# AWIPS II warnGen Localization¶ 04/31/2000

Mike Dangelo  
NWS WFO State College, PA (KCTP)  
 michael.dangelo@noaa.gov

This document covers the basics of customizing warnGen in the AWIPS II environment. It is intended for use by the warnGen focal point at WFOs. Occasionally, notes will be provided to describe some similarities and differences between the AWIPS II and legacy AWIPS environments with regard to warnGen localization.

# Contents¶

# Localization Overview

1.1. Purpose of warnGen Localization  
1.2. Major elements of warnGen  
1.2.1. Overarching configuration files  
1.2.1.a. config.xml  
1.2.1.b. VM\_global\_library.vm  
1.2.2. Template configuration files  
1.2.2.1. xml files (XML)  
1.2.2.2. vm files (Velocity Template Language)  
1.2.2.3. Reference documentation for XML, Velocity Template Language, and the Java API

# Localization procedures

2.1. Recommended Editing Practices  
2.1.1. Configuration file locations  
2.1.1.1. "BASE" (baseline)  
2.1.1.2. "SITE" (local)  
2.1.1.3. user ("username" e.g. "mmd")  
2.1.2. backup of files  
2.1.3. Localization Framework Editor  
2.2. Inputs  
2.3. GUI appearance and functionality (config.xml and \*.xml)  
2.4. Template GUI functionality (\*.xml and \*.vm)  
2.5. Bullet interaction (\*.vm)

# Common template change instructions and examples

3.1. Modify the GUI major bullets  
3.2. Add a new template to the GUI  
3.3. Add an “optional” bullet to a specific product/template  
3.4. set an optional bullet to be chosen default  
3.5. Change possible duration values  
3.6. Change zone-based v. FIPS (county)-basis  
3.7. The effect of blank lines in \*.vm files  
3.8. Suggested variable and bullet naming conventions  
3.9. Suppress the second bullet “THIS INCLUDES THE CITIES OF...” output  
3.10. Change or Add map backgrounds that display when a certain template is loaded/selected in the main GUI  
3.11. Utilize localConfigs - to add mile marker references   
3.12. Configure dam break files  
3.13. The #parse and #include commands in vm code. Example: Add warning forecaster names automatically

# 1. Localization Overview¶

## 1.1. Purpose of warnGen Localization¶

The localization (or customization) of warnGen templates is necessary in order for local sites to be able to more-specifically define the description of threats to their local area.

The templates are files which control the layout of the warnGen GUI (Graphical User Interface), and the behavior of the warning templates available to the forecasters via the GUI. The NWS Directives spell out much of the format of the warnings and their companion follow up statements. The text of the statements, and the information contained in the text or body of the statement is usually controlled very little by the Directives. The wording available to the forecasters is the most flexible part of the warnGen statements.

Local offices may wish to add functionality to the templates that allows the forecasters to more-easily choose wording which accurately describes the threat at hand. These changes can be as simple as adding a choice for a specific threat (e.g. Adding selections to the Areal Flood Warning GUI to spell out the specific amount of rainfall that has triggered the warning). This localization may save valuable seconds when generating a warning – since the forecaster will not have to worry about manually typing in the amount (nor having to delete the !“placeholder!” tags).

In general, the files that control the generation of warning statements are grouped together with regard to the relationship each of the products has with the others. For example, the templates that control the generation of parent Severe Thunderstorm Warnings (PIL: SVR) are separate from the templates that control the generation of the parent Tornado Warnings (PIL: TOR). Yet, the Severe Weather Statement (PIL: SVS) is a follow up statement to both the SVR and TOR (Tornado Warning) PILs. Thus, the templates for SVS generation is usually able to generate follow up statements for either a SVR or TOR parent warning.

## 1.2. Major elements of warnGen¶

### 1.2.1 Overarching configuration files¶

Two main files control the major functions and layout of the warnGen GUI, and allow for flexibility in some of the variables used by the templates: “config.xml” and “VM\_global\_library.vm.”

config.xml

The file “config.xml” controls three aspects of the GUI: a) the layout of the where the templates may be found on the GUI, b) the plain name of the issuing office which will be used in the headers of the warnings/statements, and c) which sites the local office may need to issue warnings for in backup mode.

