

# AWIPS II Localization

21 October 2009

*Disclaimer: This document is a rough draft of the contract deliverable that is due on February 17, 2010. This document is intended to provide the NWS with information to support testing prior to delivery of Task Order 11. It is a work in progress and will be updated periodically during TO11 as new material is developed and feedback is received from the NWS.*

*The TO11 deliverable document will be a draft and is to be finalized during the OTE process.*

## Table of Contents

1	Localization.....	2
1.1	Localization Overview.....	2
1.2	Localization Data Environment.....	3
1.3	Creating a new Localization.....	6
1.3.1	Creating a new CAVE site location configuration file.....	6
1.3.2	Map Scale Localization.....	7
1.3.3	SBN Data Ingest Localization.....	9
1.3.4	CAVE Menu Localization.....	10
1.3.5	GFE Localization.....	10
1.3.6	AvnFPS Localization.....	12
1.3.7	WarnGen Localization.....	13
1.3.8	TextDB Trigger Localization.....	15
1.3.9	D2D.....	18
1.3.10	AlertViz (Guardian).....	18
1.3.11	LAPS.....	18
1.3.12	MSAS.....	18
1.3.13	FFMP.....	18
1.3.14	SCAN.....	18
1.3.15	SAFESEAS.....	19
1.3.16	FOG.....	19
1.3.17	SNOW.....	19
1.3.18	Text Localization.....	19
1.3.19	Hydrology Applications.....	21
1.3.20	Rehosted Applications.....	22
2	CAVE Menu Customization.....	22
3	Radar Localization (Allows user to add/delete radar mosaic & change site).....	28
4	Warngen Template Customization.....	30

## 1 Localization

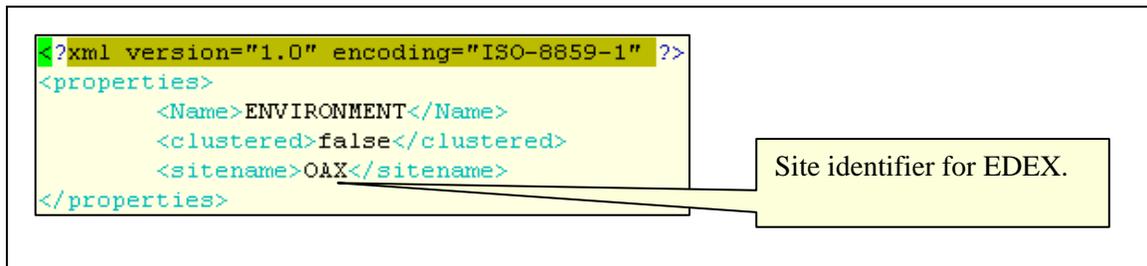
### 1.1 Localization Overview

Localization adapts (e.g. configures) the AWIPS national baseline software to the unique data and con-ops requirements of the site. AWIPS-II performs localization dynamically at system startup using data from the localization data environment.

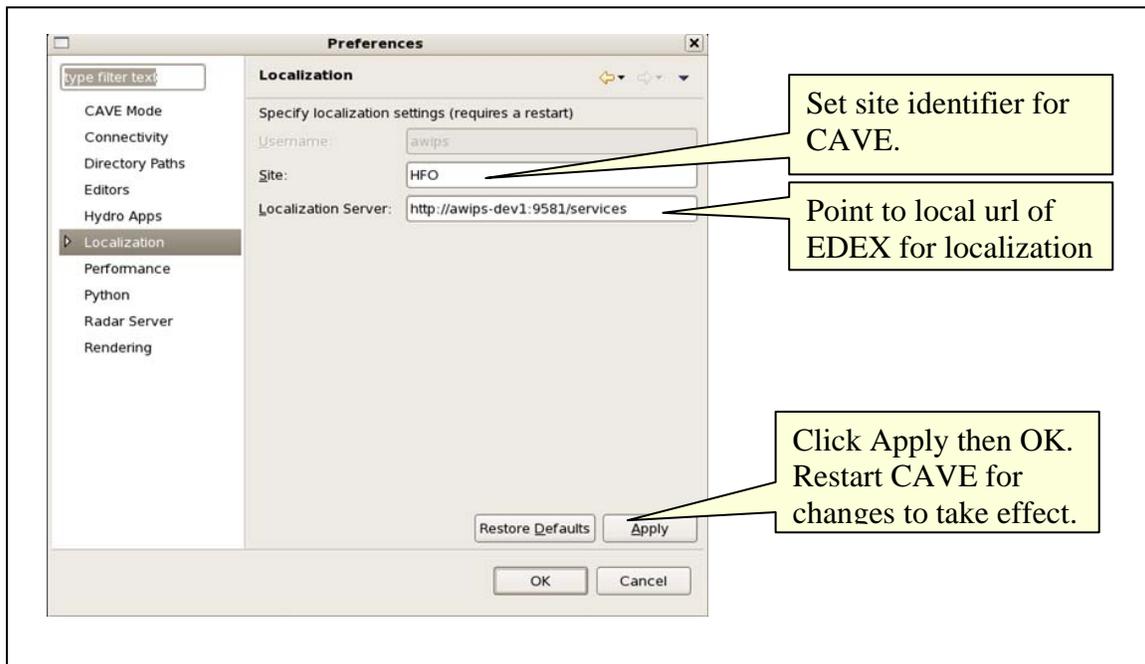
Once the localization data is set up for the primary site and the desired backup sites, the default site is then set in **environment.xml** at:

```
${EDEX_HOME}/conf/res/site
```

EDEX needs to be restarted when any changes are made to **environment.xml**. Since, all the server services run within the EDEX, all the EDEX services get restarted. Restarting EDEX takes around two minutes. The **sitename** element is used to reference the primary GFE server configuration files and ingest filters.

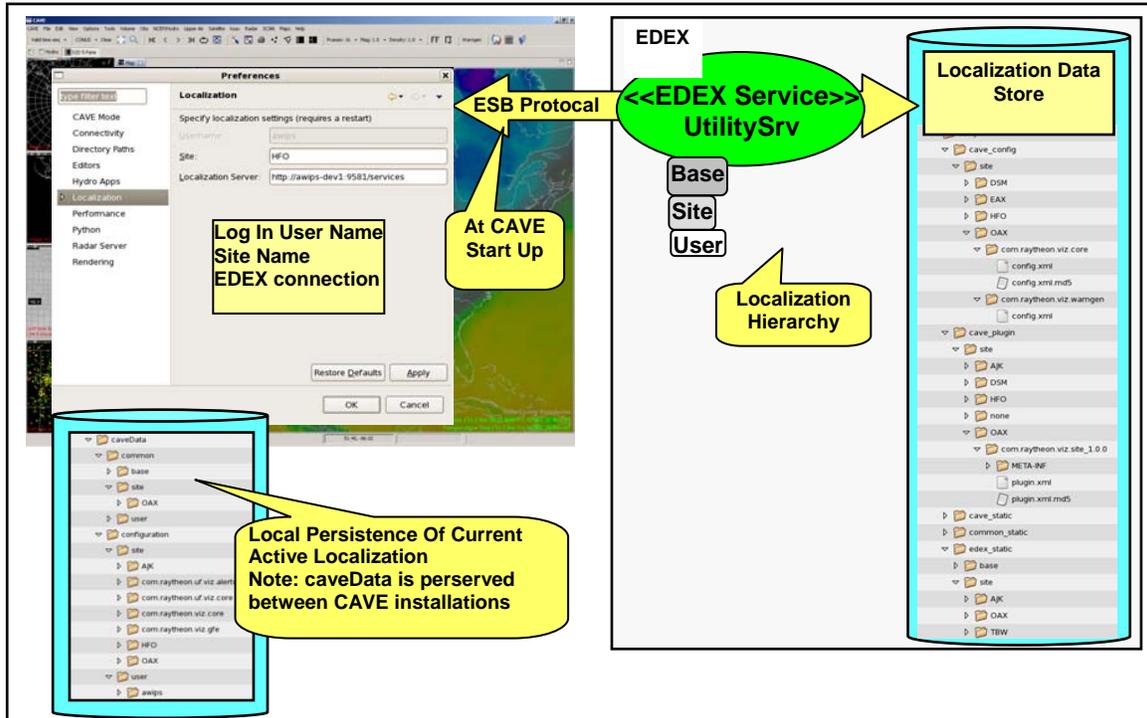


To change CAVE's active site, go to **CAVE>Preferences>Localization** to bring up the localization settings dialog.



## 1.2 Localization Data Environment

AWIPS-II has two major components, EDEX and CAVE. When CAVE starts up, the EDEX service UtilitySrv is contacted to serve out the localization data. EDEX, which provides the server functions, only has site level localization, since the server capabilities support many users. Note, however, that GFE services within EDEX are designed to support multiple sites simultaneously. Localization data are on the EDEX server in the utility data tree. The site sets up its localization by adding files to this tree. CAVE has both site and user level localization configuration files. The details of this data tree which support both EDEX and CAVE are described below.



