C) is returned to the user along with a summary of the observation just entered. At this point the users may logout or choose another Wxcoder function (including entering data again). If an error(s) is encountered, the data entry page is returned to the user with his data still in the entry boxes and a message(s) noting the error encountered (**Error! Reference source not found.**, Appendix C). The errors must be corrected before WxCoder will accept the observation.

WxCoder to AWIPS Data Transmission

WxCoder accommodates two methods of transmitting its collected data to AWIPS – through LDAD and by direct ftp. Transmitting through LDAD is the method currently in use.

Transmission through LDAD

Wxcoder transmits it data to AWIPS by appending its output to files in a pre-defined directory (defined in *_congif.php*) on the web server. The specific file appended to mimics the AFOS PIL specified at the time the site as established (e.g., data to be stored using AFOS PIL *CHIRR3CHI* is appended to file *COOPCHIRR3CHI* on the web server). At this point WxCoder no longer controls the flow of data.

A shell script, running at regular intervals as a cron job on the web server, reads the files in the pre-defined directory. If a file contains data (i.e., is not zero length), it is sent by ftp to one of four Central Region LDAD sites: Central Region Headquarters, WFO Bismarck, WFO Sioux Falls, or WFO Milwaukee. (Files that are successfully read are 'zeroed out'.) Upon being received by LDAD, entry into AWIPS is controlled by LDAD's AFOS2AWIPS.txt (A2A) file. This file contains a listing of acceptable AFOS PILs (9-character product identifiers) and WMO headers that AWIPS recognizes as valid products. ***It is critical that the receiving LDAD site's A2A file be complete and correct to insure transmission through this link***.

When successfully transmitted, a site's observation will appear in AWIPS with the appropriate AFOS identifier.

This section of the manual provides information necessary for successful installation and update of WxCoder on the servers in a web farm.

Web Farm Requirements

WxCoder II is designed to run in a Linux-based web farm. Though hardware is generally not an extra consideration in the installation or operation of its software, WxCoder requires the following minimum software configuration to operate.

- The web servers in the farm must be running Apache v1.3 and PHP v4.1.2
- Two database servers are needed (main and redundant) each running MySQL v3.23.56
- A licensed version of the software package PDFLib v5.0 (http://www.pdflib.com/) must be active on one of the servers in the farm to support downloadable Acrobat forms generation

Later versions of the software noted above (particularly PHP and MySQL) will be supported in WxCoder code as they become available and are installed.

Initial Software Installation

The following steps are necessary for a successful initial installation of WxCoder's program files on the servers in the web farm.

1. Download the latest version of Wxcoder. It is available by ftp in the file *wxcoder.tar.gz* at ftp://taz.crh.noaa.gov/wxcoder/install/wxcoder.tar.gz. Place it in Apache's 'DocumentRoot' directory (see Apache's httpd.conf for the specific location).
2. 'untar' *wxcoder.tar.gz*
   tar –xvf wxcoder.tar.gz
3. Set file permissions for WxCoder's directories
   cd *DocumentRoot*
   chown –R wxcoder wxcoder
   chmod –R 777 wcxoder
4. WxCoder is now installed and awaiting the installation of its database

It will be necessary to create WxCoder's database using the files *wxcoder.sql*, *wxcoderData.sql*, and *wxcoderDataSync.sql*, as needed. The files are available by ftp at ftp://taz.crh.noaa.gov/wxcoder/install/wxcoder.sql, etc. Place the files in the /tmp directory of the server holding the MySQL server.

1. Create the database
   a. % **mysql –u root –p < /tmp/wxcoder.sql** (*% indicates the shell prompt)*
   b. You will be prompted for "root's" password; after it is entered the database will be created using the contents of wxcoder.sql
2. Populate the database using the command
   % **mysql –u root –p < /tmp/wxcoderData.sql**

3. Configure WxCoder global database parameters
   a. Several global database parameters in WxCoder's php configuration file need to be defined. Go to *DocumentRoot*/wxcoder/include and open *_config.php* in a text editor. Comments included in the file contain helpful notes about completing the defines. The following variables have been defined for the default installation; you may change them as appropriate.

```
$dbNAME          // MySQL WxCoder II database name
$dbHOSTNAME[0]   // 'read' database server name/IP address
$dbHOSTNAME[1]   // 'write' database server name/IP address
$dbHOSTNAME[2]   // backup write database server name/IP address
$dbUSERNAME      // username for connection to the databases
$dbPASSWORD      // database password

$xmtLDAD         // full filepath for output files for LDAD transmission
$xmtFTP          // DNS name/IP address of ftp sever  for transmission
$xmtFTPusername  // username for connection to server above
$xmtFTPpassword  // password to complete connection to ftp server
$pdfHOSTHANE     // server name/IP address of PDF-generating server
```

   b. The WxCoder database is now be configured (and loaded) for use

Update Software Installation

The following steps are necessary for successfully updating WxCoder's program files on the servers in the web farm.