This XML file is currently (5 Aug 2011) found in the: $EDEX\_HOME/data/utility/common\_static/base/warngen directory where $EDEX\_HOME is typically /awips2/edex.

From Max S. at Raytheon as of 5 Aug 2011: You need permission to edit this config.xml file SITE level. Permissions are stored in edex\_static/base/roles/userRoles.xml. You can give yourself permission to edit the file at site or you can copy to user and configure the config.xml for yourself or you can give all users permission to edit at site for that file. There are examples in the userRoles.xml file.

It is highly recommended that sites create a local config.xml file in their $EDEX\_HOME/data/utility/common\_static/site/XXX/warngen directory, where XXX is the three-letter id of the WFO (e.g. LWX or CTP). This will allow the sites to keep the BASE file as is, and make desired modifications to the local file only.

This file was originally kept in a separate directory from most of the templates: /awips2/edex/data/utility/cave\_config/site/CTP/com.raytheon.viz.warngen/config.xml

VM\_global\_library.vm

"VM\_global\_library.vm" is a set of velocity macros. A "macro" is analogous to a "function" in many other programming languages. Usually a set of variables is passed to the macro and output text for a statement is generated based on the code. These macros are used in many of the templates, as the desired output text is common to each type of warning/statement.

I assume that the macros found in "VM\_global\_library.vm" will most likely not be locally configurable unless there are changes made to the Directives, or an error is found. It may be possible to add macros to this file in the future. But, for now, the most logical place for locally developed macros is inside the SITE-level/local template ".vm" file(s).

The "VM\_global\_library.vm" file is found in the same directory as the rest of the BASE WarnGen? templates: (on dx3) $EDEX\_HOME/data/utility/common\_static/base/warngen/

### 1.2.2. Template configuration files¶

Each warning/statement product has two configuration files -- unlike legacy AWIPS, where each warning/statement had only one ("\*.preWWA") file. These are the "xml" and "vm" files. The pair of files must be named exactly the same, with only the suffix different (e.g. "severethunderstorm.vm" is paired with "severethunderstorm.xml)." These files must reside in the same directory. If localized/customized/SITE-level templates are found by the WarnGen? program, it will use those templates to build the GUI instead of the BASEline templates.

The BASE template files are found in: $EDEX\_HOME/data/utility/common\_static/base/warngen/ (/data/fxa/nationalData/ in legacy AWIPS)

SITE (local) template files will be placed in: $EDEX\_HOME/data/utility/common\_static/site/XXX/warngen/ where XXX is the three-letter site ID, e.g. CTP (/data/fxa/customFiles/CTP-... in legacy AWIPS)

### 1.2.2.1. "\*.xml" configuation files (XML (Extensible Markup Language))¶

In general, each warning/statement will have a unique .xml file. A .xml file will always be paired with a like-named .vm file. (The "pairing" is done in the config.xml file - which controls the main warngen GUI)

The .xml files are necessary to set most of the parameters displayed on the WarnGen? GUI. Many of the GUI parameters are configurable with respect to each product. A few of these parameters are set by the Directives, but those may be changed in future Directives.

The .xml files are in an XML format. This format is structured much like a directory tree, as well. The flexibility of XML lies in the fact that new attributes and children may be added as the need arises.

The main variables (children) of the .xml file for any given product (and example values) are:

polygonShape: Default/starting Polygon Shape (1 = regular pathcast, 2 = square)

followups: List of allowable Follow-up VTEC Actions which will consist of one or more of the following list: NEW, CON, COR, EXT, CAN (EXA?, EXB?).

phensigs: List of allowable VTEC Phenomena and Significance codes (e.g. SV.W, TO.W, FA.Y, etc.) for the product

enableRestart: Toggle to enable/disable the "RESTART" button.

durations: List of available "DURATIONS" that will be displayed on the GUI.

bullets: Detailed list of all BULLETs available to the forecasters.