CAVE plugins that are designed to be configurable have a **config.xml** file. The data in any **config.xml** can be overridden by the site. Likewise, each user can have overrides to the site **config.xml**. The utility service in EDEX copies the site and user overrides to the local CAVE instance into the **caveData** tree in the user's home directory during CAVE startup. The **caveData** tree is a local cache of localization data and improves the startup performance of CAVE after the initial startup. To force CAVE to reinitialize its localization, **caveData** can be deleted. Instructions on where and how to set this up are discussed below.

The following CAVE plug-ins are configurable using the base/site/user hierarchical localization pattern. Each listed CAVE plug-in has a unique **config.xml** which is site/user localizable. The location of the site/user versions are on the EDEX utility data tree and need to be created for a new localization as follows:

```

${EDEX_HOME}/data/utility/cave_config/site/XXX/plugin_name/
${EDEX_HOME}/data/utility/cave_config/user/UUU/plugin_name/
    
```

Where XXX = site name (e.g. OAX)  
UUU = user log in name

The base (national controlled) CAVE configuration data are stored directly in the plug-in jar files as a **config.xml** file and are immutable. The site and user overrides are in the utility data tree. The following CAVE plug ins are configurable:

CAVE plugin_name	Configuration File	Comment
com.raytheon.viz.core	config.xml	CAVE general configuration
com.raytheon.viz.warngen	config.xml	Warngen office
com.raytheon.viz.gfe	config.xml	GFE defaults
com.raytheon.uf.viz.alertviz	config.xml	AlertViz defaults
com.raytheon.viz.aviation	config.xml	AvnFPS setup config
com.raytheon.viz.avnconfig	config.xml	AvnFPS defaults
com.raytheon.viz.radar	config.xml	Radar general configuration
com.raytheon.viz.skewt	Config.xml	SkewT defaults

CAVE general purpose data is localizable at the following location:

```

${EDEX_HOME}/data/utility/cave_static/site/XXX/{data item}
${EDEX_HOME}/data/utility/cave_static/user/UUU/{data item}
  
```

The general purpose data that can be localized is as follows:

Localizable Data Item	Description
bundles/scales	The CAVE map scale XML bundles that control the map display.
colormaps	CAVE color table files in XML cmap format (note: a python tool “convCT.py” is available in the ADE at “build.cave/tools” to convert AWIPS I netcdf color tables to AWIPS-II XML cmap files.
gfe/combinations	Text formatter combination python files and edit area combinations.
gfe/userPython/textProducts	Text formatter override python files.
menus	Menu item overrides to the baseline

Common data is localizable at the following location:

```

${EDEX_HOME}/data/utility/common_static/site/XXX/
${EDEX_HOME}/data/utility/common_static/user/UUU/
  
```

The following data items can be localized in the common static tree:

Localizable Data Item	Description
-----------------------	-------------

editAreaGroups	GFE text files
editAreas	GFE edit areas in new XML format (Note: these files are created automatically from ERSI shape files)
hydro	Apps_defaults AWIPS I file of tokens that control applications
hydro/hydroapps	Various AWIPS I geo and basin data files
radar	radarsInUse.txt file for the RadarServer process
textdb	afosmasterPil.txt, textCategoryClass.txt, textCCChelp.txt, textNNNhelp.txt, textOriginTable.txt

EDEX only data is localizable at the following location:

`${EDEX_HOME}/data/utility/edex_static/site/XXX/`

The following data items can be localized in the edex\_static tree:

Localizable Data Item	Description
mesowest_filters.xml	Area filter definitions for the mesonet ingest plugin
config/gfe	localConfig.py and siteConfig.py configuration files

### 1.3 Creating a new Localization

Creating a new site localization from the baseline AWIPS-II installation involves the following steps:

- Creating a site CAVE location configuration file.
- Creating the site map scale files that become the CAVE map scales.
- Creating the site ingest filters for Local Data Manager (LDM).
- Creating the site menu overrides for the CAVE menus.
- Creating a GFE site server configuration and customizations.
- Creating a AvnFPS site configuration.
- Creating a WarnGen site configuration and customizing warning templates.
- Creating the site TextDB triggers.
- Creating the site AlertViz configuration.
- Creating LAPS site configuration
- Creating MSAS site configuration
- Creating the site decision aid configurations (FFMP, SCAN, SAFESEAS, FOG, SNOW)
- Hydro site configuration using AWIPS I Token files.

#### 1.3.1 Creating a new CAVE site location configuration file

CAVE configuration is localized through `config.xml` files that are part of the CAVE plugin design pattern. Set the overrides for the CAVE core plugin

(com.raytheon.viz.core) configuration file (config.xml). The CAVE core plugin configuration contains the basic site identifiers and location information. The override configuration file lives at the following location on the server:

```
`${EDEX_HOME}/data/utility/cave_config/site/XXX/${PLUG_IN}
```

The following is an example of the overrides to localize CAVE to OAX.

```
./utility/cave_config/site/OAX/com.raytheon.viz.core/config.xml
```

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<configuration>
<connectionMethod>http</connectionMethod>
<jmsServerAddress>tcp://localhost:61616</jmsServerAddress>
<httpServerAddress>http://localhost:9581/services</httpServerAddress>
<siteName>DEV</siteName>
<siteFullName>Development</siteFullName>
<siteType>DEV</siteType>

<defaultGraphicColor>green</defaultGraphicColor>
<defaultGraphicColor>coral</defaultGraphicColor>
<defaultGraphicColor>cyan</defaultGraphicColor>
<defaultGraphicColor>burlywood</defaultGraphicColor>
<defaultGraphicColor>yellow</defaultGraphicColor>
<defaultGraphicColor>violet</defaultGraphicColor>
<defaultGraphicColor>darkkhaki</defaultGraphicColor>
<defaultGraphicColor>OrangeRed</defaultGraphicColor>
<defaultGraphicColor>dodgerblue</defaultGraphicColor>

<homeLocationLongitude>-96.36666666666666</homeLocationLongitude>
<homeLocationLatitude>41.31666666666667</homeLocationLatitude>
<wfoCenterPointLatitude>41.31666666666667</wfoCenterPointLatitude>
<wfoCenterPointLongitude>-96.36666666666666</wfoCenterPointLongitude>
</configuration>
```

### 1.3.2 Map Scale Localization

Map Scales are XML display bundles that control base map projections in the CAVE display panes. Data gets transformed to the map scale as the data is being rendered. The CAVE-D2D perspective tool bar has the selection pull down for the map scales available to the site. AWIPS-II has two approaches for generating new map scales. First, is a tool that is included in the CAVE menu that contains all the map projections available within geotools. During installation world wide geopolitical map data is loaded into the postgisSQL database. Therefore, a map can be generated for any location in the world at any scale.

Note: Other map data such as cities will need to be loaded into the postgisSQL database.

The second approach, is to reuse the “SUP” files from the AWIPS I installation. The SUP files contain the details of the map projection which can be transformed into an AWIPS-II map scale bundle. A tool is planned to transform the AWIPS I data to AWIPS-II.

### **1.3.2.1 Map Scale Localization for a New Site**

To build a new map for your site, use CAVE’s built in map generation tool. AWIPS-II has the map data stored in PostgisSQL database tables. The database tables are initialized during system installation and contain a superset of information that the site needs. See paragraph x.x.x.x for instructions on how to load new map data from ERSI shape files into the PostgisSQL map repository.

To run the CAVE map generation tool: Go to **CAVE>New>Map Projection...** in the CAVE-D2D perspective to bring up the map generation dialog. Enter the projection parameters for your desired scale map. These projection parameters can be extracted from the AWIPS I installation’s SUP files.

Save the generated map bundle by using the save bundle dialog at:

**CAVE>Procedures>Save Bundle...**

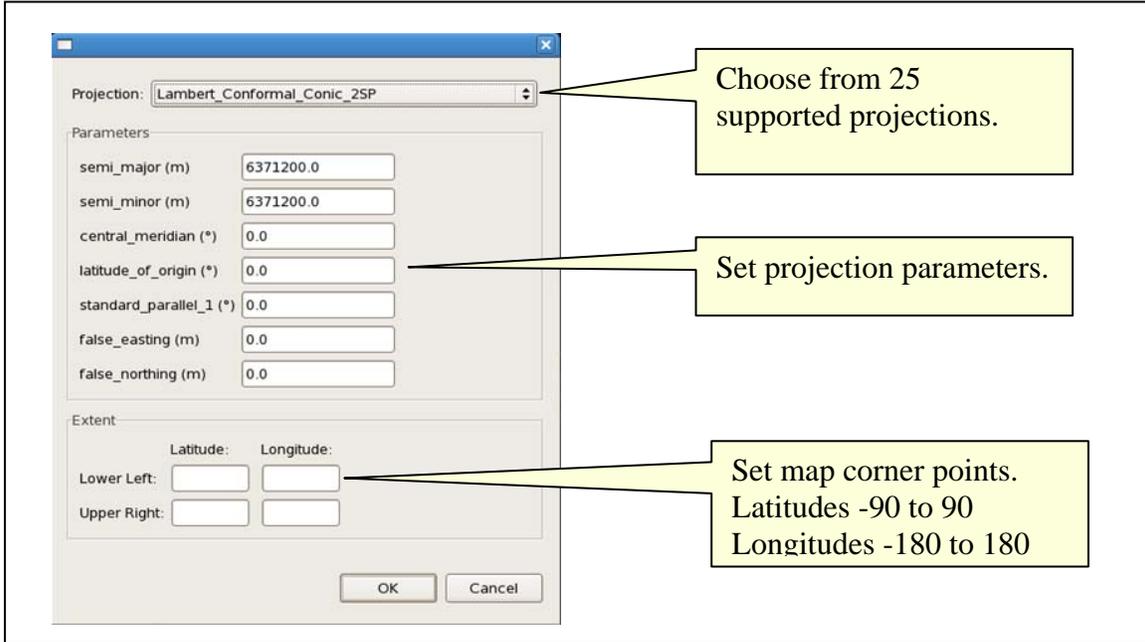
Within the save dialog browse to:

```
${EDEX_HOME}/data/utility/cave_static/site/XXX/bundles/scales/
```

Give the scale bundle a file name {e.g. CONUS.xml, N. Hemisphere.xml, North American.xml, Regional.xml, State(s).xml, WFO.xml...}. The names needs to match the default names that show up in the scale pull down in CAVE-D2D perspective. Note, the scale pull down names can be changed by modifying the **plugin.xml** located at:

```
${EDEX_HOME}/data/utility/cave_plugin/site/XXX/com.raytheon.viz.site_1.0.0
```

The scale bundle files only contain map and scale related metadata. The scale bundles know how to get the real map data out of the “PostgisSQL” database.



**Creating the default procedure**

The default procedure is the procedure that executes on CAVE-D2D perspective startup. This procedure can be anything you want but generally it loads the the desired map scale into each of the small panes and the main pane.

- Start CAVE and go to the D2D 5pane perspective
- Using the scale pull down on the tool bar select a scale for the main pane and dock it to the desired small pane
- Select and dock until you have the scales set up as desired
- Under CAVE>Procedures>Save Procedure... dialog box put “**default-procedure.xml**” into the name field and hit “OK”. Transfer the “**default-procedure.xml**” to  $\${EDEX\_HOME}/data/utility/cave\_static/site/XXX$  to make the procedure available to CAVE.

**1.3.2.2 Map Scale Localization for an Existing Site Using Legacy Data**

In progress.

**1.3.3 SBN Data Ingest Localization**

Localizing the SBN data flow involves modifying the Local Data Manager (LDM) configuration. LDM is used by AWIPS-II as a replacement of the CP ingest software. The file **pqact.conf** located usually at **/usr/local/ldm/etc** in the LDM installation contains the configuration information that tailors the data flow to AWIPS-II.

The file **pgact.conf** serves the same purpose as **acq\_patterns.txt** in AWIPS -1. The primary difference is configuring the text product data flow to specific products rather than directing everything to the text ingest end point. The unidata web site for LDM has detailed instructions on how to perform configuration. LDM is normally configured on AWIPS-II to queue data so that EDEX ingest endpoints can recover from periods of disruption.

**<http://www.unidata.ucar.edu/software/ldm/>**

- **Ingest Subgridding Localization:** Grid sub-grid definitions are available via XML files. The sub-grid definitions get placed in the localization data store at the following location:

**`${EDEX_HOME}/data/utility/edex_static/site/XXX/grib/subgrids`**

The sub-grid localization files define the sub-grid in grid coordinates starting from the upper left corner of the grid. Some initial examples are in the baseline for OAX. Each sub-grid is defined for a given model based on the title field of gribModels.xml located at:

**`${EDEX_HOME}/data/utility/common_static/base/grid`**

configured on AWIPS-II to queue data so that EDEX ingest endpoints can recover from periods of disruption.

#### ***1.3.4 CAVE Menu Localization***

The CAVE menus are highly customizable by XML files that override the base configuration. The menu override files get placed in the localization data store at the following:

**`${EDEX_HOME}/data/utility/cave_static/site/XXX/menus`**

Likewise for user overrides:

**`${EDEX_HOME}/data/utility/cave_static/user/UUUUU/menus`**

See the section on menu customization for how to create the overrides. Some initial examples are in the baseline for OAX and HFO.

#### ***1.3.5 GFE Localization***

Localizing GFE to your site reuses many of the same files that AWIPS I used. The following are the primary aspects that can be localized.

- The map scale visible in the spatial editor of GFE
- The siteConfig.py sets up the local high level GFESUITE identifiers
- The localConfig.py that overrides the base settings in ServerConfig.py
- The text product local overrides
- The local smart inits (these have been included as part of the local applications migration effort)
- The local smart tools (these have been included as part of the local applications migration effort)
- The local combinations files
- The local edit areas
- GFE Client /ifpImage Configuration

#### ***1.3.5.1 GFE Map Scale Localization***

All the GFE map scales for all the active AWIPS-I sites are stored in the database table “**gfe\_spatial**” in the PostgreSQL metadata database. The gfe map scales are referenced by the the siteid which is set in the “siteConfig.py” file for the site. See the next section for siteConfig.py localization.

#### ***1.3.5.2 GFE siteConfig.py localization***

The **siteConfig.py** files from AWIPS I are reusable by placing them at the following location in the localization data store:

```
${EDEX_HOME}/data/utility/edex_static/site/XXX/config/gfe
```

The fields in your **siteConfig.py** will need to be checked to make sure they are valid.

#### ***1.3.5.3 GFE localConfig.py localization***

The **localConfig.py** files from AWIPS I are reuseable by placing them at the following location in the localization data store:

```
${EDEX_HOME}/data/utility/edex_static/site/XXX/config/gfe
```

**LocalConfig.py** follows same pattern as AWIPS I by being overrides to the base **ServerConfig.py**. Generally this is where your site’s controls for ISC are located.

#### ***1.3.5.4 GFE local text product overrides***

The AWIPS I tools are reused to create text formatter product scripts. These are located at the following location:

```
${EDEX_HOME}/data/utility/edex_static/base/textproducts
```

Included at this location is a “Readme.txt” that gives detailed instructions on how to set up the site’s text formatters.

#### **1.3.5.5 GFE local smart inits**

The baseline smart inits are located at the following:

```
${EDEX_HOME}/data/utility/edex_static/base/smartinit
```

The local smartinits are being converted as part of the local apps migration effort. Where to store the local smartinits is TBD (i.e. to avoid overwriting them with an install)

#### **1.3.5.6 GFE local smart tools**

The local smart tools are being converted as part of the local apps migration effort. Where to store the local smart tools is TBD.

#### **1.3.5.7 GFE local combinations files**

In progress.

#### **1.3.5.8 GFE local edit areas**

The GFE edit areas are built automatically from the ERSI shape files when EDEX starts up for the first time with a new localization.

#### **1.3.5.9 GFE Client/ifpImage Configuration**

The GFE Client uses the same localization preferences as CAVE. These settings can be found at:

```
~/caveData/.metadata/.plugins/org.eclipse.core.runtime/.settings/
```

In the file: `localization.prefs`

### **1.3.6 AvnFPS Localization**

AvnFPS uses the AWIPS-II localization pattern for storing configuration files. The site specific files are stored on the utility data store at the following location:

```
${EDEX_HOME}/data/utility/cave_static/site/XXX/aviation
```

The “Save” buttons in the setup GUIs for AvnFPS save the locally edited setup data to the above location.

AvnFPS is localized entirely by the use of the setup dialogs, which are identical to the AWIPS I dialogues, and users do not need to concern themselves with files and where the files are stored.

- Editing Monitoring Rules
- Editing AvnFPS TAF Site Information
- Editing AvnFPS TAF Product Definition
- Creating AvnFPS Database Triggers
- Using the AvnFPS Text Editor
- Configuring GFE to Send Gridded Data to AvnFPS

The specific climate HDF5 files should be stored on the workstation at:

`${CAVE_HOME}/etc/aviation/climate`

### 1.3.7 WarnGen Localization

The following describes how to localize WarnGen to a specific WFO site. Localizing WarnGen requires access to the following data locations on the EDEX server:

- `${EDEX_HOME}/data/utility/cave_config/site` Contains general WarnGen configuration options.
- `${EDEX_HOME}/data/utility/common_static/base/warngen` Contains all WarnGen templates – including default (“baseline”) templates, local site templates, and backup site templates.

#### 1.3.7.1 WarnGen Localization Procedure

- Select **CAVE>Preferences** to open the CAVE preferences dialog. Under the **Localization** tab change the Site field to your site (e.g. OAX), then click **OK** and exit CAVE.



- Open a console to the EDEX server and cd to `${EDEX_HOME}/data/utility/cave_config/site/`
- Create the directory `{XXX}/com.raytheon.viz.warngen` where **XXX** is the site identifier.

- Copy in a new file into the above directory **config.xml** from an existing configuration such as:

```
${EDEX_HOME}/data/utility/cave_config/site/OAX/com.raytheon.viz.warngen  
/config.xml
```

- Edit the file as necessary.

### Example WarnGen Configuration File

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>  
<configuration>  
<warngenOfficeShort>OMAHA/VALLEY NE</warngenOfficeShort>  
<warngenOfficeLoc>OMAHA</warngenOfficeLoc>  
<backupCWAs>  
EAX/KANSAS CITY,  
DMX/DES MOINES,  
BOX/BOSTON,  
LBF/NORTH PLATTE,  
PQR/PORTLAND  
</backupCWAs>  
<siteNode>OMA</siteNode>  
<otherWarnGenProducts>  
Severe Weather Statement/SVS,  
Flash Flood Statement/ffs,  
Extreme Wind Warning/eww,  
Extreme Wind SVS/ewws,  
non-convective FFW (Dam Break)/dambreak,  
non-convective Flash Flood Statement/dambreakffs,  
Areal Flood Warning/flw,  
Areal Flood Warning Followup/fls,  
Areal Flood Advisory/fla,  
Areal Flood Advisory Followup/flas,  
Special Marine Warning/smw,  
Marine Weather Statement (SMW Follow)/smws,  
Marine Weather Statement standalone/marinestatement  
</otherWarnGenProducts>  
</configuration>
```

- **warngenOfficeShort:** The Office description included in the heading of a warning.
- **warngenOfficeLoc:** The short office description included in the heading of a warning.
- **BackupCWAs:** This is a list of the sites which WarnGen will be able to back up, along with a description of the site to be included in the template. Each item

- in this list must contain both values separated by “/” and the list elements should be comma-separated.
- **siteNode:** This field provides a default site node if one can not be looked up in the **afos2awips** table. If **afos2awips** contains the value then this field remains unused.
  - **otherWarngenProducts:** This is a list of the WarnGen products in the “other” dropdown list and the template associated with it. For example, the line “Extreme Wind Warning/eww” indicates that the dialog should show an option containing “Extreme Wind Warning” which refers to the “eww” template. Each item in this list must contain both values separated by “/” and the list elements should be comma separated. The items in this list are case sensitive.

***Add section for adding local points to WarnGen.***

### ***1.3.8 TextDB Trigger Localization***

These instructions provide a guide for setting up AFOS PIL triggers from scratch. AWIPS-II triggers provide the capability of running scripts in response to product arrival. This script running capability is a part of the EDEX server system.

There are two types of product arrival subscriptions available: AFOS PIL triggers and more generic data URI triggers. These instructions only refer to the AFOS PIL triggers. In AWIPS-II, when a text product is ingested and stored in the text database (fxatext), the product’s AFOS PIL is sent to the LDAD script runner. If the PIL matches the subscription, the corresponding script is executed.

Setting up an AFOS trigger involves the following steps:

1. Identify the AFOS PIL. AWIPS-II utilizes the same wild card/pattern matching as AWIPS-I.
2. Create the trigger script. In this case, the script can be any executable that satisfies two basic requirements; first, it must be accessible to the EDEX server. Secondly, it must be able to handle and process a single string argument; the AFOS PIL.
3. Register the trigger. On AWIPS-II, registering the trigger is accomplished by using the Command Line Interface (CLI) tools. AFOS PIL triggers are registered using the AWIPS-II “textdb” tool.
4. Validate the trigger. This can be done either using the CLI subscription tool or a standard SQL tool such as PGAdmin to check the database table.

When the AFOS PIL trigger executes, two things occur; first the contents of the triggering product are written to a file. The name of the file is the AFOS PIL that triggered the script execution; it is located in \$FXA\_DATA/trigger. FXA\_DATA is set into the environment when EDEX starts. Secondly, the registered script is executed with the trigger being passed to the script as the only argument.

When writing an AFOS PIL trigger script, keep in mind that EDEX\_HOME (usually /awips/edex) and FXA\_DATA (usually /awips/edex/data/fxa) are available to the script from the environment.

Note, FXA\_DATA is set by the **start.sh** script that starts EDEX. The EDEX installer modifies the script to match the location chosen when the installer runs.

### AFOS PIL Trigger Example:

Consider this scenario: You want to register an AFOS PIL trigger for the following products:

Origin (CCC): OMA  
 Category (NNN): MTR  
 Designator (XXX): any

When triggered, a script called metar.sh, located in the EDEX bin directory will run. This script will reside in the EDEX bin directory (/awips/edex/bin). For this example, metar.sh will simply log the event and the contents of the product triggering the event. The result will be logged to a file called metar-trigger.log located in the EDEX log directory.

1. Identify the PIL: In this case, the PIL to use is OMAMTRXXX.
2. Create the trigger script: The code for the trigger script is shown below. Note that the script assumes EDEX\_HOME and FXA\_DATA have been set in the environment; the EDEX setup script sets these values at startup.

```
#!/bin/bash
#-----
# Test Script for AFOS PIL Trigger
# Logs execution and product contents into a log file.
#-----
LOG_FILE=$EDEX_HOME/logs/metar-trigger.log
TIME_NOW=`date`
if [ -z $1 ]; then
  echo "$TIME_NOW: invalid process call, no AFOS PIL provided" >> $LOG_FILE
  exit 1
fi
AFOS_PIL=$1
if [ -z $FXA_DATA ]; then
  echo "$TIME_NOW: invalid configuration, \"$FXA_DATA\" not in environment" >> $LOG_FILE
  exit 1
fi
FILE_PATH=$FXA_DATA/trigger/$AFOS_PIL
if [ ! -f $FILE_PATH ]; then
  echo "$TIME_NOW: unable to find file matching AFOS PIL: $AFOS_PIL" >> $LOG_FILE
  exit 1
fi
echo "$TIME_NOW: processing $AFOS_PIL" >> $LOG_FILE
echo "$TIME_NOW: file $FILE_PATH contents:" >> $LOG_FILE
cat $FILE_PATH >> $LOG_FILE
exit 0
```

Save this script in /awips/edex/bin as metar.sh and make it executable.

3. Register the trigger: The CLI is used to register the AFOS PIL trigger. It can be registered using either the CLI textdb tool or the CLI subscription tool. The preferred method is to use the CLI textdb tool, which has been written to support the same options as the AWIPS I textdb tool.

To register this trigger, first change directory into the install directory of the CLI tools. (The location of the CLI tools may vary; on AWIPS II only systems, it is normally installed in /awips/fxa/bin. Make sure you identify the correct install directory; it will also include executable files named *subscription* and *uengine*.) Once in the correct directory, execute

```
./textdb -ldad -a OMAMTRXXX <EDEX_HOME>/bin/metar.sh
```

After a short pause, textdb should print "Database insert was successful." and exit.

Note: replace <EDEX\_HOME> with the EDEX install location, usually /awips/edex.

4. Validate the trigger: To verify that the insert was successful, run the trigger registered, run the following command from the command line:

```
./subscription -o read -p OMAMTRXXX
```

After a short pause, the tool will display:

ID	ACTIVE	TYPE	RUNNER	TRIGGER	SCRIPT
208	True	ldad	ldad	OMAMTRXXX	null

You can also verify the trigger by connecting to the AWIPS II metadata database (using a standard database tool) and executing the following SQL command (entered on a single line):

```
select * from subscription.subscriptions where trigger='OMAMTRXXX'
```

From *psql* the result is

id	active	type	runner	trigger	script
filepath			arguments		
208	t	ldad	ldad	OMAMTRXXX	
/common/mfegan/awips/edex/bin/metar.sh				%TRIGGER%	

This approach provides additional information; specifically the full path of the script and the argument passed to the script. In this case, *%TRIGGER%* indicates the trigger PIL is passed to the script.

- NWRWaves/CRS Triggers

- AvnFPS Triggers
- LDAD Triggers?
- Climate Triggers
- HazCollect Triggers
- METAR Triggers

**DataURI Trigger Example:** *To be added.*

### ***1.3.9 D2D***

D2D configuration that is done through GUIs will continue to be done the way they are with AWIPS I. Current AWIPS I documentation (e.g. UM) still applies.

### ***1.3.10 AlertViz (Guardian)***

AlertViz is configured the same way Guardian is and the AWIPS I documentation still applies.

### ***1.3.11 LAPS***

The migration of this application is to be done by the Government. Documentation for configuring the application will be provided by the Government with the application.

### ***1.3.12 MSAS***

The migration of this application is to be done by the Government. Documentation for configuring the application will be provided by the Government with the application.

### ***1.3.13 FFMP***

- **Install Basin Shape Files:** After the install of EDEX, the shapefiles for the RFC basins should be installed into tables called “ffmp\_basins” and “ffmp\_streams” according to your localization. If they have not been loaded or you wish to load more RFC basin shapefiles, you can use the tool “shapeToSQL.sh” to load additional basins.
- **Basic HUC Templates:** The HUC templates for your localization will be created automatically the first time the FFMP EDEX processor runs. This will add a brief delay to the first run of FFMP.