1. Download the latest updates to Wxcoder. They are available by ftp in the file *wxcodeUpdater.tar.gz* at ftp://taz.crh.noaa.gov/wxcoder/install/wxcoderUpdate.tar.gz.
2. 'untar' *wxcoderUpdate.tar.gz*

```
tar –xvf  wxcoderUpdate.tar.gz
```

It will be necessary to update WxCoder's database using the file *wxcoderUpdate.sql*, which is available by ftp at ftp://taz.crh.noaa.gov/wxcoder/install/wxcoderUpdate.sql. Place the files in the /tmp directory of the server holding the MySQL server.

1. Commit the updates to the database by executing the following command

```
% mysql –u root –p  < /tmp/wxcoderUpdate.sql
```

Troubleshooting

This section provides solutions/answers for some of most common discrepancies encountered in the operation of WxCoder.

When an observer attempts to access the WxCoder login page, he encounters a screen that reads **WxCoder is currently unavailable!** *(The database server is 'down')*.

WxCoder must have access to its database at the very instant the user reaches the login page (to create and store a session ID). If WxCoder cannot make a connection to its database, the message above is returned to the observer.

The most common reason that WxCoder cannot connect to its database is that the MySQL database server is 'down.' The remedy for this error is to restart the database server. The following command sequence will serve to restart the MySQL server.

An observer using **Netscape 4.7** (the standard NWS browser) reports that he cannot access WxCoder or that the pages cannot be read.

