



Netism Map Extreme 2.2.2

Complete User Guide

Netism Solutions Inc.

<http://netismsolutions.com> | map@netismsolutions.com

Table of Content

- Terminology
- Requirements
- Architecture Overview
- Map Settings
- Map Locations
- Custom Queries
- Batch Geocoding
- Custom Templates
- Search Form
- License
- Permission

Document Change Log

10/17/2009 Created

Terminology

Latitude - Distance in degrees north or south of the equator. Ex: 40.68525.

Longitude - Distance in degrees east or west from Greenwich, England. Ex: -74.00331.

Geocode – A pair of latitude and longitude.

Geocoding - The process of assigning a geocode to a geographic feature.

Map Extreme (ME) – All license levels of the Netism Map Extreme module.

Module – An instance of the Netism Map Extreme module presented on a page.

Map – The map display from the Netism Map Extreme module.

Location – A record with/without address/geocode information, which is intended to be shown on a map. Ex: a portal, group, person, store, office, product, article, event and photo etc.

Pin – An image or an icon displayed on a map to represent a location.

Requirements

ME supports DNN 4.x/5.x, IIS 6+, ASP.NET 2.0+ and SQL 2000+. It runs completely independent from Netism Solutions the company and the website itself. It's processed completely within your own server and the visitor web browsers. It uses the Bing Map, or formerly Microsoft Virtual Earth, for rendering map tiles, navigation, directions, geocoding and pins display.

Architecture Overview

Data Integrations

You can use the ME add/edit location form, or define queries to select data from external data sources such as, Oracle, MySQL, ODBC, MS SQL and OleDB (Excel, Access, CSV and TSV etc). You can define multiple queries for a single module and match them by URL parameters. Data returned from external sources can also be templated and searched. Bing Map Collection, GeoRSS Feed and KML documents can also be loaded on the map as external data sources.

Module Extensions

ME can be used to extend all other DNN modules with its mapping, listings, geocoding, templating and searching functionalities. You can easily attach location/geocode information to data from any DNN modules in the same database as the ME, such as:

DNN Core Objects: Tabs, Portals, Users and Roles.

DNN Core Modules: Events, Blogs, Repository, Forums, HTML, Document, Feedback, Announcement, and Forms and List (UDF) etc.

Commercial Modules: Events Calendar, XMod, Business Directory, Dynamic Form, Active Social/Forum, Smarter-Thinker modules, CataLook Store, Property Agent, Engage Publish, DNN Articles, Media Gallery and Video Gallery etc.

Display Customizations

All display style and layout are template driven and can be customized. The pin, pin popup, search form, fields, location item list, detail and even the overall module layout.

Search Configurations

The search form, the fields and the field values are totally customizable. Single line location input with auto location suggestions and distance filter. A location can be a full address, just a zip code, just a city name, just a state name, just a country name, or even the name of an attraction, or a point of interest.

A search field can be configured to:

Filter on any data column.

Populate a list of values from database or a predefined set of values.

Preselect a default value from a URL parameter or a static value.

Show the values in a dropdown menu, a radio button, a checkbox, a link, a list of radio buttons, a list of checkboxes or a list of links.

Map Setting

The map preview on map setting mainly provides a quick preview of the default map display region. It does not always reflect the final map display.

Map Settings

Latitude:

Longitude:

Zoom:

If any of the Latitude, Longitude and Zoom field is blank, the map would automatically display all the locations. For birdseyes view, it's best to leave it blank.

Language:
Not all languages are supported in all countries. Local languages might not be available on the street level.

Control:

Style:

Map Size: Width: px
Height: px

Bing Map Collection CID:

GeoRss Url Import:

KML Url Import:

List Page Size:

Use Custom Query: Yes No

Use Custom Template: Yes No

Use Visitor Location: (GeoIP Targeting) Yes No

Reference jQuery: (mostly for DNN 4.x) Yes No

[Update](#) | [Cancel](#) | [Exit](#)

Not all the settings would be reflected on the below map. Always check the public view.

Latitude, Longitude and Zoom

If any of these fields is blank, then the map would automatically show the best view to display all the locations. For birdseye view, it's best to leave it blank. If "Use visitor location" is enabled, then it will override the settings here. The values for the 3 fields here will automatically be update as you drag around or zoom in/out the map.