### ***1.3.14 SCAN***

- **Menu configuration:** All menu configuration for SCAN is handled in the “radarsInUse.txt” file. Any radar/terminal radar added to the “Scan” list can be processed by SCAN. See the section on radar configuration for more details on updating “radarsInUse.txt”.
- **Processing:** Processing for SCAN is preformed identically to the CAVE menu configuration for SCAN. Any radar/terminal radar added to the “Scan” menu in the “radarsInUse.txt” file can be processed by SCAN.

### 1.3.15 *SAFESEAS*

To be delivered with the completed application in the “snap-up”

### 1.3.16 *FOG*

To be delivered with the completed application in the “snap-up”

### 1.3.17 *SNOW*

To be delivered with the completed application in the “snap-up”

### 1.3.18 *Text Localization*

- **Modifying the Default Origination Node in the Text Editor:** To set or override the origination for sites in the AFOS Browser, add/update

```
${EDEX_HOME}/data/utility/common_static/site/XXX/textdb/textOriginTable.txt
```

Add/Update to have an entry for XXX Y, where Y is the default region for the site (R-Regional, W-West, E-East, C-Central, I-International). To change, set the default origin to the desired origin. [Note: This is currently not working, DR3303 addresses this.]

- **Adding Products to the Text Browser:** Products can be added to the TextBrowser by adding an xml file to the localization server. The path to the file and a sample of the file are below:

```
${EDEX_HOME}/data/utility/cave_static/site/XXX/menus/textws/products.xml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<menuTemplate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
<contribute xsi:type="command" menuText="Latest SFT"
commandId="com.raytheon.viz.texteditor.afosproduct">
  <parameter key="afoscommand" value="OMASFT000"/>
</contribute>
```

```
<contribute xsi:type="command" menuText="Latest PFM"
commandId="com.raytheon.viz.texteditor.afosproduct">
  <parameter key="afoscommand" value="OMAPFM000"/>
</contribute>
```

```
<contribute xsi:type="command" menuText="All SFT NE"
commandId="com.raytheon.viz.texteditor.afosproduct">
  <parameter key="afoscommand" value="OMASFTNE "/>
</contribute>
```

```
<contribute xsi:type="command" menuText="All PFM OAX"
commandId="com.raytheon.viz.texteditor.afosproduct">
  <parameter key="afoscommand" value="OMAPFMOAX"/>
```

```
</contribute>
```

```
</menuTemplate>
```

- **Adding Tools to the TextBrowser:** New tools can be added to the text browser by entering them into the `toolsMenu.xml` file listed below. A new python tool can be put into the directory specified below and linked to in the `toolMenus.xml`. The tools must extend `BaseTool.py`. An interface is given to make call backs to the text workstation.

The `toolsMenu.xml` location:

```
${EDEX_HOME}/data/utility/cave_static/site/XXX/menus/textws
```

New python tool location:

```
${EDEX_HOME}/data/utility/cave_static/site/XXX/menus/textws/python
```

Sample tools location:

```
${CAVE_HOME}/etc/textws/python
```

Sample `toolsMenus.xml`:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<menuTemplate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <contribute xsi:type="command" menuText="Print Latest TAFs: OMA SNY
VTN AFK" commandId="com.raytheon.viz.texteditor.launchtool">

    <parameter key="script" value="ExecuteAFOSCommands.py"/>

    <parameter key="args"
value="OMATAFOMA,OMATAFSNY,OMATAFVTN,OMATAFAFK"/>
  </contribute>

  <contribute xsi:type="command" menuText="Print Last 4 Hours of
FPUS63" commandId="com.raytheon.viz.texteditor.launchtool">

    <parameter key="script" value="ExecuteAwipsQuery.py"/>

    <parameter key="args"
value="wmoid=FPUS63,lasthours=4,fullread=true"/>
  </contribute>
</menuTemplate>
```

- **Update the “afos2awips.txt” file:** To update the afos2awips text entries, copy the file at  
`${EDEX_HOME}/data/utility/edex_static/base/textproducts/library/afos2awips.txt`

and add additional entries needed to the new file. Run the tool at

```
${EDEX_HOME}/bin/afos2awipstranslator.sh
```

passing it the path to the new `afos2awips.txt` file and redirecting the output to a file:

```
${EDEX_HOME}/bin/afos2awipstranslator.sh afos2awipsNew.txt >  
afos2awips.sql
```

Shutdown EDEX and execute the `afos2awips.sql` against the database to update the `afos2awips` table.

- **Purging Products:** The `stdtextproducts` table is purged every hour based on the `textProductInfo` table. If no data is in the `textProductInfo` table one is created with a default of 5 versions.

### *1.3.19 Hydrology Applications*

With few exceptions, AWIPS II localization for Hydrology applications is the same as AWIPS I and the existing AWIPS I documentation for the hydrology applications still applies. Hydrology applications, reengineered and rehosted, use the AWIPS I configuration file “App\_defaults”. The format of this file was preserved, and it is intended to be reused for site migration. This includes:

- HydroBase Manager.
- Obsfcst Monitor
- PrecipMonitor
- Dam Crest + Catalog Data Management
- Rate of change checker
- Build Hourly Precip Reports
- Database purger + File Purger
- FFG + QPE Mosaicking
- Flood Archiver
- Satellite Precipitation Estimate (SPE) Data Retrieval
- Grib Encoder
- Hi Resolution Precipitation Estimator (HPE)
- Hi Resolution Precipitation Nowcaster (HPN)
- Metar2Shef
- RiverPro
  - (1) RiverPro product Definition Files
  - (2) RiverPro template files
  - (3) RiverPro Special Phrase File
- XSETS
- RiverMonitor
- Timeseries + Time Series Lite
- SSHP (Site Specific Hydrologic Prediction Function)
- HydroGen

- PrecipMonitor
- XNAV
- XDAT
- IVP
- IFP
- ENS

Hydview Overlay configurations will change due the change in the use of maps in AWIPS II.

### ***1.3.20 Rehosted Applications***

The localization and configuration of these applications are the same as for AWIPS I, and existing documentation for AWIPS I still applies, except as otherwise noted in this document. The existing AWIPS I site localization/configuration files for these applications can be reused, for the most part, for site migration. Rehosted applications include:

- Climate
- NWRWAVES
- HazCollect
- NOAA Weather Wire Service (NWS) Interface
- Asynchronous Product Scheduler
- Administrative Message Handling System
- NWS Scheduler
- Hourly Weather Roundup (HWR)
- Local Data Acquisition and Dissemination (LDAD)
- River Pro
- RFC Applications
- RFC Historical Data Browser
- Hi Resolution Precipitation Estimator (HPE)
- Hi Resolution Precipitation Nowcaster (HPN)
- NCF Archive access.
- Local Storm Reporting (LSR)
- 4-D Storm Investigator (FSI)
- Legal Archiver
- Data Archiver
- AWIPS System Monitor

## ***2 CAVE Menu Customization***

The CAVE menu is designed to be highly customizable through XML files. The menu system allows considerable flexibility in structuring menus. It is possible to structure menus in as few or as many files as desired to improve manageability. The baseline

CAVE-D2D menu structure is defined in standard Eclipse RCP plugin.xml files in the following plug ins:

`com.raytheon.viz.ui.personalities.awips/plugin.xml`

`com.raytheon.uf.viz.d2d.ui/plugin.xml`

Individual CAVE-D2D plug ins build on these menus by making their own baseline contributions. The following baseline contributions for CAVE-D2D are as follows:

Plug in menu contribution path	CAVE-D2D menu items
menus/lightning	Lightning plots under Obs>Lightning
menus/radar	kxxx radar and Radar menus
menus/satellite	Satellite menus
menus/xml	Volume Browser sub menu
menus/warnings	Warning displays under Obs>Hazards
menus/mos	Local menus
menus/obs	Contributions in Obs
menus/upperair	Contributions in Upper Air

Localizing the menus involves adding your own XML files to the EDEX utility data tree at the following location:

`/${EDEX_HOME}/data/utility/cave_static/site/XXX/menu/...`  
`/${EDEX_HOME}/data/utility/cave_static/user/UUU/menu/...`

If you do not have any local contributions, this directory tree can be blank.

The following describes how to make your own site/user overrides to the baseline menu configuration. All menus are connected together using an “index” XML file that specifies which subfiles to use. For example, here is how overrides to the CAVE-D2D obs menu would be made for the site OAX. The file `index.xml` can be overridden at the site level to add or remove items without changing the “base” files. Create an `index.xml` file and put it at the following location on the EDEX utility tree:

`/${EDEX_HOME}/data/utility/cave_static/site/OAX/menus/obs/index.xml`

The following is an example of `index.xml` overrides:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<menuContributionFile>
  <include installTo="menu:#Levels?after=METAR"
    fileName="menus/obs/baseMetar.xml" />
  <include installTo="menu:obs?after=METAR"
    fileName="menu/obs/koaxMetar.xml" />
</menuContributionFile>
```

```
<include installTo="menu:obs?after=MARITIME"
  fileName="menus/obs/koaxBuoy.xml">
  <remove>FixedBuoys</remove>
  <remove>MovingMaritime</remove>
</include>
</menuContributionFile>
```

Two local additions were added:

- A site has added their own contribution files presumably adding additional menu items. (koaxMetar.xml and koaxBuoy.xml)
- A site has removed two menu items from the base configuration. (FixedBuoys and MovingMaritime)

The fact that the base file is not modified directly means that upgrades will be smoother without requiring the merging of new files into the site's custom files.