WxCoder implements several DHTML features that are not supported by old versions of the Netscape browser. In particular, Netscape v4.7 is not supported. WxCoder will support browsers that implement DHTML. For the most common browsers, WxCoder supports Internet Explorer v5.5 or greater and Netscape v7.0 or greater.

```
CREATE TABLE `codesMet` (
  `codeID` varchar(8) NOT NULL default '0',
  `code` varchar(8) NOT NULL default '',
  `value` varchar(8) NOT NULL default '',
  `graphic` blob,
  `decode` varchar(32) NOT NULL default '',
  `longDecode` varchar(255) default NULL
) TYPE=MyISAM COMMENT='meteorological codes, values, decodes';


CREATE TABLE `codesNames` (
  `code` varchar(8) NOT NULL default '0',
  `name` varchar(16) NOT NULL default ''
) TYPE=MyISAM COMMENT='codes vs names';


CREATE TABLE `dataCoop` (
  `dataID` int(10) unsigned NOT NULL auto_increment,
  `siteID` varchar(16) NOT NULL default '0',
  `date` date NOT NULL default '0000-00-00',
  `time` varchar(6) NOT NULL default '',
  `datetime` varchar(14) NOT NULL default '',
  `type` varchar(16) NOT NULL default '',
  `units` enum('english','metric') NOT NULL default 'english',
  `AF` varchar(16) default NULL,
  `AM` varchar(16) default NULL,
  `EP` varchar(16) default NULL,
  `GD` varchar(16) default NULL,
  `GT` varchar(16) default NULL,
  `HP` varchar(16) default NULL,
  `HG` varchar(16) default NULL,
  `HI` varchar(32) default NULL,
  `HS` varchar(32) default NULL,
  `HT` varchar(16) default NULL,
  `IC` varchar(16) default NULL,
  `IT` varchar(16) default NULL,
  `PP` varchar(16) default NULL,
  `PPM` varchar(16) default NULL,
  `PPW` varchar(16) default NULL,
  `PT` varchar(16) default NULL,
  `QI` varchar(16) default NULL,
  `QT` varchar(16) default NULL,
  `RT` varchar(16) default NULL,
  `SD` varchar(16) default NULL,
  `SF` varchar(16) default NULL,
  `SI` varchar(16) default NULL,
  `SW` varchar(16) default NULL,
  `TA` varchar(16) default NULL,
```

```
  `TBIRZN` varchar(32) default NULL,
  `TBIRZX` varchar(32) default NULL,
  `TN` varchar(16) default NULL,
  `TPIRZN` varchar(16) default NULL,
  `TPIRZX` varchar(16) default NULL,
  `TSIRZN` varchar(32) default NULL,
  `TSIRZX` varchar(32) default NULL,
  `TVIRZN` varchar(32) default NULL,
  `TVIRZX` varchar(32) default NULL,
  `TX` varchar(16) default NULL,
  `UD` varchar(16) default NULL,
  `UG` varchar(16) default NULL,
  `ULD` varchar(16) default NULL,
  `UP` varchar(16) default NULL,
  `UR` varchar(16) default NULL,
  `US` varchar(16) default NULL,
  `XR` varchar(16) default NULL,
  `XRIRZN` varchar(16) default NULL,
  `XRIRZX` varchar(16) default NULL,
  `XW` varchar(16) default NULL,
  `XWD` varchar(32) default NULL,
  `ZC` text,
  `ZR` text,
  `coded` varchar(255) default NULL,
  `entered` varchar(14) default NULL,
  `entryID` smallint(5) unsigned NOT NULL default '1',
  `xmitted` varchar(14) default NULL,
  PRIMARY KEY  (`dataID`),
  UNIQUE KEY `OneOb` (`siteID`,`datetime`,`type`)
) TYPE=MyISAM COMMENT='storage for coop observer data';


CREATE TABLE `dataMarine` (
  `dataID` int(10) unsigned NOT NULL auto_increment,
  `siteID` varchar(64) NOT NULL default '',
  `date` date NOT NULL default '0000-00-00',
  `time` time NOT NULL default '00:00:00',
  `type` varchar(16) NOT NULL default '',
  `entered` timestamp(14) NOT NULL,
  `submitID` smallint(5) unsigned NOT NULL default '0',
  PRIMARY KEY  (`dataID`)
) TYPE=MyISAM COMMENT='storage for marine observer data';


CREATE TABLE `email` (
  `emailID` smallint(5) unsigned NOT NULL auto_increment,
  `wfoID` varchar(16) NOT NULL default '',
  `fromm` varchar(255) NOT NULL default '',
  `too` varchar(255) NOT NULL default '',
  `cc` varchar(255) default NULL,
  `bcc` varchar(255) default NULL,
  PRIMARY KEY  (`emailID`),
  UNIQUE KEY `wfoID` (`wfoID`)
) TYPE=MyISAM COMMENT='WFO email addresses';
```

```
CREATE TABLE `emailText` (
  `subjectID` varchar(32) NOT NULL default '',
  `language` enum('en','es') NOT NULL default 'en',
  `subject` varchar(128) NOT NULL default '',
  `text` text NOT NULL,
  `html` text,
  PRIMARY KEY  (`subjectID`,`language`)
) TYPE=MyISAM COMMENT='standard text for email sent from WxCoder (multi-language)';


CREATE TABLE `entryThrough` (
  `entryID` smallint(5) unsigned NOT NULL default '0',
  `method` varchar(64) NOT NULL default '',
  PRIMARY KEY  (`entryID`)
) TYPE=MyISAM COMMENT="source of observation posted to WxCoder's data tables";


CREATE TABLE `formControls` (
  `controlID` smallint(5) unsigned NOT NULL auto_increment,
  `type` varchar(16) NOT NULL default '',
  PRIMARY KEY  (`controlID`)
) TYPE=MyISAM COMMENT='HTML controls that may placed on entry forms';


CREATE TABLE `formGroupings` (
  `groupingID` tinyint(3) unsigned NOT NULL auto_increment,
  `type` varchar(16) NOT NULL default '',
  `grouping` varchar(128) NOT NULL default '',
  `elements` varchar(255) NOT NULL default '',
  `es` varchar(132) default NULL,