Each <BULLET> tag contains 2 mandatory qualifiers, and up to 4 optional qulifiers/descriptors:

bulletName (mandatory) = value is a "string" passed to the companion vm template file to refer to the specific bullet (e.g. "oneInchHail")

bulletText (mandatory) = value is a "string" of the descriptive text that will be displayed on the GUI (e.g. "SKYWARN SPOTTERS")

bulletType = if specified, the value can be either "title" - usually used as spacer or heading for like/grouped bullets, or "basin" which refers to a "Geometry" in the "customlocations" table. A "title" bullet is not user-selectable.

bulletGroup = if specified, the value is a "string" which groups bullets into categories where only one bullet of each group is allowed to be selected ("radio button" functionality). The bulletGroup value must be exactly the same for each of the bullets in the group. (<VAR | value=radio | var=XXXXXX\_LOCK> used in conjunction with <XXXXXX> label applied to bullets in legacy WarnGen?)

bulletDefault = if the value is positive ("true") that bullet is turned "on" (selected) by default when the product is chosen in the "Product Type" section of the GUI. If "bulletDefault" is not set, the bullet is not "on" by default. (the carat "" in legacy WarnGen?)

parseString = if specified, the value is a "string" in the original product that WarnGen? attempts to match/find when a follow up statement has been chosen in the "Product Type" section of the GUI. If the string is found/matched in the parent/issued product, the bullet is turned "on" as a default. (=STRING TO LOOK FOR= in legacy WarnGen?)

pathcastConfig: contains settings for precision and format of the pathcast

areaConfig: contains settings for the inclusion of an area, and how to describe the warning area/polygon. (some of the info in areaConfig is set in the "wwaConfig.txt" file in legacy AWIPS)

closestPointsConfig: contains settings that control the precision used in naming the closest point to the storm.

geospatialConfig: contains settings that control what areas will be allowed in the warning/statement.

riverBasinConfig (for flooding templates): contains settings that control what river basins will be allowed in the warning/statement.

From Raytheon Doc AWIPS-II Localization Companion 11Aug2010:

Configuration Values in Template .xml Files

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

Note: This is a system-level setting and should only be modified if necessary.

phenomena:

Defines which Phenomena this template can create Followup products for.

Note: This is a system-level setting and should only be modified if necessary.

Significance:

Defines for which Significance this template can create Followup products.

Note: This is a system-level setting and should only be modified if necessary.

Durations:

Available selections (forecaster-selectable) of duration of the warning

DefaultDuration?:

Default duration of the warning (pre-selected upon loading the template)

Bullets:

Defines the forecaster-selectable optional bullets like Call(s) To Action

bulletText: Defines the text displayed in the WarnGen? dialog bulletGroup: Provides a mechanism for limiting selections in WarnGen?. Only one item per bulletGroup can be selected so related items should list the same (radio buttons) bulletType:

bulletType=”title” makes the bullet unable to be selected bulletType=”basin” correlates the bullet to a geometry of the same name in the custom locations table

pathcastConfig:

Enabled “1” will enable the Storm Track, or “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

Note: This is a system-level setting and should only be modified if necessary.

pointField:

Relation to a field name in a table in the maps database

Note: This is a system-level setting and should only be modified if necessary.

areaNotationField:

Relation to a field name in a table in the maps database

Note: This is a system-level setting and should only be modified if necessary.

areaField:

Relation to a field name in a table in the maps database

Note: This is a system-level setting and should only be modified if necessary.

fipsField:

Relation to a field name in a table in the maps database

Note: This is a system-level setting and should only be modified if necessary.

parentAreaField:

Relation to a field name in a table in the maps database

Note: This is a system-level setting and should only be modified if necessary.

timezoneField:

Relation to a field name in a table in the maps database.

Note: This is a system-level setting and should only be modified if necessary.

closestPointsConfig:

numberOfPoints:

This configuration sets the number of “Closest Points” that 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 used when making a query to the map database. This can be used to include or exclude points of interest with specific WarngenLev? values.

geospatialConfig: Note: This is a system-level setting and should only be modified if necessary.

pointSource:

Relation to a table in the maps database

Note: This is a system-level setting and should only be modified if necessary.

areaSource:

Relation to a table in the maps database.

Note: This is a system-level setting and should only be modified if necessary.

parentAreaSource:

Relation to a table in the maps database

Note: This is a system-level setting and should only be modified if necessary.

maskSource:

Relation to a table in the maps database.