Language

21 localized languages within the map display. Not all languages are supported in all countries. Even for the supported countries, the localized languages might not be available down to the city or street levels.

If it's set to "DNN Page", then it will follow the DNN localization setting, or use the map default if the DNN localization does not match one of the below 21 localized languages.

Czech – Czech Republic (cs-cz)
Danish – Denmark (da-dk)
German – Germany (de-de)
English – Australia (en-au)
English – Canada (en-ca)
English – India (en-in)
English – United Kingdom (en-gb)
English – United States (en-us)
Spanish – Mexico (es-mx)
Spanish – Spain (es-es)
Spanish – United States (es-us)

Finnish –Finland (fi-fi)
French – Canada (fr-ca)
French – France (fr-fr)
Italian – Italy (it-it)
Japanese – Japan (ja-jp)
Dutch – Netherlands (nl-nl)
Norwegian – Norway (nb-no)
Portuguese – Brazil (pt-br)
Portuguese – Portugal (pt-pt)
Swedish – Sweden (sv-se)

Control

The size of the map navigation control for map panning and zooming:
None, normal, small or tiny.

Style

The style of the map tiles: Road, Shaded, Aerial, Hybrid, Birdeye or BirdeyeHybrid.

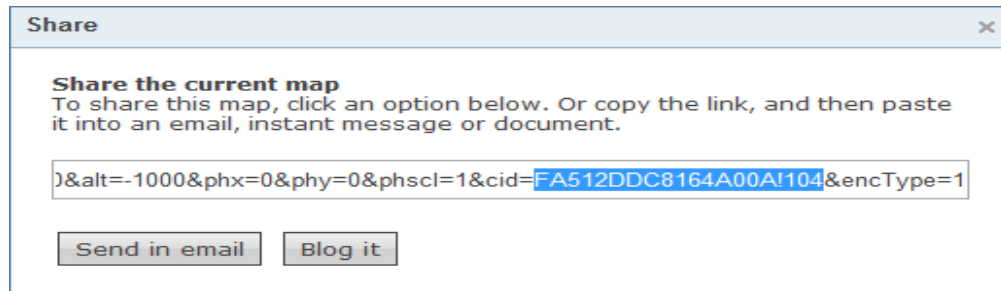
Map Size (Width & Height)

The width and height in pixel unit of the map display.

Bing Map Collection CID

Custom paths (polylines) and areas (polygons) saved from <http://bing.com/map/>.

1. Go to <http://bing.com/map/> and login with a Live ID.
2. Click on "collections", then "Open your collections."
3. Select a collection, or create one. Draw paths and areas, and then save it.
4. Click "share", then locate the CID parameter on the URL.



5. Copy the CID value, then set it in ME setting.



GeoRss Url Import

URL to the GeoRSS feed. It must be on the same domain. The server must be configured to allow the file extension of the feed.

KML Url Import

URL to the KML feed. It must be publicly accessible.

List Page Size

The page size of the location list.

Use Custom Template

Set the module to use the custom templates.

Use Custom Query

Set the module to evaluate and execute custom queries.

To debug, adding debug=query on the URL, which would display the executing query.

Use Visitor Location

Set the module to try to match the visitor location by the visitor IP address. It would then be used for the default map view and searching location.

- Download the Free GeoLite City database file:
 1. Go to <http://www.maxmind.com/app/geolitecity>.
 2. Click on the link "Download the latest GeoLite City Binary Format".
 3. Extract GeoLiteCity.dat (27 MB) from GeoLiteCity.dat.gz (18 MB).
 4. Upload GeoLiteCity.dat to the ME module folder.
Ex: C:\inetsrv\wwwroot\DesktopModules\NetismMapExt\GeoLiteCity.dat
 5. Uploading the file via FTP is recommended. To upload via the DNN FileManager, you need to allow .dat extension under host setting and watch out for page timeout error if you don't have a really fast upload speed.
- Note that localhost or internal/private IP addresses would not work.
- To simulate a visitor IP, add `_ip=<ip address>` on the URL. Try:
 - `_ip=74.208.185.13` for Wayne, PA
 - `_ip=208.75.255.146` for Fort Worth, TX
 - `_ip=74.125.45.100` for Mountain View, CA
- To debug, add `debug=geoip` on the URL.
- It offers over 99.5% accuracy on country level and 79% on city level for the US within 25 mile radius. More info: http://www.maxmind.com/app/geolite_city_accuracy

Reference jQuery

ME uses jQuery for certain functions. It's usually already referenced in DNN 5.x, but not in DNN 4.x. So unless jQuery is already referenced somehow, check "yes". If you are not sure, if you see that all function works properly, then don't worry about it, otherwise check "yes" here and see again.