  `fr` varchar(132) default NULL,
  PRIMARY KEY  (`groupingID`)
) TYPE=MyISAM COMMENT='obElement groupings for placement on entry forms';


CREATE TABLE `formObElementDescriptions` (
  `elementID` varchar(16) NOT NULL default '',
  `obTypeID` tinyint(3) unsigned NOT NULL default '0',
  `en` varchar(64) NOT NULL default '',
  `es` varchar(64) default NULL,
  `fr` varchar(64) default NULL,
  PRIMARY KEY  (`elementID`)
) TYPE=MyISAM COMMENT="descriptions for an entry form's observational elements";


CREATE TABLE `formObElements` (
  `elementID` varchar(16) NOT NULL default '',
  `obTypeID` tinyint(3) unsigned NOT NULL default '0',
  `setableRange` enum('na','minmax','min','max') NOT NULL default 'na',
  `controlID` smallint(5) unsigned NOT NULL default '0',
  `controlInfo` varchar(64) NOT NULL default '',
  `controlUnits` varchar(32) NOT NULL default '',
  `controlUnitsMetric` varchar(32) NOT NULL default '',
  `units` varchar(32) NOT NULL default '',
  `unitsMetric` varchar(32) NOT NULL default '',
  `helpInfo` varchar(32) NOT NULL default '',
```

```
  PRIMARY KEY  (`elementID`)
) TYPE=MyISAM COMMENT='observational elements that may be placed on entry forms';


CREATE TABLE `formOptions` (
  `optionID` smallint(5) unsigned NOT NULL auto_increment,
  `selectname` varchar(64) default NULL,
  `text` varchar(64) default NULL,
  `value` varchar(64) default NULL,
  `exclude` tinyint(1) default '0',
  PRIMARY KEY  (`optionID`)
) TYPE=MyISAM COMMENT='options that may be placed in form "select" controls';


CREATE TABLE `limits` (
  `limitID` mediumint(8) unsigned NOT NULL auto_increment,
  `wfoID` varchar(16) NOT NULL default '',
  `obTypeID` tinyint(3) unsigned NOT NULL default '0',
  `elementID` varchar(16) NOT NULL default '',
  `min` varchar(16) NOT NULL default '',
  `max` varchar(16) NOT NULL default '',
  PRIMARY KEY  (`limitID`)
) TYPE=MyISAM COMMENT='WFO-specific range limits for obElements';


CREATE TABLE `log` (
  `logID` int(10) unsigned NOT NULL auto_increment,
  `sessionID` varchar(32) NOT NULL default '',
  `datetime` timestamp(14) NOT NULL,
  `siteID` varchar(16) NOT NULL default '',
  `username` varchar(16) NOT NULL default '',
  `action` varchar(255) NOT NULL default '',
  `location` varchar(255) default NULL,
  `extraInfo` varchar(255) default NULL,
  `errDescription` varchar(255) default NULL,
  PRIMARY KEY  (`logID`),
  KEY `errNo` (`logID`)
) TYPE=MyISAM COMMENT="WxCoder's transaction log";


CREATE TABLE `obType` (
  `obTypeID` tinyint(3) unsigned NOT NULL auto_increment,
  `type` varchar(16) NOT NULL default '',
  PRIMARY KEY  (`obTypeID`)
) TYPE=MyISAM COMMENT='observation types supported by WxCoder';


CREATE TABLE `region` (
  `regionID` char(1) NOT NULL default '',
  `name` varchar(128) NOT NULL default '',
  `city` varchar(128) NOT NULL default '',
  `stateID` char(2) NOT NULL default '',
  `lon` varchar(16) NOT NULL default '',
  `lat` varchar(16) NOT NULL default '',
  `elev` varchar(16) NOT NULL default '',
  `tzID` char(1) binary NOT NULL default '',
```

```
  PRIMARY KEY  (`regionID`)
) TYPE=MyISAM COMMENT='NWS regional office information/definitions';


CREATE TABLE `sites` (
  `siteID` varchar(16) NOT NULL default '',
  `siteNumber` varchar(32) default NULL,
  `name` varchar(128) NOT NULL default '',
  `county` varchar(64) default NULL,
  `state` char(2) NOT NULL default '',
  `type` varchar(16) NOT NULL default '',
  `obType` varchar(32) NOT NULL default '',
  `obTime` varchar(16) NOT NULL default '',
  `obTZ` char(1) NOT NULL default '',
  `obElements` varchar(255) NOT NULL default '',
  `obElementFlags` int(10) unsigned NOT NULL default '0',
  `obUnits` enum('english','metric') NOT NULL default 'english',
  `xmitID` varchar(32) NOT NULL default '',
  `reports` set('B91','B17','B83a') default NULL,
  `lon` varchar(16) default NULL,
  `lat` varchar(16) default NULL,
  `elev` varchar(16) default NULL,
  `WFO` varchar(16) NOT NULL default '',
  PRIMARY KEY  (`siteID`),
  KEY `WFO` (`WFO`)
) TYPE=MyISAM COMMENT='WxCoder observation site information';


CREATE TABLE `sitesRiver` (
  `siteID` varchar(16) NOT NULL default '',
  `river` varchar(64) NOT NULL default '',
  `gageType` enum('Staff','Wire weight','LARC') NOT NULL default 'Staff',
  `gageZero` varchar(32) default NULL,
  `floodStage` varchar(8) default NULL,
  `normalPool` tinyint(3) unsigned default NULL,
  `obTime` tinyint(3) unsigned NOT NULL default '0',
  `wfoID` varchar(16) NOT NULL default '',
  PRIMARY KEY  (`siteID`)
) TYPE=MyISAM COMMENT='   additional information for river observation sites';


CREATE TABLE `states` (
  `stateID` char(2) NOT NULL default '',
  `state` varchar(32) NOT NULL default '',
  `FIPS` varchar(4) NOT NULL default '',
  `zipCode` varchar(45) NOT NULL default '',
  `areaCode` varchar(150) NOT NULL default '',
  PRIMARY KEY  (`stateID`)
) TYPE=MyISAM COMMENT='US state information/definitions';