Menu item positioning is done by URI and placed using the "id" components. For example:

- **menu:obs?after=METAR** (finds the obs menu, and installs contributions after the separator or menu item ID'd as "METAR")

Now on to the menu files which like the **index.xml** files subscribe to a strict schema that is validated at parse time. (menu schema is available in CAVE in the **com.raytheon.uf.viz.menus...jar** plug in as menus.xsd)

For example, OAX decided to add the menu file **koaxBuoy.xml** which had a previous reference added to the **index.xml**:

```
#{EDEX_HOME}/data/utility/cave_static/site/OAX/menu/obs/koaxBuoy.xml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<menuTemplate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <contribute xsi:type="bundleItem"
file="bundles/maritimeFixedBuoy.xml"
  menuText="Fixed Buoys" id="FixedBuoys"
  productInterval="3600" productOffset="1800">
  <dataURI>/sfcobs/%/1005/%</dataURI>
  </contribute>
</menuTemplate>
```

This file installs a menu item with the label "Fixed Buoys" and id "FixedBuoys". The display bundle file "bundles/maritimeFixedBuoy.xml" will be executed. The dataURI reference "/sfcobs/%/1005/%" will be used for updating the display time in the menu and is used to control the query for the data when the menu item is selected.

The **productInterval** and **productOffset** refer to the time controls for updating the looping frame for the data display layer in the CAVE-D2D perspective. The **productOffset** is the time in seconds before the data's valid time that a new loop frame gets created. Then **productInterval** is the interval between frames for this data display layer. In the example above, for the "Fixed Buoys" display item, a new frame gets created 30minutes before the valid time and the frame is 60minutes long.

DataURIs in AWIPS-II refer to specific data items in the metadata database. The fields in the dataURI are metadata attributes with % being a wild card. The wild card (%) is typically used for time fields so the when the menu item is selected the default behavior of retrieving the latest data that matches the dataURI is retrieved.

DataURIs are created automatically during data ingest for each data item, get stored in the metadata database, and are used for real time alerting. The datatype plug in contains a record class that has java annotations for each field of the dataURI. The structure of the dataURI is as follows:

**/PlugInName/dataTime/position1/.../positonN**

The first field is always the PlugInName and the second field is always the dataTime. A wild card (%) is typically used in a menu reference for dataTime to indicate that the latest data is to be retrieved. The set of available PlugInNames are listed in column 1 in the following table. Column three lists the dataURI fields that get specified in **position1...positionN** for that particular plug in.

A tool is planned (DR#2891) for a latter release for generating this table based on the installed set of plug ins. This tool would be available from the CAVE menu.

PlugInName	Plug In Record Class	DataURI fields in Postion Order
airep	AirepRecord.java	reportType, corIndicator, location
binlightning	BinLightningRecord.java	startTime, stopTime
Bufrua	UAObs.java	reportType, corIndicator, location
Ccfp	CcfpRecord.java	validtime, issuetime, location
Gfe	GFERecord.java	parmId, dbId
Goessounding	GOESSounding.java	location
Grib	GribRecord.java	modelInfo
Obs	MetarRecord.java	reportType, stationed, stationLat, stationLon
Sfcobs	ObsCommon.java	reportType, corIndicator, location
Pirep	PirepRecord.java	reportType, corIndicator,

		location
Poessounding	POESSounding.java	location
Profiler	ProfilerObs.java	reportType, location
Radar	RadarRecord.java	icao, productCode, primaryElevationAngle,
Recco	RECCORecord.java	reportType, corIndicator, location
Redbook	RedbookRecord.java	wmoTTAAii, productId, corIndicator, originatorId, fcstHours, fileId
satellite	SatelliteRecord.java	Source, creatingEntity, sectorID, physicalElement
Taf	TafRecord.java	stationed, corIndicator, amdIndicator, issue_time
Text	TextRecord.java	productId
Warning	WarningRecord.java	pil, xxxid, countyheader, act, etn, seg, phensig
Cwat	CWATRecord.java	icao, fieldname
Qpf	QPFRecord.java	Icao, fieldname
Acars	ACARSRecord.java	tailNumber, location
Acarssounding	ACARSSoundingRecord.java	timeObs, location
mesowest	MESOWestRecord.java	networkType, timeObs, location

For example, to retrieve the latest visible satellite imagery, the following dataURI would be used in the menu reference:

```
<dataURI>/satellite/%/NESDIS/GOES%/%/Imager_Visible</dataURI>
```

Positon1...PositionN is defined in **SatelliteRecord.java**. Note, the reference to the GINI satellite ICD for the valid names to use.

```
/**
 * The source of the data - NESDIS
 */
@Column(length = 31)
@DataURI(position = 1)
@XmlAttribute
@DynamicSerializeElement
private String source;

/** The creating entity. See table 4.5 of GINI satellite ICD */
@Column(length = 63)
@DataURI(position = 2)
@XmlAttribute
@DynamicSerializeElement
private String creatingEntity;

/** The sector ID. See table 4.6 of the GINI satellite ICD */
@Column(length = 63)
@DataURI(position = 3)
@XmlAttribute
@DynamicSerializeElement
private String sectorID;
```

```

/** The physical Element. See table 4.7 of the GINI satellite ICD */
@Column(length = 63)
@DataURI(position = 4)
@XmlAttribute
@DynamicSerializeElement
private String physicalElement;

```

Variable substitution in the `index.xml` file allows you to reuse an entire menu with global substitutions. For example, the baseRadar menu system is reused but with a different icaos:

```

<menuContributionFile>
  <include installTo="menu:radar?after=LOC1"
    fileName="menus/lightning/baseRadar.xml"
    subMenu="koax">
    <substitute key="icao" value="koax" />
  </include>
  <include installTo="menu:radar?after=LOC1"
    fileName="menus/lightning/baseRadar.xml"
    subMenu="kdsms">
    <substitute key="icao" value="kdsms" />
  </include>
</menuContributionFile>

```

Submenus can be created in the menu files using the “submenu” type and nesting structures. For example:

```

<contribute xsi:type="subMenu" menuText="Other Plots">
  <contribute xsi:type="bundleItem"
file="bundles/15minSurfacePlot.xml"
    menuText="15 min Plot" id="15MinSurfacePlot"
    productInterval="900" productOffset="450">
    <dataURI>/obs/%</dataURI>
  </contribute>
</contribute>

```

Submenus can also be created by using the “submenu” tag on the <include> statement. For example:

```

<menuContributionFile>
  <include installTo="menu:obs?after=HAZARDS"
    fileName="menus/lightning/baseLightning.xml"
    subMenu="Lightning" />
</menuContributionFile>

```

Menu item types can be the following:

- Bundle Loader
- Separator (lines)
- Placeholder (displays “not implemented” dialog)
- Submenu

In general, the menu system described above can coexist with the standard Eclipse RCP menus which are defined in “plugin.xml” files. The Eclipse RCP plugin.xml files should be used for setting up the basic menu structures, and contributing items like dialogs and programs.

Menu files can be placed in CAVE plugins for the base localization at: “localization/menus/pluginName”.

### **3 Radar Localization (Allows user to add/delete radar mosaic & change site)**

Radar localization is handled by the text file, **radarsInUse.txt** in the localization store. When EDEX starts up the file is used to automatically modify the D2D menus, mosaicing, and FFMP radar controls. The file, **radarsInUse.txt** was added to simplify common localization actions that every site needs to perform and is located in the localization store at:

`#{EDEX_HOME}/data/utility/common_static/site/XXX/radar/radarsInUse.txt`

- **Change Local Site’s Radar:** In radarsInUse.txt there is a section called “# RadarServer Radars - MUST HAVE THIS LINE”. Following this, are the list the local radars using the ICAO identifier. These radar menus show up after the <Satellite> menu item in the D2D perspective.
- **Change Dail Radars:** Similar to the local sites’s radar except the radars will go into the “Radar” menu under the “Dail Radars” heading. It has the same menu as the RadarServer radars only that it does not get its own menu in CAVE.
- **Change Terminal Radars:** Similar to the local site’s radar. Each entry gets in own menu in CAVE.
- **Change FFMP Radars:** These sites are put in the SCAN menu and ard used for the FFMP program. They are under the heading “FFMP”. Any radar under the Radar Servers radars is also here, so it is not necessary to add those as well.
- **ARSR Radars and ASR Radars:** These two headings are for Airport Surveillance Radars and Air-Route Surveillance Radars. Not all sites have these. These are given submenus under the Radar menu, and have two entries under them.
- **Change Radar Mosaic Setup:** In radarsInUse.txt there is a section called “# Mosaic Radars - MUST HAVE THIS LINE”. Following this are the list of all the radars that are possible when mosaicing is performed. However, if there is no data for a site within this list then it will not be mosaiced. Edit this text file for mosaics and the new sites will now show up in mosaicing.
- **Change CWA:** This is a list of the country warning areas that FFMP will cover.

