WESTERN REGION TECHNICAL ATTACHMENT
NO. 03-07
June 9, 2003

IFPS Server/Directory Documentation

Shelby Sharples, Western Region Headquarters, Salt Lake City, Utah

As part of implementing the gridded forecast WR has implemented a new web farm to support the more complex demands of gridded data.

This paper addresses the following issues:
1. New hardware configuration
2. New directory structure – where the files are
3. New security procedures – how a webmaster accesses these pages to work on them, and what they can and cannot not do
4. High level description of how the IFPS software works
5. The proper way to put links from public pages to digital forecast pages
6. The proper way to feed data from local GFE to the Web Farm

1. NEW HARDWARE CONFIGURATION (Fig.1)

Western Region HQ has changed its hardware configuration for IFPS functions. Instead of running the information on maul, a new Linux box, Gridiron, has been implemented for this purpose. Right now only IFPS related information runs on Gridiron. There are two main parts to the new hardware configuration. The two parts are Gridiron, the computer, and IFPS, the web server cluster.

Gridiron:

Western Region HQ has implemented a Red Hat Linux 8.0 computer as a front end device. This computer is named Gridiron. Responsibilities and capabilities of Gridiron include:

- Secure Shell (SSH)
- File Transfer Protocol (FTP)
- Firewall

- Cache (Proxy)
- Load balancer

All IFPS related traffic, either going in or out, must pass through Gridiron.

The IFPS Cluster:

Gridiron is designed to support clusters of web servers. A cluster is a group of servers containing the same information. The cluster running on Gridiron right now is IFPS. The IFPS cluster has two servers: ifps1 and ifps2. ifps1 acts as the primary server and ifps2 as the secondary. This means that ALL data comes in through Gridiron by either FTP or SSH and is stored directly on ifps1. As soon as ifps1 starts getting data, it immediately copies it to ifps2. This way, every server contains the exact same data.

2. NEW DIRECTORY STRUCTURE (Fig. 2)

Just like the hardware reconfiguration, the directories needed to be reconfigured. The new directories can be categorized according to function. The two categories are **data** and **software**.

A) The data category contains the *images*, *tmp*, and *data* directories. These three directories contain all the gridded weather data and dynamically created images. The files in these directories are updated in the cache much more frequently than files that fall under the software category and don't need to be backed up.

images: The images directory contains all dynamically created png image files. These types of images include the meteogram and the zoomed static images that are created when a customer selects the zoom option from the digital forecast page.

tmp: The tmp directory is where all temporary weather data, coming directly from the local office GFE, is stored. Files will be sent in a compressed netCDF format. It takes approximately 6 minutes to upload and process each new gridded weather data file. Because this process takes this long, a Perl program running in the tmp directory waits and watches until the whole file is uploaded. Once it is uploaded, the information is copied to the other web servers in the cluster. Each server then unzips the data file and sends it to the data directory.

data: The data directory is where all gridded weather data is stored until it is replaced by new information from the tmp directory. When a user requests a forecast, Gridiron pulls the weather information from this directory and produces the specific forecast.

B) The software category consists of the *html, perl, cgi-bin*, and *static* directories. These directories contain programs and files that need not be updated or backed-up as often as files in the data category because the information isn't changing nearly as often.

html:	The html directory is where all html documents reside. This can include any IFPS format pages.
perl:	The perl directory is for any mod-perl scripts. Perl scripts here will run under apache's mod-perl.
cgi-bin:	Web scripts reside here. cgi-bin also contains scripts that reside at a higher level than individual sites. These scripts are run by every digital forecast page.
static:	The static directory contains the static forecast area images and information. These standard files include: gfeweb.conf, shapes.txt, CITYLIST, GFE_SIDstatic.net, and the shape directory.

Having the directories broken up this way creates better security as well as making the system easier to maintain concerning caches and backup procedures.