Note: This is a system-level setting and should only be modified if necessary.

maskFilter:

Filter used to exclude geometry outside of the local CWA.

Note: This is a system-level setting and should only be modified if necessary.

basinConfig: Note: This is a system-level setting and should only be modified if necessary.

maskSource:

Relation to a table in the maps database

Note: This is a system-level setting and should only be modified if necessary.

bulletColumn:

Relation to a field name in a table in the maps database.

Note: This is a system-level setting and should only be modified if necessary.

maskFilter:

Filter used to exlude geometry outside the local CWA

Note: This is a system-level setting and should only be modified if necessary.

### 1.2.2.2. "\*.vm" template files (VTL (Velocity Template Language))¶

The function of the vm template file is to read input from many sources, especially the forecaster's input from the warnGen GUI, and create the entire text of the warning/statement product.

A .vm file will always be paired with a like-named .xml file.

The logic used in VTL is quite similar to most other scripting languages. See the reference material listed below for syntax.

Some examples:

Comment:

## This line is a comment

Multi-Line Comment:

#\*

This is a

multi-line

comment

\*#

However, a double pound (##) at the end of a line will word-wrap the output of this line to the next line.

FIRST LINE, ##

SECOND LINE.

results in:

FIRST LINE, SECOND LINE.

Set a variable and value:

#set ($reportSource = "TRAINED SPOTTER")

Call a variable:

${reportSource}

A simple if...then statement:

#if (${stormType} == "line")

#set ($newVariable = "now you know it's a line.")

THIS LINE OF STORMS...

#end

### 1.2.2.3. Reference documentation for XML, Velocity Template Language (VTL), and the Java API¶

XML:

 http://www.w3.org/TR/xml/

VTL:

 http://velocity.apache.org/engine/devel/user-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

Java API:

 http://download.oracle.com/javase/1.5.0/docs/api/

# 2. Localization procedures¶

## 2.1. Recommended Editing Practices¶

The recommended editor for all AWIPS II localization changes is to use the "Localization Perspective" editor in CAVE.

### 2.1.1. Configuration file locations (repositories) on dx3¶

### 2.1.1.1. "BASE" (baseline)¶

The warnGen BASE files repository: dx3:/awips2/edex/data/utility/common\_static/base/warngen/ (analogous to legacy AWIPS's /data/fxa/customFiles/ - where localized files were prefixed with LLL-)

### 2.1.1.2. "SITE" (local)¶

The warnGen SITE (LLL) files repository: dx3:/awips2/edex/data/utility/common\_static/site/LLL/warngen/ (analogous to legacy AWIPS's /data/fxa/nationalData/)

This is the place where all your local template files (and the VM\_global\_library.vm) will reside. The changes you may here will not affect the BASEline files. However, care must be taken to not use the "Save As..." option in the Localization Perspective to save your open file into the improper directory.

### 2.1.1.3. user ("nnn" e.g. "mmd")¶

The warnGen user files repository: dx3:/awips2/edex/data/utility/common\_static/user/nnn/warngen/

For the near future, it would probably be a best practice to not set up any user level warnGen templates. This will ensure that all local users will have the same working templates to use. Even small changes can negatively affect the performance of the templates. This is a new option/level of customization that is available to us in AWIPS II. Perhaps at some point, the user level of localization may be useful in testing template and configuration changes.

### 2.1.2. backup of files¶

It is highly recommeded that sites keep backup copies of all localized warnGen files in a separate directory (apart from the SITE, BASE and user) directories. This will aid troubleshooting and serve as a fail-safe to revert back to what was working before any changes were made (either to the templates/config files, or AWIPS itself). As always, the BASE templates should never be modified.

### 2.1.3. Localization Perspective Editor¶

## 2.2. Inputs¶

The Localization Perspective Editor reads and displays the files that are possible to localize. It reads the warnGen files in the directories listed in section 2.1.1.

## 2.3. GUI appearance and functionality (config.xml and \*.xml)¶

To start the Localization Editor: On the CAVE window, look for the small button with the picture of a window and a small yellow plus(+) sign on it. The "tool tip" for this button should say "Open Perspective." It is located immediately to the upper-left of the D2D window, next to the "tabs" (buttons) labeled "GFE" and "D2D." Press the "Open Perspective" button, and choose "Localization." A new tab labeled "Localization" should open next to the "Open Perspective" button (next to the D2D and GFE buttons).