CREATE TABLE `tz` (
  `tzID` char(1) binary NOT NULL default '',
  `name` varchar(16) NOT NULL default '',
  `timezone` varchar(8) NOT NULL default '',
  `utcOffset` smallint(6) NOT NULL default '0',
```

```
  PRIMARY KEY  (`tzID`)
) TYPE=MyISAM COMMENT='time zone information/definitions';


CREATE TABLE `users` (
  `userID` varchar(32) NOT NULL default '',
  `username` varchar(16) NOT NULL default '',
  `fname` varchar(128) default NULL,
  `lname` varchar(128) default NULL,
  `company` varchar(128) default NULL,
  `address1` varchar(128) default NULL,
  `address2` varchar(128) default NULL,
  `city` varchar(128) default NULL,
  `state` char(2) default NULL,
  `zip` varchar(10) default NULL,
  `phone` varchar(16) default NULL,
  `phoneExt` varchar(8) default NULL,
  `email` varchar(128) default NULL,
  `emailType` enum('text','HTML') default 'text',
  `language` enum('en','es','fr') default 'en',
  `updated` datetime NOT NULL default '0000-00-00 00:00:00',
  PRIMARY KEY  (`userID`),
  UNIQUE KEY `username` (`username`)
) TYPE=MyISAM COMMENT='WxCoder user information';


CREATE TABLE `usersAdmin` (
  `userID` tinyint(3) unsigned NOT NULL auto_increment,
  `wfoID` varchar(16) NOT NULL default '',
  `fname` varchar(128) default NULL,
  `lname` varchar(128) default NULL,
  `position` varchar(32) default NULL,
  `address1` varchar(128) default NULL,
  `address2` varchar(128) default NULL,
  `city` varchar(128) default NULL,
  `state` char(2) default NULL,
  `zip` varchar(10) default NULL,
  `phone` varchar(16) default NULL,
  `phoneExt` varchar(8) default NULL,
  `fax` varchar(16) default NULL,
  `email` varchar(128) default NULL,
  `emailCC` varchar(255) default NULL,
  PRIMARY KEY  (`userID`)
) TYPE=MyISAM COMMENT='WxCoder admin users information';


CREATE TABLE `wfo` (
  `wfoID` varchar(16) NOT NULL default '0',
  `name` varchar(128) NOT NULL default '',
  `stateID` char(2) NOT NULL default '',
  `lon` varchar(16) default NULL,
  `lat` varchar(16) default NULL,
  `elev` varchar(16) default NULL,
  `tzID` char(1) binary NOT NULL default '',
  `regionID` enum('A','C','E','P','S','W') NOT NULL default 'C',
  PRIMARY KEY  (`wfoID`)
```

```
) TYPE=MyISAM COMMENT='WFO information/definitions';


CREATE TABLE `wxsamPages` (
  `pageID` smallint(5) unsigned NOT NULL auto_increment,
  `url` varchar(128) NOT NULL default '',
  `title` varchar(128) NOT NULL default '',
  `allowProfiles` smallint(5) unsigned NOT NULL default '0',
  PRIMARY KEY  (`pageID`)
) TYPE=MyISAM COMMENT='WxCoder Security & Administrative Manager: protected pages';


CREATE TABLE `wxsamProfiles` (
  `profileID` smallint(5) unsigned NOT NULL default '0',
  `siteType` varchar(64) NOT NULL default '',
  PRIMARY KEY  (`profileID`)
) TYPE=MyISAM COMMENT='WxCoder Security & Administrative Manager: access profiles';


CREATE TABLE `wxsamSessions` (
  `sessionID` varchar(32) NOT NULL default '0',
  `expires` int(11) unsigned default NULL,
  `data` text NOT NULL,
  PRIMARY KEY  (`sessionID`)
) TYPE=MyISAM COMMENT='WxCoder Security & Administrative Manager: session data';


CREATE TABLE `wxsamSettings` (
  `settingsID` smallint(5) unsigned NOT NULL auto_increment,
  `setting` varchar(64) NOT NULL default '',
  `value` varchar(128) NOT NULL default '',
  PRIMARY KEY  (`settingsID`)
) TYPE=MyISAM COMMENT='WxCoder Security & Administrative Manager: system settings';


CREATE TABLE `wxsamUsers` (
  `userID` varchar(32) NOT NULL default '',
  `username` varchar(16) NOT NULL default '',
  `password` varchar(16) NOT NULL default '',
  `tempPassword` tinyint(1) unsigned NOT NULL default '0',
  `siteID` varchar(128) NOT NULL default '',
  `multipleSites` tinyint(1) unsigned NOT NULL default '0',
  `rights` smallint(5) unsigned NOT NULL default '0',
  `lockcount` tinyint(3) unsigned NOT NULL default '0',
  `active` tinyint(3) unsigned NOT NULL default '0',
  `created` datetime NOT NULL default '0000-00-00 00:00:00',
  `lastAccessed` datetime NOT NULL default '2000-00-00 00:00:00',
  PRIMARY KEY  (`userID`),
  UNIQUE KEY `username` (`username`)
) TYPE=MyISAM COMMENT='WxCoder Security & Administrative Manager: user information';


CREATE TABLE `xmit` (
  `xmitID` tinyint(3) unsigned NOT NULL auto_increment,
  `method` enum('ftp','disk') NOT NULL default 'ftp',
```

```
  `extraMethod` varchar(16) default NULL,
  `delay` tinyint(1) unsigned NOT NULL default '0',
  `ftpHost` varchar(64) default NULL,
  `ftpUserID` varchar(16) default NULL,
  `ftpPassword` varchar(16) default NULL,
  `remoteSiteDirectory` varchar(128) default NULL,
  PRIMARY KEY  (`xmitID`)
) TYPE=MyISAM COMMENT='data transmission information/definitions';


CREATE TABLE `xmitID` (
  `wfoID` varchar(16) NOT NULL default '0',
  `coop` varchar(16) NOT NULL default '',
  `marine` varchar(16) default NULL,
  PRIMARY KEY  (`wfoID`)
) TYPE=MyISAM COMMENT='AFOS-style IDs used for product transmission to AWIPS';
```