3. NEW SECURITY PROCEDURES

To improve security, Gridiron is set up so that users have their own individual account with individual logins and passwords. Gridiron recognizes these logins and directs the user to their own directory where the information for the users' forecast office only will be located. In this way, only the people from a specific forecast office will have write permissions to their forecast office data. Users may read other forecast office directories and files, but will only have write permissions to their own. All users will need to contact WRH-SIB (801-524-5120) directly to get set up with an account and password.

Each user will have subdirectories in their home directory containing links to their sites directory structure. This will make it easy to navigate the site. Here is an example:

```
~rweatherly/
_ifps.slc.www/
    in this subdirectory will be:
            static
            data
            cgi-bin
            html
            tmp
            perl
            images
```

each subdirectory will link to the users own site directory. For example, in this home directory,

_ifps.slc.www/static actually links to .../static/slc.

There will also be an _ifps.slc.dev subdirectory. This subdirectory contains links to the development directories. These directories will be configured exactly the same as the production directories, but will not be accessible by the public. This is where users should test scripts and web pages before moving them to the www(active) directories. To test changes point browsers to http://ifps-dev.wrh.noaa.gov along with the file/path you are testing. To place files in the active directory use a move/copy command and move/copy from _ifps.xxx.dev to _ifps.xxx.www.

4. HIGH LEVEL DESCRIPTION OF HOW THE IFPS SOFTWARE WORKS

The forecaster, using a GFE workstation, creates a forecast by editing spatial and temporal grids. The GFESuite System created by FSL allows for forecasts of temperature, winds, sky cover, etc., at thousands of individual points around the U.S. without the forecaster actually typing in the forecasts. Once the forecaster is confident with the forecast, it is published and sent directly to ~user/ifpsxxx.www/tmp. This data is then uploaded, and sent to the data directory.

When a customer accesses data at a specific location and duration via the Internet, Gridiron receives all requests then passes them to one of the ifps servers which contain the data to create a forecast. The following programs, all residing in /cgi-bin, are run to create this forecast: (program names in **bold**)

User Input Functions:

dwf (Digital Web Forecast)
-is a Perl program developed by Don Britton and Randy Weatherly, which acts as the main, browser-based user interface. **dwf** is *recursive program,* which means that is calls itself, responds to arguments submitted to it, and wraps all the results within a consistent HTML interface. Users select options via an HTML form (e.g., choosing table, meteogram, duration, interval, zoom, etc). Upon submission, **dwf** examines the arguments, makes any necessary calculations, reads the appropriate configuration files, executes one of more of the programs listed below, and presents the output wrapped within the **dwf** HTML page.

Users can bookmark the URL created by **dwf** when a forecast is created so that subsequent forecasts can be created using the same point. (This URL could also be used by resourceful programmers who wish to collect digital weather forecasts programmatically.)

mapGen – **mapGen** is a compiled C program written by the Salt Lake forecast office ITO, Randy Weatherly. This program generates new png images when a customer requests a zoomed map. These new images are created with the center of the map being the point selected by the customer. **mapGen** is also used to create a site's main.png file for the main page. On gridiron, **mapGen** is on each user's path. When run by hand, it can be used to generate a sites main.png file. The syntax to create the main.png file is: mapGen –w SID.

Data display:

findCWA

If x/y or lat/lon coordinates are submitted to **dwf**, **findCWA** is called and executed to determine which WFO's CWFA the selected point falls within. This is done because each site's map extends beyond it's own CWA and a user could click in a neighboring CWA and gridiron must know where to look for the data. **findCWA** needs two special files. These files are **cwa.conf** and **cwa.png**. Both of these files reside in the upper level of the /cgi-bin directory.

Text with map:

gfe2txt

The text is generated by a C program written by the Boise forecast office WCM, Paul Flatt, called **gfe2txt**. This program is: passed a –w SID and uses either an (x,y) coordinate system, or lat, long coordinates. This program gets the coordinates found by the **mapGen** program.