The GUI runs inside CAVE, with a tab/window look that displays in place of/on top of D2D, GFE, or any other perspective (hydro, etc.) you have open in CAVE. The editor displays these files in a directory tree fashion, with the tree to the left of a large blank space. Be aware that this file tree appears somewhat backwards from an actual AWIPS II file system tree. That is, similarly named files appear first on the tree before the repository location - which defines the level of local importance (e.g. D2D > warngen > severethunderstorm.vm > SITE).

In the Localization Editor tab of CAVE, you will see a directory tree with a number of folders on the left side. A large blank tab in the middle of the tab, and a smaller tab to the far right with the header "outline."

Double click (or hit the little arrow next to) the D2D "directory" to expand the tree. Expand the warnGen folder. In the warnGen tree, you should see a number of filenames that look like trees. If you expand those tress, you will see "locations" where files of the same name reside. Each of these locations/entries are name of the localization sub-directory in AWIPS II where that specific file resides. That is, there will most likely be multiple versions of each file - a "BASE"line file, a file in your local office's directory ("SITE(CTP)"), and most likely other files of your backup sites' templates (SITE ("BGM"), SITE("LWX)"), as well as any user-level files ("uuu", "mmd", etc.).

In order to modify a file, you should expand the tree fully, so that you find the appropriate filename and the repository levels (BASE, SITE, user). Double-clicking on the proper version (BASE, SITE, or user) of the file desired will display the file in the main/blank tab of the larger Localization Perspective window/tab. A third tab in the window to the left will display an outline if the file is set up in that form (like the XML .xml files). The .vm files do not have a structure like the .xml files, and therefore, no outline is available.

Again, it is suggested to keep all localization changes to the SITE level for the time being.

You may edit the .vm files using the editor just like a normal text editor. However, the .xml files are more structured, and the Localization Perspective editor is much more ridgid when editing XML. While this rigidity is a positive when wishing to avoid introducing errors by missing a tag, etc, it can be a negative when looking to make wholesale changes.

## 2.4. warnGen Template functionality (\*.xml and \*.vm)¶

### 2.4.1 The template .xml (config) files¶

See section 1.2.2 for details on the structure of the .xml files.

As stated above, the .xml file's variables and the children of each variable set up the visible portion of the GUI.

Most of the variables are set once and changed rarely. The most common change and interaction between the .cfg file contents and the appropriate .vm file happens via the bullets. Each bullet that shows up in the warnGen GUI's "Optional Bullets" list is listed in the <BULLETS> and given a name via the attribute: bulletName.

### 2.4.2 The template .vm (Velocity Macro) files¶

See section 1.2.2 for details on the structure of the .vm files.

Again, the .vm language is the script/programming logic which reads the various input sources, and gives text output to the forecaster.

When the forecaster selects the "Create Text" button, the .vm code is used to create the flat text of work files which show up in the text workstation editor (WRKWG#). Inputs include, but are not limited to: the initial location of the weather feature, the speed of movement, if any, the extrapolated pathcast/track points, the warning area, and appropriate input from the GUI and the .xml file.

## 2.5. Bullet interaction (\*.vm)¶

## 3. Common template change instructions and examples¶

Changes to the config.xml file require a CAVE restart to take effect. Most changes to the .xml and .vm files can be done on the fly. There is no need to run anything like a mainScript, etc, as in legacy AWIPS. Beware, if you have a user-level template file, any changes to the site-level template file of the same name will not have any effect for that specific user. This is one of the main reasons to not create any user-level templates.

