'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 local configuration will most likely be interstate/road mile-marker information.'

Evan Bookbinder – WFO EAX & Mike Dangelo – WFO CTP

This document can also be found at:

<https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/AddingMileMarkers>

**Executive Summary:**

* Import your mile marker id file info into the mapdata database.
* Copy the mileMarkers.xml file into SITE and edit.
* "include" the mileMarkers.xml file in every xml template file.
* Copy the mileMarkers.vm file into SITE and edit.
* "#parse" the mileMarkers.vm file inside every vm template file.

**Details:**

**Import Mile Marker Info from AWIPS 1**

In order to have the mile marker 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 uses an already existing ".id" file (from legacy AWIPS) as input. The ".id" file is from the Warngen GELT, located on the LX box in directory /aiwps/fxa/data/localizationDataSets/LLL (where LLL is the local site ID).

*To run the script(s) below, the password for the Postgres "awips" user account is required.*

You may have your local mile marker "id" files saved off to the side somewhere. This will make the process a little easier. You would just need to copy those files to somewhere where AWIPS II can read them. Just copying them to the /tmp directory on AWIPS II's dx1 will be enough. The file(s) need not be in /tmp, this just seems like a good place to put it/them for a short time. The

The following is an example for running the script. In this case, WFO LWX is importing the I-495 mile markers from a file named "i495mm.id"):

1) log onto AWIPS II using your "awips" user account

2) ssh dx1

3) scp -p lx1:/awips/fxa/data/localizationDataSets/LWX/i495mm.id /tmp

4) cd /awips2/database/sqlScripts/share/sql/maps

5) ./importMarkersInfo.sh /tmp/i495mm.id mapdata i495mm

usage: ./importMarkersInfo.sh <full filepath/filename> mapdata <new table name>

Step 5 takes the GELT id file from "/tmp/i495mm.id" and using the Postgres schema "mapdata," creates (or overwrites) the database table "i495mm."

Likewise in CTP's area, we also reference larger US routes and State roads. For US Route 30 (from file "route30mm.id" into the database table "rte30"):

./importMarkersInfo.sh /tmp/route30mm.id mapdata rte30

This second example also shows that the "id" filename and database table name need not be the same.

**Add Mile Markers to AWIPS 2**

Following the first example, once the mile marker/road information has been imported into an AWIPS 2 database table, it can now be utilized by [WarnGen](https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/WarnGen). Note, if you find you need to make edits to the "id" files, simply re-run the import script(s) above to update the AWIPS 2 database table.

(Jan 27 2012 version of templates or newer required)

You can then reference this database table in an xml file, and pull data out of it using vm code. Therefore you will need to edit both some XML entries in each template, and add some code in the vm part of the template. However, you would need to make the same edits to each of your templates if you wished to use the mile markers in all of them. These operations can be handled more efficiently by using some files which will be included and/or parsed (read more about #parse: [ParseCommand](https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/ParseCommand)) by the XML and vm. This allows you to maintain just two files instead of tweaking every template. These BASE files which have examples in them are mileMarkers.xml and mileMarkers.vm.

You will need to edit the mileMarkers.xml and mileMarkers.vm files to put in information specific to your mile marker database tables.

Two Warngen objects must be instantiated for each mile marker table (one for the markers themselves and one for the sequential IDs that accompany each entry). To do this, one will need to edit the mileMarkers.xml file 'via the localization perspective in CAVE.' New sites may have to copy the "BASE" version to "SITE" to maintain the integrity of the file. An example for an interstate is provided in the templates. Follow the instructions commented into the template file.

An example edit/addition to "mileMarkers.xml" is provided here.

<pointSource variable="i435mm">

<pointSource>i435mm</pointSource>

<pointField>NAME</pointField>

<searchMethod>POINTS</searchMethod>

<withinPolygon>true</withinPolygon>

<maxResults>1000</maxResults>

<distanceThreshold>100</distanceThreshold>

</pointSource>

<pointSource variable="i435mmid">

<pointSource>i435mm</pointSource>

<pointField>GID</pointField>

<searchMethod>POINTS</searchMethod>

<withinPolygon>true</withinPolygon>

<maxResults>1000</maxResults>

<distanceThreshold>100</distanceThreshold>

</pointSource>

The above example sets up two variables: i435mm and i435mmid. These names are totally arbitrary, but a consistent naming convention is suggested. You will reference these variables in the "mileMarkers.vm" code (see immediately below).