Appendix B: WxCoder Directories and Files

WxCoder uses a combination of about 225 files totaling approximately 1.5 MB to facilitate its operation. The files are stored in a directory structure as show below.

wxcoder
wxcoder/es
wxcoder/images
wxcoder/include
wxcoder/include/help
wxcoder/include/jpgraph
wxcoder/include/reports
wxcoder/include/templates
wxcoder/WFO and its subdirectories

/wxcoder

This is WxCoder's root directory. The files in this directory are those needed to access WxCoder's user and admin function at the topmost level. Many of the files serve only as a 'switchboard' to redirect the server to a more task-specific file located in the restricted /wxcoder/include directory.

The directory also holds files associated with WxCoder security and administrative features -- *_init.php* and *_restrict.php*. These files limit access to any file which 'include' them to registered WxCoder users.

filename	description
_init.php	a lite version of _restrict.php that establishes a db connection
_restrict.php	restricts access to a page that "includes" it; inits a session
account.php	facilitates operations necessary for user account maintenance
admin.php	'switchboard' to admin functions appropriate for admin user type
dataEntry.php	'switchboard' to data entry functions appropriate for observer
favicon.ico	icon for IE bookmarked 'Favorites'
forgot.php	generates "forgot my user identifier" page
help.php	generates all help pages
index.html	alias for login.php; redirects to login.php
login.html	alias for login.php; redirects to login.php
login.php	generates user login page
logout.php	generates user logout page
logoutAdmin.php	generates admin logout page
multipleSites.php	facilitates presentation of multiple sites for appropriate users
noCookies.php	informs user of requirement/technique to allow cookies
noRights.php	informs user of inadequate rights to view a page
noScripting.php	informs user of requirement/technique to allow Javascript
popup.php	generates popup windows
print.php	'switchboard' to print functions
reports.php	'switchboard' to report generating functions
success.php	generates 'successful operation' for all observer pages
tempPassword.php	facilitates use of one-time user identifier at login
unavailable.html	displayed when there is no connection to the WxCoder database

WxCoder.admin.js	reserved for future use
WxCoder.css	WxCoder style sheet
WxCoder.js	Javascript used by WxCoder pages
WxCoder.user.js	reserved for future use

/wxcoder/es

Though considerable text is placed on each of its pages, WxCoder needs no external sense of language to operate. As a result it can insert text in any appropriate language as part of its runtime customization. It only needs to know what the preference is for displaying its text at login. The files in this directory are login aliases that set the preferred language to Spanish.

filename	description
index.html	sets language to Spanish; redirect to login.php in root directory
login.html	sets language to Spanish; redirect to login.php in root directory
login.php	sets language to Spanish; redirect to login.php in root directory

/wxcoder/images

WxCoder's images are stored in this directory.

filename	description
banner0.jpg	general purpose WxCoder banner
banner1.jpg	user WxCoder banner with menu item 1 selected
banner2.jpg	user WxCoder banner with menu item 2 selected
banner3.jpg	user WxCoder banner with menu item 3 selected
banner4.jpg	user WxCoder banner with menu item 4 selected
banner5.jpg	user WxCoder banner with menu item 5 selected
banner6.jpg	user WxCoder banner with no menu items selected
bannerEmail.jpg	WxCoder banner to include in HTML email
bannerPopup0.jpg	WxCoder banner to include in popup windows
btnClose.gif	'Close Window' button for use in help popup windows
btnDelete.gif	'Delete' button
btnDetails.gif	'Details' button
btnDisplay.gif	'Display' button
btnUpdate.gif	'Update' button
bug.gif	default NWS 'bug'
bug. jpg	default NWS 'bug'
cookiesIE6.gif	how to enable cookies for IE6
cookiesNS6.gif	how to enable cookies for NS6
getAcrobat.gif	link-associated image for Adobe Acrobat site
help.jpg	link-associated image to open a help popup window
hr.jpg	horizontal rule
poweredByMySQL.png	MySQL logo
poweredByPHP.gif	PHP logo
scriptingIE5.gif	how to enable Javascript for IE5
scriptingNS6.gif	how to enable Javascript for NS6
sortASC.gif	link-associated image to sort a column in ascending order
sortDESC.gif	link-associated image to sort a column in descending order

top.jpg	link-associated image to return to the top of a page
underConstruction.gif	gif for pages with incomplete code
vhr.jpg	vertical rule
warn.gif	animated gif to highlight a data entry error
warn2.gif	Headline bullet for error messages
WxCoder.gif	large WxCoder logo
WxCoder.ico	WxCoder icon
WxCoder2.gif	small WxCoder logo

/wxcoder/include

The 'meat' of WxCoder is stored in this directory. The naming convention used for filenames is suggestive of the function performed by the code in each of the files.

Within the files listed below, the majority implement code used to support WxCoder administrative functions. At the top level, administrative switchboard functions generate a menu appropriate for the user. These functions are prefixed with *admin*; the remainder of the filename denotes the type of admin user menu functions implemented by the code. At the next level, administrative functions/files are broken into three broad categories – user management, site management, and options. Files supporting these functions will be prefixed with *user*, *site*, or *options*; the remainder of the filename describes the features/function implemented by the code contained in the file.