## 3.1. Modify the GUI major bullets¶

## 3.2. Add a new template to the GUI¶

Edit the config.xml file: /awips2/edex/utility/cave\_config/site/CTP/com.raytheon.viz.warngen/config.xml

To add a new selection/template to the "Other:" Product Types drop-down list: Inside the <otherWarngenProducts> tags, add a descriptive name for the template which will be displayed on the GUI (ex: Severe Weather Statement). Follow it with a forward slash (/) and the common name of the template files (ex: svs). In order to separate each template listing from the others, use a comma (,) in between them. The final template in your list will not be followed by a comma. ex: <otherWarngenProducts>Severe Weather Statement/SVS,Flash Flood Statement/ffs,New Template Name/new,Other New Template/new2</otherWarngenProducts>

## 3.3. Add an “optional” bullet to a specific product/template¶

Find the appropriate .xml file. Add/Delete? any of the <bullet > entries in the template. All bullets must be between the tags: <bullets> and </bullets>. Assign a unique and appropriate bulletName (and other values) to any new or changed bullets.

In the companion .vm file. Make any changes to have the .vm recognize and act upon the renamed/new bullet. A common query to see if the bullet has been selected is:

#if (${list.contains($bullets, "nameofbullet")})

The quotes are necessary

If you have deleted any bullets from the .xml file, you should probably remove or remark-out (##) references to it in the .vm file to reduce confusion. Beware to not introduce errors in the vm code when deleting anything.

## 3.4. Set an optional bullet to be chosen default¶

Find the appropriate .xml file. Set the desired bullet's "bulletDefault" attribute to "true": bulletDefault="true"

## 3.5. Change possible duration values¶

Find the appropriate .xml file. Edit the .xml file and add the desired <duration>MMM</duration> tag(s). Use the number of minutes in place of "MMM." You may also delete any of the duration values in the .xml file.

## 3.6. Change zone-based v. FIPS (county)-basis¶

While more rare, you may wish to add a new template, and this may come in handy.

## 3.7. The effect of blank lines in \*.vm files¶

A blank line in any part of the .vm file will result in a blank line in the text output.

## 3.8. Suggested variable and bullet naming conventions¶

It is suggested that all Call to Action bullets be named with an ending of "CTA" This allows for code to more-efficiently query the contents of the $bullets array, and place appropriate CTA begin/end markers in the text output.

## 3.9. Suppress the second bullet “THIS INCLUDES THE CITIES OF...” output¶

Remark out all the code which pertains to this feature. Note that the code is nested inside a large if-else-end statement which begins above the top of all of the product text. See example below:

${WMOId} ${vtecOffice} 000000 ${BBBId}

FFW${siteId}

#if(${specialCorText})

${specialCorText}

#else

${ugcline}

#################################### VTEC LINE ###################################

/${productClass}.${action}.${vtecOffice}.FF.W.${etn}.${starttime}-${dateUtil.format(${expire}, ${timeFormat.ymdthmz}, 15)}/

/00000.0.${ic}.000000T0000Z.000000T0000Z.000000T0000Z.OO/

#################################### MND HEADER ###################################

BULLETIN - EAS ACTIVATION REQUESTED

#if(${productClass}=="T")

TEST...FLASH FLOOD WARNING...TEST

#else

FLASH FLOOD WARNING

#end

NATIONAL WEATHER SERVICE ${officeShort}

#backupText(${backupSite})

${dateUtil.format(${now}, ${timeFormat.header}, ${localtimezone})}

#if(${productClass}=="T")

...THIS MESSAGE IS FOR TEST PURPOSES ONLY...

#end

#headlineext(${officeLoc}, ${backupSite}, ${extend})

#################################

######## FIRST BULLET ###########

#################################

\* ##

#if(${productClass}=="T")

THIS IS A TEST MESSAGE. ##

#end

FLASH FLOOD WARNING FOR...

#if (${hycType} != "")

<L> ${hycType}</L>

#end

#foreach (${area} in ${areas})

##

#if(${area.partOfArea})

#areaFormat(${area.partOfArea} true false) ##

#end

${area.name} ${area.areaNotation} IN #areaFormat(${area.partOfParentRegion} true false) ${area.parentRegion}...

############## remmed out list of cities in first bullet -Mike D #########################

## #if(${list.size($area.points)} > 0)

## #if(${list.size($area.points)} > 1)

## THIS INCLUDES THE CITIES OF... #foreach (${city} in ${area.points})${city}... #end

##

## #else

## THIS INCLUDES THE CITY OF ${list.get(${area.points},0)}

## #end

## #end

#end

#################################

####### SECOND BULLET ###########

#################################

## 3.10 Change or Add map backgrounds that display when a certain template is selected on the GUI¶

Find the appropriate .xml file. Add or substitute the desired map-name as a <map>-tagged entry inside the <maps> tags near the top of the template. The list of available map-names is the same as the name of the Maps overlay available in D2D. So, if the map name on the "Maps" menu is "Rivers," add: <map>Rivers</map> to the template inside the <maps> tags.