Map Locations

If the module is set to use custom queries, then this section does not apply.

Edit Location
▼

Title:

Link:

User Name:

Display Order:

Active: Yes No

Address:


City:

State/Region:

Zip/Postal:

Country:

[\[Plot Address on the Map\]](#)



Latitude:

Longitude:

Zoom:

Icon:

Field 1:

Field 2:

Field 3:

Field 4:

Field 5:

Description:

Basic Text Box
 Rich Text Editor

Source

B I U abc x₂ x²

Font Size

This is some description ... This is some description ... This is some description ... This is some description ... This is some description ... This is some description ...

[Show custom editor options](#) | [Refresh Editor](#)

[Update](#) | [Cancel](#) | [Delete](#)

You can only manipulate locations via ME when the below 2 conditions are satisfied:


1. The module is not set to use custom queries.
2. The user is a portal admin, super (host) user or in the Edit Location Role.

Add location

Two ways to start:

1. Shift+ click on the map to add.
2. "Add Location" below the map, bottom of the container, or on the action menu.

Edit location

Click on the  icon to edit the location.

Delete location

Click on the Delete link below the edit location page.

Title – The title (or name) of the location, included in the pin popup.

Link – The URL to be linked for the title.

User Name – The name of the signed in user.

Display Order – The display order on the location list. Smallest number would make the location on the top, largest number would make it the last. Default is 0, it can be negative.

Active – Setting it “yes” would make it appear for public and under view mode of the page. Setting it “no” would only show it under edit mode of the page.

Address, City, State/Region, Zip/Postal, Country – Location information.

Latitude, Longitude, Zoom – Geocode of the location

Plot Address on the Map – Filling the location information then click this would plot the location on the map and fill in the geocode automatically. The geocode and zoom level will also be automatically updated as you drag around or zoom in/out the map.

Icon – The URL of the custom icon image for the location to be used for the pin.

Field 1, Field 2, Field 3, Field 4, Field 5 – Custom fields, up to 50 characters.

Description – Descriptions of the location. Unlimited characters length.

Custom Queries

You can define multiple queries for each module and match them by URL parameters when the page is loaded each time. You can also define queries to run on external data sources. Make sure to set “Use Custom Query” to “Yes” under Map Setting.

Match Order

The order in which the parameters of the query will be matched against the querystring parameters on the URL. If the same parameter is defined for more than one query, then the query of the first matched parameters will be executed.

Data Provider

- DNN Default – The database the hosting DNN is using.
- MS SQL – Microsoft SQL Server.
- MySQL – MySQL Server.
- OLE DB – Excel, Access, CSV and TSV etc.
- ODBC – Older generations of data engine drivers.
- ORACLE – Oracle databases.

Connection String

The connection string to connect to the external data provider, blank for DNN default.

Match Parameters

URL querystring parameter names to match, multiple names are separated by commas. Ex: “country, state, city” or just “zip” etc. There are 2 special parameters:

1. Blank - default/fallback query. It is used when there’s no match on all queries.
2. BatchGeocode – by entering “BatchGeocode” for the match parameter on a query, it will be used to pull address information for the batch geocoding form.

Executing Query

ME first do a security check, then replace all tokens, then execute it.

Tokens: {PortalId}, {TabId}, {UserId}, {ModuleId} and {url-param-name}.

If you have a URL with ?zip=12345, then you can reference it in the query like:

SELECT ... WHERE zip = '{zip}', which will be evaluated to:

SELECT ... WHERE zip = '12345' before it’s executed.

For security reasons, queries with comments or any of the below list of keywords would be denied: *drop, truncate, delete, insert, update, set, exec, execute, create and alter*.

Furthermore, values retrieved from the querystring will always be escaped by ME, so you should always put ' and ' around the token like zip = '{zip}' instead of just zip = {zip}.

If { or } is intended to be a literal character, you can escape { by {{ and } by }}. You can put debug=query on the URL to see the final executing query.