- **Change FFMP Run:** This is used for FFMP. It is a colon separated list of three different values. The first value is the radar icao that is needed to run FFMP. The second value is the default county warning area. The third value is the river forecast center (RFC) that is used in FFMP. The format for these is (using OAX as an example) `koax:OAX:KKRF`.

The following is an example of `radarsInUse.txt`.

```
# Mosaic Radars - MUST HAVE THIS LINE  
NOT EDIT LINES BEGINNING WITH '#'  
# RadarServer Radars - MUST HAVE THIS LINE  
ktxl
```

```
# Dial Radars - MUST HAVE THIS LINE  
kabr  
kcys  
kdmx  
kgld  
kuex  
keax  
karx  
kmpx  
klnx  
kdvn  
kudx  
kfsd  
ktxw
```

```
# Terminal Radars - MUST HAVE THIS LINE  
tmsp  
tmci
```

```
# FFMP Radars - MUST HAVE THIS LINE
```

```
# ASR Radars - MUST HAVE THIS LINE  
eeri
```

```
# ARSR Radars - MUST HAVE THIS LINE  
fqwa
```

```
# Mosaic Radars - MUST HAVE THIS LINE  
koax  
kabr  
kcys
```

kgld  
kuex  
kdmx  
keax  
karx  
kmpx  
klnx  
kdvn  
kudx  
kfsd  
ktwx

# CWA - MUST HAVE THIS LINE

OAX  
GID  
DMX  
FSD  
EAX

# FFMP Run - MUST HAVE THIS LINE

koax:OAX:KKRF

#### 4 WarnGen Template Customization

WarnGen templates for the AWIPS II system are \*.vm files written in the **Velocity Template Language (VTL)** with supporting configurations in an xml file with \*.cfg suffixes. Velocity and the VTL is released under the Apache Jakarta Project and thus the current documentation can be found on the Apache Jakarta Project website. The following URLs are direct links to the web-based VTL reference.

[http://svn.apache.org/repos/asf/velocity/engine/tags/V\\_1\\_0\\_1/docs/vtl-reference-guide.html](http://svn.apache.org/repos/asf/velocity/engine/tags/V_1_0_1/docs/vtl-reference-guide.html) <http://click.sourceforge.net/docs/velocity/vtl-reference-guide.html>

#### Configuration Values in WarnGen \*.cfg files.

*Values in red are system level settings and should only be modified if necessary*

- **polygonShape:** Use “1” for the pathcast shape based on storm track or “2” for a square
- **followups:** Defines the options which can appear in the “UPDATE LIST” dropdown
- **phenomena:** Defines which Phenomena this template can create Followup products for.
- **Significance:** Defines which Significance this template can create Followup products for.
- **DefaultDuration**

- **Durations**
- **Bullets**
- **bulletText:** Defines the text displayed in the WarnGen dialog.
- **bulletName:** Defines a name to refer to the bullet as in the template code.
- **bulletGroup:** Provides a mechanism for limiting selections in WarnGen. Only one item perbulletGroup can be selected so related items should list the same bulletGroup.
- **bulletType:** bulletType="title" makes the bullet unselectable, bulletType="basin" correlates the bullet to a geometry of the same name in the customlocations table.
- **pathcastConfig**
- **enabled:** "1" will enable the Storm Track, "2" will disable the Storm Track
- **defaultSpeedKt:** Default Storm Track speed in knots
- **defaultDirection:** Default Storm Track direction
- **overThreshold:** Distance for a city or POI to be considered "over" the Storm Track
- **nearThreshold:** Distance for a city or POI to be considered "near" the Storm Track
- **lineofStormsDistance:** The distance in km between the 2 original storm points when Line of Storms is selected.
- **lineofStormsAzimuth:** The azimuth between the 2 original storm points when Line of Storms is selected.
- **areaSource:** Relation to a table in the maps database
- **pointField:** Relation to a field name in a table in the maps database
- **pointFilter:** A filter when making a query to the map database. This can be used to include or exclude points of interest with specific WarnGenLev values.
- **areaNotationField:** Relation to a field name in a table in the maps database
- **areaField:** Relation to a field name in a table in the maps database
- **fipsField:** Relation to a field name in a table in the maps database
- **parentAreaField:** Relation to a field name in a table in the maps database
- **timezoneTable:** Relation to a table in the maps database
- **timezoneField:** Relation to a field name in a table in the maps database
- **areaNotationTranslationFile:** File noting the difference between "county" and "parish"
- **areaConfig**
- **inclusionPercent:** The minimum percentage for a county or zone to be included in a warning
- **inclusionAndOr:** Determines which of the inclusion Area and Percentage are used for inclusion
- **inclusionArea:** The minimum area in square kilometers for a county or zone to be included in a warning
- **pointField:** Relation to a field name in a table in the maps database

- **pointFilter:** A filter when making a query to the map database. This can be used to include or exclude points of interest with specific WarnGenLev values.
- **fipsField:** Relation to a field name in a table in the maps database
- **parentAreaField:** Relation to a field name in a table in the maps database
- **timezoneField:** Relation to a field name in a table in the maps database
- **closestPointsConfig**
- **numberOfPoints:** This configuration sets the number of “Closest Points” which will be used in the first bullet. Any point included in the “Closest Points” variable will be excluded from the “Other Points” variable so that a location is not repeated.
- **pointFilter:** A filter when making a query to the map database. This can be used to include or exclude points of interest with specific WarnGenLev values.
- **geospatialConfig**
- **pointSource:** Relation to a table in the maps database
- **areaSource:** Relation to a table in the maps database
- **parentAreaSource:** Relation to a table in the maps database
- **maskSource:** Relation to a table in the maps database
- **maskFilter:** Filter used to exclude geometry outside of the local CWA
- **basinConfig**
- **maskSource:** Relation to a table in the maps database
- **bulletColumn:** Relation to a field name in a table in the maps database
- **maskFilter:** Filter used to exclude geometry outside of the local CWA

## Values passed to WarnGen \*.vm templates

The following values from the Java WarnGen plugin code are passed to the templates for WarnGen products:

- **vtecOffice** – is the 4 letter id set by the logic in “getSite4LetterId” shown above
- **siteId** – String: The 3 letter WFO set in Localization
- **officeShort** – String set in site localization config.xml
- **officeLoc** – String set in site localization config.xml EX: officeShort: “OMAHA/VALLEY NE” officeLong: “OMAHA”
- **backupSite** – The officeLoc value for a site that WarnGen is backing up via backup mode.
- **localtimezone** – Time Zone where the warning starts in single letter format
- **secondtimezone** – If the Warning spans multiple time zones, the second Time Zone is stored here
- **stormType** – “line” if it is a line of storms, “single” otherwise
- **now** – Date value representing now
- **start** – Date value representing the start time
- **expire** – Date value representing the expire time
- **event** – Date value representing the event time

- **fipsline** – String representing the UGC Header
- **fipslinecan** – String representing a UGC Header
- **areaPoly** – String representing the polygon
- **movementInMph** – double representing movement speed
- **movementDirectionRounded** – double representing movement direction
- **movementDirection** – double representing movement direction
- **movementInKnots** – double representing movement speed
- **action** – String representing the VTEC Action
- **oldvttec** – String representing the old VTEC tracking number for followup products
- **phenomena** – String representing the VTEC pp field
- **mode** – String representing test mode if applicable
- **bullets** – Array of Strings representing the bullets selected in Warngen
- **eventLocation** – array of Point2D values (java.awt.geom.Point2D) – **Do not edit**
- **otherPoints** – array of Strings representing other points impacted by the storm – **Do not edit**
- **pathCast** – PathCast value (com.raytheon.viz.warngen.gis.PathCast) – **Do not edit**
- **closestPoints** – array of ClosestPoints (com.raytheon.viz.warngen.gis.ClosestPoint) – **Do not edit**
- **areas** – array of AffectedAreas (com.raytheon.viz.warngen.gis.AffectedAreas) – **Do not edit**
- **cancelareas** – array of AffectedAreas (com.raytheon.viz.warngen.gis.AffectedAreas) – **Do not edit**
- **timeFormat** – Hashtable containing 6 SimpleDateFormats

Hashtable key	Hashtable Value
header	hhmm a z EEE MMM d yyyy
plain	hhmm a z EEEE
clock	hmm a z
ymdthmz	yyMMdd'T'HHmm'Z'
ddhhmm	ddHHmm
Time	HHmm

- **list** – ListTool (org.apache.velocity.tools.generic.ListTool)

Method use examples:

\$primes is an array of integers containing {2, 3, 5, 7}

\$lists.size(\$primes) -> 4

\$lists.get(\$primes, 2) -> 5

\$lists.set(\$primes, 2, 1) -> (primes[2] becomes 1)

\$lists.get(\$primes, 2) -> 1

\$lists.isEmpty(\$primes) -> false

`$lists.contains($primes, 7) -> true`

- **mathUtil** – WarnGenMathTool (extending `org.apache.velocity.tools.generic.MathTool`)

Methods:

(<http://velocity.apache.org/tools/devel/javadoc/org/apache/velocity/tools/generic/MathTool.html>)

`$mathUtil.roundTo5(num)`  
`$mathUtil.roundToInt(num, multiple)`  
`$mathUtil.abs(num)`  
`$mathUtil.add(num1, num2)`  
`$mathUtil.ceil(num1)`  
`$mathUtil.div(num1, num2)`  
`$mathUtil.floor(num1)`  
`$mathUtil.getAverage(nums)`  
`$mathUtil.getRandom()`  
`$mathUtil.Total(nums)`  
`$mathUtil.matchType(num1, num2)`  
`$mathUtil.max(num1, num2)`  
`$mathUtil.min(num1, num2)`  
`$mathUtil.mod(num1, num2)`  
`$mathUtil.mul(num1, num2)`  
`$mathUtil.pow(num1, num2)`  
`$mathUtil.random(num1, num2)`  
`$mathUtil.round(num1)`  
`$mathUtil.roundTo(decimals, num2)`  
`$mathUtil.sub(num1, num2)`  
`$mathUtil.toDouble(num1)`  
`$mathUtil.toInteger(num1)`  
`$mathUtil.toNumber(num1)`

- **dateUtil** – DateUtil (`com.raytheon.viz.warngen.util.DateUtil`)

Methods

`$dateUtil.format(Date, DateFormat)`  
`$dateUtil.format(Date, DateFormat, Interval)`  
`$dateUtil.format(Date, DateFormat, Interval, TimeZone)`  
`$dateUtil.format(Date, DateFormat, TimeZone)`

Here is a listing for reference of the logic used by WarnGen to create the 4 letter office id. Note that it accounts for the OCONUS sites.

```
/**
 * Converts a 3 letter site ID into a 4 letter ID, e.g. OAX to KOAX
 *
 * @param site3LetterId
 *         the 3 letter site id
 * @return
```

```

*/
public static String getSite4LetterId(String site3LetterId) {
    // this code was ported from legacy GFE
    if (site3LetterId.equals("SJU")) {
        return "TJSJ";
    } else if (site3LetterId.equals("AFG") || site3LetterId.equals("AJK")
        || site3LetterId.equals("HFO") || site3LetterId.equals("GUM")) {
        return "P" + site3LetterId;
    } else if (site3LetterId.equals("AER") || site3LetterId.equals("ALU")) {
        return "PAFC";
    } else {
        return "K" + site3LetterId;
    }
}
}

```

### How to access the data contained in a value in a template.

In general, any String or primitive type method or attribute can be accessed directly within the Warnngen template using the following syntax:

*String or primitive type: `#{myStringValue}`*

In the above context, myStringValue was a String passed directly into the template  
*String method: `#{dateUtil.format(#{pc.time}, #{timeFormat.clock}, #{localtimezone})}`*

In the above context, dateUtil is a com.raytheon.viz.warnngen.util.DateUtil Object which contains the method *format*, which accepts 3 arguments. Additionally, `#{pc.time}` (A date value from the pathCast) is replaced by the time attribute of the pc object, `#{timeFormat.clock}` is replaced by the SimpleDateFormat addressed in the timeFormat Hashtable by “clock,” and `#{localtimezone}` is replaced by the localtimezone object.

### Template Modifications

Plain text in Warnngen templates can be freely edited as needed. When modifying the Velocity Template Language careful attention should be paid to the syntax so that errors are not introduced. Any line which begins with a pound (#) symbol or any variable surrounded by `#{ }` could potentially be dangerous to alter.

### Modifying Bullets in Warnngen

Bullets can be added to warnngen by editing **cave/etc/warnngen/severethunderstorm.cfg**. The `<bullets>` tag contains several existing bullets and additional bullets can be added by following the basic pattern `<bullet bulletName="x" bulletText="y">` in the proper order. After the bullet has been successfully added to severethunderstorm.cfg, the **cave/etc/warnngen/severethunderstorm.vm** file must be modified to take that bullet into account.

When adding a **Call To Action** a statement similar to Example #4 in the following section should be followed. First check if the **list** contains the bulletName using an if statement. Insert the text for the Call to Action on the following line, followed by an empty line. Finally, finish the if statement with a `#end`.

## VTL Examples

The following VTL examples are taken directly from the **severethunderstorm.vml** template:

### Example #1:

```
#if(${mode}=="test" || ${mode}=="practice")
TEST...SEVERE THUNDERSTORM WARNING...TEST
#else
SEVERE THUNDERSTORM WARNING
#end
```

The above example is a typical if statement. If the **mode** is set to *test* or *practice* then “TEST...SEVERE THUNDERSTORM WARNING...TEST” is printed to the SVR product, otherwise “SEVERE THUNDERSTORM WARNING” is printed

### Example #2:

```
${officeLong} HAS ISSUED A
```

The above example will output “THE NATIONAL WEATHER SERVICE IN OMAHA HAS ISSUED A” if **officeLong** is set correctly in `caveData/configuration/site/OAX/com.raytheon.viz.warngen/config.xml`

### Example #3:

```
#if(${list.contains($bullets, "doppler")})
  #if(${stormType} == "line")
    #set ($report = "NATIONAL WEATHER SERVICE DOPPLER
RADAR INDICATED A LINE OF SEVERE THUNDERSTORMS")
  #else
    #set ($report = "NATIONAL WEATHER SERVICE DOPPLER
RADAR INDICATED A SEVERETHUNDERSTORM")
  #end
#end
```

The above example shows a compound control structure. If the bullets list sent from warngen contains the “doppler” bullet (as indicated by the bulletName in **cave/etc/warngen/severethunderstorm.cfg**) the inner if statement is reached. At that point if the **stormType** is **line** then the report text is set to a message indicating a line of storms. Otherwise the report text is set to a message indicating a single storm. Finally, the first #end ends the stormType if statement and the second #end ends the list.contains if statement.

### Example #4:

```
#if(${list.contains($bullets, "torWatchRemainsInEffect")})  
${testMessage}A TORNADO WATCH REMAINS IN EFFECT FOR THE  
WARNED AREA. IF A TORNADO IS SPOTTED... ACT QUICKLY AND  
MOVE TO A PLACE OF SAFETY IN A STURDY STRUCTURE...SUCH AS A  
BASEMENT OR SMALL INTERIOR ROOM.  
  
#end
```

This example is a typical call to action line and should be replicated for any additional Call to Action lines. The line beginning with `#if` determines if the **torWatchRemainsInEffect** bullet was highlighted in Warngen. If it is, the contents of the **testMessage** variable are printed followed by the next three lines of text. The `#if` statement ends at the `#end` line.

### Example #5:

```
#foreach (${city} in ${pc.points})  
#if(${city.roundedDistance} < 3)  
## close enough to not need azran, considered OVER the area  
${city.name}##  
#else  
## needs azran information  
${city.roundedDistance} MILES  
#direction(${city.roundedAzimuth}) OF ${city.name}##  
#end  
#end
```

This example shows a **foreach** loop. **pc.points** is an array of cities affected by the pathcast of the storm. These objects should not be modified, but they can be used and displayed.

The loop covers the first line of the example until the final line for each item in the array **pc.points**. Each item in this array is temporarily renamed to **city** when it goes through the rest of the template. The `#if` statement checks if the **roundedDistance** of the city (which determines how far the city is from the pathcast) is less than 3 miles. If this is true the cities name is printed. Otherwise, the **roundedDistance** is printed to indicate in the warning that the storm is **roundedDistance** miles in **roundedAzimuth** direction of the city. Notice that the line beginning with two pound symbols (**##**) is a comment. These can be used anywhere within a template.

## Dam and Drainage Basin Translator

The dam\_info.txt file from AWIPS I has been replaced by a metadata table named “mapdata.customlocations” in AWIPS II. In order to populate this table the user must run a script located on the EDEX server at:

```
${EDEX_HOME}/bin/daminfotranslator.sh
```

usage: DamInfoTranslator [cwa [rfc]] xxx-dam\_info.txt

- 1 Copy the file XXX-dam\_info.txt to a directory on the Edex server. ~/ for instance.
- 2 Change directories to \$EDEX\_HOME/bin/
- 3 Run the command sh daminfotranslator.sh ~/XXX-dam\_info.txt > ~/XXX-dam\_info.sql
- 4 Run the SQL created in ~/XXX-dam\_info.sql against the Edex database.

### **Afos2Awips Translator**

The afos2awips.txt file from AWIPS I has been replaced by a metadata table named “afos\_to\_awips” in AWIPS II. In order to update this table the user must run a script located on the EDEX server at \$EDEX\_HOME/bin/afos2awipstranslator.sh.

usage: Afos2AwipsTranslator afos2awips.txt

- 1 Copy the file afos2awips.txt to a directory on the Edex server. ~/ for instance.
- 2 Change directories to \$EDEX\_HOME/bin/
- 3 Run the command sh afos2awipstranslator.sh ~/afos2awips.txt > ~/afos2awips.sql
- 4 Run the SQL created in ~/afos2awips.sql against the Edex database.