## 3.11. Utilize localConfigs - to add mile marker references¶

The local offices may use local reference points not included in a nationally-available database in their warnings. The most common use of this type of localConfig will most likely be interstate/road mile-marker information.

In order to have the points referenced in the warnings, a postgres database table must first be created to hold the points of reference. Raytheon has developed a script to make the migration of interstate/road mile-marker information easier. The script is named "ImportMarkersInfo?.sh" usually found in the "/awips2/database/sqlScripts/share/sql/maps/" directory. The script will use an already existing ".id" file (from legacy AWIPS) as input, in order to make the transfer easiest.

xml part of the code used to make the "i495" mile markers available for use in a specific template:

<!-- mile marker (and other local configurations/points) info

each <localConfig> refers to a unique table in postgres (ex. i495)

therefore, these <localConfig>s lend themselves to be include files full of the mile marker info

pointSource: name of the table in psql

pointField: field of the appropriate table

variable: name of the array referenced in the vm code

maxCount: the most points it will process

pointFilter: which points to consider

sortBy: type of sort to use - mainly "name" which seems to be alpha-numeric -->

<localConfigs>

<!-- Example#1 - output is an alphanumeric listing of the names of valid i495 points -->

<localConfig>

<pointSource>i495</pointSource>

<pointField>name</pointField>

<variable>i495\_alpha</variable>

<maxCount>50</maxCount>

<pointFilter>

<key>WARNGENLEV</key>

<value>0</value>

<type>EXCLUSIVE</type>

</pointFilter>

<sortBy>

<sort>name</sort>

</sortBy>

</localConfig>

<!-- example 2 - output will be numeric place (id) of valid any i495 points -->

<localConfig>

<pointSource>i495</pointSource>

<pointField>id</pointField>

<variable>eyeD</variable>

<maxCount>50</maxCount>

<pointFilter>

<key>WARNGENLEV</key>

<value>0</value>

<type>EXCLUSIVE</type>

</pointFilter>

<sortBy>

<sort>name</sort>

</sortBy>

</localConfig>

<!-- example#3 - output will be un-ordered (raw output) list of the gid (first field) of the valid i495 points -->

<localConfig>

<pointSource>i495</pointSource>

<pointField>gid</pointField>

<variable>raw</variable>

<maxCount>50</maxCount>

<pointFilter>

<key>WARNGENLEV</key>

<value>0</value>

<type>EXCLUSIVE</type>

</pointFilter>

</localConfig>

<!-- end of examples -->

</localConfigs>

vm part of the code that pulls data from the appropriate pointSource (in our example this is "i495") table, and places it into the text of the product

## example of a for-each loop which will output a line-by-line list of interstate/mile marker information - using the "i495\_alpha" variable defined in the companion xml file:

## in this loop, the array "i495\_alpha" has been passed to the vm. The array has a single attribute of "name".

#foreach (${example1} in ${i495\_alpha})

${example1.name}...

#end

This may not be correct, but the "name" attribute seems to be the only thing held in the array/variable passed to the vm code.

It appears we may be able to sort the array further using common java and vm methods. This functionality has not yet been tested.

## example of a for-each loop which will output a line-by-line list of the id number for the interstate/mile marker information - using the "eyeD" variable defined in the companion xml file:

## in this loop, the array "eyeD" has an attribute of "name" as well.

#foreach (${example2} in ${eyeD})

${example2.name}...

#end

## 3.12. Configure dam break files¶

See Phil Kurimski's instructions: SiteSpecificDams

## 3.13. The #parse and #include commands in vm code. Example: Add warning forecaster names automatically¶

It is possible to #include a file in the vm code. The file you wish to include or parse must be located in the BASE directory (This location may change or more locations may be added in the near future)

You are able to make one common file to be included into multiple templates, and keep individual templates much shorter. This command will include the code for parsing (#parse), and/or flat contents (#include) in the output.