filename	description
_class.WxSAM.php	the WxCoder Security and Administrative Manager
_class.mysql.php	wrapper functions to the WxCoder MySQL database
_config.php	defines db connection params; inits a WxCoder session
_sessions.php	callback functions for PHP's session handler
about.php	generates WxCoder's 'about' page
adminMaintenance.php	'switchboard' to all maintenance user admin functions
adminNational.php	'switchboard' to all national user admin functions
adminRegion.php	'switchboard' to all regional user admin functions
adminSuccess.php	generates 'successful operation' for all admin pages
adminSuperuser.php	'switchboard' to all superuser admin functions
adminWFO.php	'switchboard' to all WFO admin functions
class.FastTemplate.php	FastTemplate template functions
class.cryptRC4.php	RC4 functions for user identifier encryption
class.csv.php	generates comma-separated data files for download
class.email.php	email functions using sendmail
class.fileUpload.php	admin user file uploading functions
class.vbMsgBox.php	class to generate VB-like message boxes
class.xls.php	class to generate Excel-compatible output
class.xlsDB.php	class to generate Excel-compatible output from database query
commonForms.php	common form-generation functions
dataEntryCoop.php	facilitates data entry operations for coop observers
dataEntryMarine.php	facilitates data entry operations for marine observers
dataEntrySpotter.php	facilitates data entry operations for spotters
defines.php	WxCoder program constants
encodeSHEF.php	encodes observations in SHEF
mainRegion.php	generates regional administrative menu
mainSuperuser.php	generates superuser administrative menu
mainWFO.php	generates WFO administrative menu
messages.en.php	WxCoder English language string definitions

messages.es.php	WxCoder Spanish language string definitions
messages.fr.php	WxCoder French language string definitions
optionsLimits.php	‘switchboard’ to setting data limits for appropriate user type
optionsLimitsCoop.php	facilitates setting data limits for coop sites
optionsLimitsMarine.php	facilitates setting data limits for marine sites
optionsLimitsNA.php	notifies that setting data limits is not available for this observer type
optionsMessage.php	‘switchboard’ to setting type-appropriate localized logout message
optionsMessage2.php	facilitates setting localized logout message for coop observers
optionsMessagePreview.php	facilitates preview popup of localized logout message
optionsPassword.php	facilitates resetting admin user identifier
optionsSettings.php	facilitates setting WxCoder global values
optionsUpdate.php	facilitates setting WFO information
optionsViewLog.php	facilitates viewing the system log
reportsCoop.php	‘switchboard’ to coop observer report generation
reportsCoopB91.php	Facilitates generation/download of WS Form B-91
reportsCoopDwnld.php	Facilitates generation/download of observer data in Excel format
siteAdd.php	‘switchboard’ to appropriate type of site to add to WxCoder
siteAddCoop.php	facilitates addition of a coop site
siteAddMarine.php	facilitates addition of a marine site
siteAddSpotter.php	facilitates addition of a spotter site
siteDelete.php	facilitates deletion of a site
siteDetailsPopup.php	facilitates popup window displaying site-specific information
siteLinkMultiple.php	facilitates linking multiple site to a single observer
siteShow.php	presents a list of all WFO’s subordinate sites
siteUpdate.php	‘switchboard’ to updating information for a site
siteUpdateCoop.php	facilitates updating coop site information
userAdd.php	facilitates addition of a user to WxCoder
userBackupDataEntry.php	‘switchboard’ to backup of observer data entry by an administrator
userBackupDataEntryCoop.php	facilitates backup data entry for coop observer
userDelete.php	facilitates deletion of an observer from WxCoder
userDetailsPopup.php	facilitates popup window displaying user-specific information
userReports.php	‘switchboard’ to user reports to be generated
userReportsActivityLog1.php	facilitates stage 1 of generating user activity report
userReportsActivityLog2.php	facilitates stage 2 of generating user activity report
userReportsActivityLogPopup.php	displays popup window with user activity
userReportsActivityLogPrint.php	facilitates printing of user activity popup window
userResendWelcome.php	facilitates resending of welcoming email to subordinate user
userSendEmail.php	facilitates sending of email to all subordinate users
userSetReset.php	facilitates user account reactivation/disabling
userShow.php	presents list of all WFO’s subordinate users
userTempPassword.php	facilitates generation of a temporary user identifier
userUpdate1.php	facilitates stage 1 of updating a user account
userUpdate2.php	facilitates stage 2 of updating a user account
validateDataAccount.php	validates entries on form for user account information
validateDataAddUpdateCoop.php	validates entries on forms for adding/updating coop information
validateDataAddWFO.php	validates entries on form to add a WFO
validateDataCoop.php	validates coop data entries; adds ob to database
validateDataMarine.php	validates marine data entries; adds ob to database
validateDataOptionsLimits.php	validates entries on form to set data limits
validateDataOptionsUpdate.php	validates entries on form to update WFO information
validateDataUpdateCoop.php	Validates entries on form to update coop site information
validateDataUserSendEmail.php	validates information for sending bulk email to observers
validateDataUserUpdate.php	validates user information update entries
validationChecks.php	miscellaneous validation functions (date, email, zip code, etc.)
wfoAdd.php	facilitates additions of a WFO account to WxCoder
wfoDelete.php	facilitates deletion of a WFO account from Wxcoder
wfoShow.php	presents a list of all WFO sites
wfoUpdate1.php	facilitates stage 1 of updating a WFO account

wfoUpdate2.php	facilitates stage 2 of updating a WFO account
xmit.php	facilitates relay/transmission of WxCoder observation to AWIPS