Table with map:

weatherTable.pl
ll2ptdst
ct

weatherTable.pl is a perl script written by Randy Weatherly. It calls **ll2ptdst** and **ct** using the coordinates found by **mapGen**. It then uses to output of those programs to create an html table.

ll2ptdst is a C program written by Paul Flatt that calculates the distance to the nearest city from CITYLIST. The output of **ll2ptdst** is displayed at the top of the table, telling the user the requested location relative to the nearest city.

ct is a C program that creates an ascii table of weather element data. See enclosure

gfw (*g*ridded *f*orecast for the *w*eb)
- is a perl program that accepts input from the form created by the **dwf** program (see above). **gfw** reads and parses netCDF digital weather forecast files created by the IFPS/GFE system. **gfw** produces usable output for web pages in a variety of formats, including a plain-text table suitable for command-line use, a CSS formatted HTML table, XML, and an SVG (Scalable Vector Graphics) based meteogram.

gfw accepts a number of arguments to control the following parameters:

- site identifier (e.g., siteID=SLC)
- output format
  - html (e.g., format=html)
  - xml (e.g., format=xml)
  - meteogram (e.g., format=meteogram)
  - plain text table (e.g., format=table - Note: this output option will only work when gfw is called from the command-line)

- latitude and longitude (e.g., lat=47.29 lon=-111.33)
- duration of forecast in hours (e.g., dur=72)
- interval of forecast in hours (e.g., int=6 - Note: this option available soon)

Normally, gfw is called by the dwf program. However, gfw, can also be called from the command-line (e.g., ./gfw siteID=TFX lat=47.29 lon=-111.33 format=xml int=3 dur=48).

gfw relies upon a home-grown perl module named IFPSGrid.pm to handle reading the GFE-generated netCDF digital forecast file. IFPSGrid.pm can easily be used by other perl applications to read these files. In addition, the SVG-based meteogram is generated with the help of another perl module named svgMeteo.pm. For information on how to use these modules in other programs, contact Don Britton.

5. THE PROPER WAY TO PUT LINKS FROM PUBLIC PAGES TO THESE PAGES

Under Forecast on each sites welcome page, a new link labeled "Prototype Digital Forecast" should be added. The link should be the absolute path of http://ifps.wrh.noaa.gov/cgi-bin/dwf?siteID=XXX where XXX is your 3 letter identifier. All Forecast Offices should have everything set up and ready to roll. Any questions can be referred to Ed Ridley (SSD). Don Brittion also created a web page that lists links to all forecast office prototype pages. Sites can include this URL somewhere on your page: http//ifps.wrh.noaa.gov/wrlinks.html

6. THE PROPER WAY TO FEED DATA FROM LOCAL GFE TO WEB FARM

The proper way to send new netCDF files to Gridiron is using Paul Flatt's **sendgfe.sh** Perl program. When a digital forecast is published and the official database is updated, the new netCDF files should be sent to Gridiron. This can be done by installing the **sendgfe.sh** script in the Generate Products pull down menu.

Instructions on how to install the **sengfe.sh** script:

1) Run the GFE as SITE, utilizing the GFE config file normally used in operations (gfeConfig).

2) Once the GFE comes up, from the GFE pull down menu select **Define Config and ifpIMAGE Files....** A pop up window will appear with all of the available GFE configuration files; right click the appropriate file (gfeConfig) to make modifications. Scroll down towards the end of the file to the PRODUCT GENERATION SCRIPTS section; here you will need to add an entry for SendGFE.sh that should be similar to:

```
Scripts = {

    "Send Grids to Web...:" +
    "/home/fxa/release/SendGFE.sh &",

    }
```

Note: You will need to modify the pathway to SendGFE.sh accordingly in the above entry. Please take care to not modify any other entries in the PRODUCT GENERATION SCRIPTS section.

Once SendGFE.sh is installed correctly, the netCDF files will be sent to Gridiron(web) every time the 'Send Grids to Maul' button is pushed. The 'Send Grids to Web' button will be located in the Generate Products pull down menu.