Using the #parse command will try to parse the file contents as vm code:

#parse("filename.vm")

So, whatever you have in the file you specify will be processed exactly as vm code, just as if you had cut the code from the template file.

If I just want to drop in some pre-defined text, I can write a flat file with the desired contents, and "include" the file. For example, if I write the forecaster's name/id into a certain file (forecasterName.txt), the following code will only place the forecaster's name in that exact place in the output text:

#include("forecasterName.txt")

The following content has been taken from AWIPS-II Localization Companion Doc (from Raytheon) dated 11 Aug 2010:¶

Warngen Templates¶

Warngen templates can be used on a user, site or base level. Base templates should never be edited, and are stored on the EDEX localization server in the following directory: $EDEX\_HOME/data/utility/common\_static/base/warngen Warngen templates are created using velocity template language (VTL) with supporting configurations in an xml formatted file with a .xml extention. More information on this can be found:

**Site level templates and product configuration files** can be stored in the following location on the EDEX localization server: $EDEX\_HOME/data/utility/common\_static/site/XXX/warngen Where XXX is a site's localization ID. Site specific templates should also be named in the following manner, substituting the site ID for XXX: <templateName>\_XXX For example: dambreak\_CTP NOTE: If custom templates are not loaded from the EDEX server, try creating ~/caveData/common/site/XXX/warngen and restarting CAVE.

**Values Passed to WarnGen? \*.vm Templates** The following values from the Java WarnGen? plug-in code are passed to the templates for WarnGen? products. start: Date value representing the start time. expire: Date value representing the expiration 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 oldvtec: 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): NOTE: DO NOT EDIT otherPoints: Array of Strings representing other points impacted by the storm NOTE: DO NOT EDIT pathCast: PathCast? value (com.raytheon.viz.warngen.gis.PathCast?) NOTE: DO NOT EDIT closestPoints: Array of ClosestPoints? (com.raytheon.viz.warngen.gis.ClosestPoint?) NOTE: DO NOT EDIT areas: Array of AffectedAreas? (com.raytheon.viz.warngen.gis.AffectedAreas?): NOTE: DO NOT EDIT cancelareas: Array of AffectedAreas? (com.raytheon.viz.warngen.gis.AffectedAreas?): NOTE: DO NOT EDIT TimeFormat?: Hastable 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: HHmm 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

(extends org.apache.velocity.tools.generic.MathTool).

Methods:

(http://velocity.apache.org/tools/devel/javadoc/org/apa

che/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 FateUtil (com.raytheon.viz.warngen.util.DateUtil).

Methods:

??

$dateUtil.format(Date,

$dateUtil.format(Date,

$dateUtil.format(Date, TimeZone)

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

DateFormat, Interval,

DateFormat, TimeZone)

??

**How to Access the Data Contained in a Value Inside A Template** In general, any String, primitive type method or attribute can be accessed directly within the WarnGen? 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.warngen.util.DateUtil? Object that contains the method format, which accepts 3 arguments. ${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,” ${localtimezone} is replaced by the localtimezone object.

**Modifying / Editing Templates** Plain text in WarnGen? templates can be freely edited as needed. When modifying the VTL, careful attention should be paid to the syntax so that errors are not introduced. NOTE: Any line that begins with a pound (#) symbol or any variable surrounded by ${} could potentially be dangerous to alter.

**To Add Bullets:** Bullets can be added to WarnGen? by editing or creating a site or user level .xml file. Look at the <bullets> tag in an existing template and note it 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 .xml file, the associated .vm file must be modified to take that bullet into account.

**Call To Actions** When adding a Call To Action a statement, Example #4 below should be followed. First, check to see if the list contains the bulletName is 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.

**Examples** The following VTL examples are taken directly from the severethunderstorm.vm 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 the 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 severethunderstorm.xml) the inner if statement is reached. At that point if the stormType is equal to 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 completes the stormType if statement, and the second #end completes 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 is 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 city’s 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. Note: Lines beginning with two pound symbols (##) are comments. These should begin the line that you wish to comment-out. If used anywhere within a template, the "##" causes a text-wrapping of the output text.