/wxcoder/include/help

WxCoder provides considerable context-sensitive help. It is usually accessed by clicking the Help glyph that appears near the item of interest. The help text is provided in these files. With one exception the files are text files containing both the help text and layout instructions (included in each file).

filename	description
hlpAccount.txt	help for user account maintenance
hlpAddCoop.txt	help for operations related to adding a user
hlpAddUser.txt	admin help for adding a user account
hlpAddWFO.txt	admin help for adding a WFO account
hlpDew.txt	help with dew-related topics
hlpGeneral.txt	help with general data entry topics
hlpMisc.txt	help with user topics not covered in other files
hlpPcpn.txt	help with precipitation-related topics
hlpReports.txt	help with report generation topics
hlpRiver.txt	help with river-related topics
hlpSoil.txt	help with soil temperature-related topics
hlpTemp.txt	help with temperature-related topics
hlpUpdateWFO.txt	admin help for updating WFO information
HlpWind.txt	Help with wind-related topics
hlpWxCoder.txt	top-level menu help
synopticCodes.html	complete list of NWS synoptic codes

/wxcoder/include/jpgraph

WxCoder has the capability to generate charts and graphs. The feature is implement through the non-commercial use of the graphing library JPGraph (http://www.aditus.nu/jgraph). This directory contains files included in the standard distribution of the software.

/wxcoder/include/reports

WxCoder allows observers to generate several standard NWS forms that are filled-in with data they have previously entered into the WxCoder database. The files are generated as Adobe Acrobat PDF files. A commercial software package, PDFLib, is used to implement this feature. PDFLib allows data to be placed on pre-existing PDF files. This directory contains PDF versions of the various NWS forms used in this process.

filename	description
B-91.pdf	NWS Form B-91

/wxcoder/include/templates

Although WxCoder pages are customized for each user at runtime, the general layout of a page remains constant from user to user. There is a need to retain the common elements (e.g., page title, menus, buttons) while at the same time allowing flexibility for inclusion of a wide range of (e.g., type and number of data entry boxes, site-specific information). To achieve this level of flexibility, WxCoder makes extensive user of templates – in particular FastTemplate, a widely used PHP package.

The basic idea behind FastTemplate is that a single page consists of multiple logical parts. FastTemplate differentiates between these logical pieces (templates) by giving each piece of the page a name – a template name. Within a template, there may be any number of variables that are replace at runtime with the actual data/information that will make up the final page output.

The list below names the templates (file extension .tpl) that WxCoder uses to generates its pages. The files define the template used by the corresponding WxCoder .php file to generate page output.

about.tpl	optionsSettings.tpl
account.tpl	optionsUpdate.tpl
adminMaintenacne.tpl	reportsCoop.tpl
adminNational.tpl	siteAdd.tpl
adminRegion.tpl	siteAddCoop.tpl
adminSuccess.tpl	siteAddMarine.tpl
adminSuperuser.tpl	siteAddSpotter.tpl
adminWFO.tpl	siteDelete.tpl
adminWFOlite.tpl	siteDetailsPopup.tpl
dataEntryCoop.tpl	siteLinkMultiple.tpl
dataEntryMarine.tpl	siteShow.tpl
feedback.tpl	siteUpdate.tpl
feedbackAdmin.tpl	siteUpdateCoop.tpl
feedbackBackupDataEntry.tpl	success.tpl
forgot.tpl	userActivityLogAdmin.tpl
forgotSent.tpl	userAdd.tpl
forgotTryAgain.tpl	userBackupDataEntry.tpl
login.tpl	userBackupDataEntryCoop.tpl
logout.tpl	userDelete.tpl
logoutAdmin.tpl	userDetailsPopup.tpl
main.tpl	userReports.tpl
mainPopup.tpl	userReportsActivityLog1.tpl
mainPopup2.tpl	userReportsActivityLog2.tpl
mainPrint.tpl	userReportsActivityLogPopup.tpl
mainRegion.tpl	userReportsActivityLogPrint.tpl
mainSuperuser.tpl	userResendWelcome.tpl
mainWFO.tpl	userSendEmail.tpl
multipleSites.tpl	userSetReset.tpl
noRights.tpl	userShow.tpl
optionsLimits.tpl	userUpdate1.tpl
optionsLimits2.tpl	userUpdate2.tpl
optionsLimitsNA.tpl	wfoAdd.tpl
optionsMessage.tpl	wfoDelete.tpl
optionsMessage2.tpl	wfoShow.tpl
optionsMessagePreview.tpl	wfoUpdate1.tpl
optionsPassword.tpl	wfoUpdate2.tpl

/WFO & its subdirectories

This directory contains a subdirectory for each WFO and region. The subdirectories are used to store customization files. Typical files that would be placed here include a localized logout file (.txt) for each type of observer supported by the site along with any images (.gif, .jpg) that are included with the logout text. The WFO will also place is customized page 'bug' (bug.jpg) in this directory.