After you have put all your mile marker table info into "mileMarkers.xml," you need to add the XML include syntax to the XML file of every template in which you wish to put mile marker info: <include file="mileMarkers.xml"> The above line is already there in the BASE template files, but may be commented out.

Next, you will need to edit the mileMarkers.vm file that will control the actual text output in your products. Similarly, to do this, one will need to edit the mileMarkers.vm file via the 'localization perspective in CAVE.' New sites may have to copy the "BASE" version to "SITE" to maintain the integrity of the file. An example for an interstate is provided. Simply follow the instructions included in the file.

In the vm file of each template you wish to include mile marker info, you need to add the vm #parse syntax: #parse("mileMarkers.vm")

(read more about #parse: [ParseCommand](https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/ParseCommand))

mileMarkers.vm file runs through a loop of all the mile marker info to pick out just the locations inside the box. This line is already there in the BASE template files, but is initially commented out. In the big loop in this mileMarker.vm file, there are calls to a vm macro (mmarkers) which is located in the VM\_global\_library. There are a few arguments to mmarkers which define how the output (if any) will fall out in the text. You will need to build up three arrays in mileMarkers.vm. EXAMPLE:

#set ($databaseName = [${i435mm},${i70momm},${i35momm},${rte30}])

#set ($databaseId = [${i435mmid},${i70mommid},${i35mommid},${rte33id}])

#set ($specificName = ['INTERSTATE 435','INTERSTATE 70','INTERSTATE 35','U.S. ROUTE 30'])

In the example above, we have three Interstates (495, 70 and 35). You need to pair the text name (specificName) with the proper databaseName and databaseId for each mile marker table. If you have 18 Interstates/roads, you will need 18 entries in each of these three arrays.

Your mile marker info can then be imported by uncommenting the mileMarkers.xml and mileMarkers.vm lines in each Warngen template's xml and vm file respectively.

You can (alternatively) make calls to the "mmarkers" macro straight from your vm templates, but this may be a little advanced for those who are new to the XML/vm template language and interaction. The mileMarkers.xml and .vm files are there to help make it more simple.

**Summary:**

* Import your mile marker id file info into the mapdata database.
* Copy the mileMarkers.xml file into SITE and edit.
* "include" the mileMarkers.xml file in every xml template file.
* Copy the mileMarkers.vm file into SITE and edit.
* "#parse" the mileMarkers.vm file inside every vm template file.

go back to: [WarnGen](https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/WarnGen) (main) go to: [WarnGenLocalizationTutorial](https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/WarnGenLocalizationTutorial)

(read more about #parse: [ParseCommand](https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/ParseCommand))

related info: A second Raytheon script exists to import local point information (marine sites/cities/locations/landmarks) for inclusion in products. The script uses an already existing local cities text file (from legacy AWIPS) as input. The script currently requires that only one blank space separate every field in the local cities file. DR 10697 requests that the script be made more friendly to allow multiple spaces in the field separators. To run the script, the password for the Postgres "awips" user account is required.

The following are instructions for running the script (in this case, WFO LWX is importing the local cities file):

1) log on using your "awips" user account 2) ssh dx1 3) scp -p /data/fxa/customFiles/LocalCitiesInfo.txt /tmp 4) cd /awips2/database/sqlScripts/share/sql/maps 5) ./importPointsInfo.sh /tmp/LocalCitiesInfo.txt mapdata localcities

Step 5 here takes the local cities file from dx1:/tmp/LocalCitiesInfo.txt and using the Postgres schema mapdata, creates (or appends to) table localcities.

go back to: [WarnGen](https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/WarnGen) (main) go to: [WarnGenLocalizationTutorial](https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/WarnGenLocalizationTutorial)

(read more about #parse: [ParseCommand](https://collaborate.nws.noaa.gov/trac/siteconfig/wiki/ParseCommand))