To use custom query with default templates

Query must return below fields for the default templates like from the Netism_MapExtreme_Locations table:

LocationID, Title, Link, Description, Address, City, State, Zip, Country, Latitude, Longitude

To use custom query with custom templates

Custom query must return at least *Latitude, Longitude and LocationID* for proper map display. Query fields can be used in custom templates. If you have a query "select count(*) as total, city ...", then you can use {total} and {city} in custom templates.

Query Builder

Many types of records can be associated with a geographical location, such as a video, a blog, a news article, an event and a user etc. Many modules handle them very well. However, most of them don't offer a function to even just simply display them on a map, let alone support of templating the map view and searching by distance etc.

Query Builder offers a quick and easy way to create a query that retrieves data from certain modules for certain portals or module instances, and attach location and geocode fields to them. It is mainly used to build queries for the Batch Geocoding form. It only works on the same DNN database that is running SQL 2005 or up.

It selects data from a table that is used by another module, and then left joining it to one of the ME tables on the table name and the primary keys. The ME table Netism_MapExtreme_AttachedLocations would then contains address and geocode fields for the records from the other modules.

Once a query is defined, you can use it on the Batch Geocoding form to enter the address, and it would geocode all of them with a single click. You will then be able to use the same query to retrieve records and geocodes to accurately display them on a map.

Example – to attach location information to a Repository Module

Start by clicking on the “Start Wizard!” button. The wizard loads all tables from the DNN database. In this case, the Repository Items, which is the grmRepositoryObjects table.

1. Module Table: --- 121 tables, please select one --- <input checked="" type="checkbox"/> Tables with a ModuleId or PortalId column only	3. Portals: --- 5 portals, please select one ---
2. Key Columns:	4. Tabs: --- select a portal ---
	5. Modules: --- select a tab ---

Selecting the table would load all columns with the Primary Key and Identity columns selected for the Key Columns. The table name (grmRepositoryObjects) and the key columns (ItemID) would be used for joining condition to join the Netism_MapExtreme_AttachedLocations table.

1. Module Table: grmRepositoryObjects <input checked="" type="checkbox"/> Tables with a ModuleId or PortalId column only	3. Portals: --- 5 portals, please select one ---
2. Key Columns: <input checked="" type="checkbox"/> ItemID <input type="checkbox"/> Image <input type="checkbox"/> Approved <input type="checkbox"/> ModuleID <input type="checkbox"/> Author <input type="checkbox"/> Name <input type="checkbox"/> AuthorEmail <input type="checkbox"/> PreviewImage <input type="checkbox"/> Clicks <input type="checkbox"/> RatingAverage <input type="checkbox"/> CreatedByUser <input type="checkbox"/> RatingTotal	4. Tabs: --- select a portal --- 5. Modules: --- select a tab ---

Select a portal from the Portals dropdown, in this case “My personal site.”
 All tabs on the selected portal will be loaded.

1. Module Table: grmRepositoryObjects <input checked="" type="checkbox"/> Tables with a ModuleId or PortalId column only	3. Portals: My personal site (id=0)
2. Key Columns: <input checked="" type="checkbox"/> ItemID <input type="checkbox"/> Image <input type="checkbox"/> Approved <input type="checkbox"/> ModuleID <input type="checkbox"/> Author <input type="checkbox"/> Name <input type="checkbox"/> AuthorEmail <input type="checkbox"/> PreviewImage <input type="checkbox"/> Clicks <input type="checkbox"/> RatingAverage <input type="checkbox"/> CreatedByUser <input type="checkbox"/> RatingTotal	4. Tabs: --- 40 tabs, please select one --- 5. Modules: --- select a tab ---

Select a tab (or page) from the Tabs dropdown, in this case “data e.”
All modules on the selected tab will be loaded.

1. Module Table: grmRepositoryObjects <input checked="" type="checkbox"/> Tables with a ModuleId or PortalId column only	3. Portals: My personal site (id=0)
2. Key Columns: <input checked="" type="checkbox"/> ItemID <input type="checkbox"/> Image <input type="checkbox"/> Approved <input type="checkbox"/> ModuleID <input type="checkbox"/> Author <input type="checkbox"/> Name <input type="checkbox"/> AuthorEmail <input type="checkbox"/> PreviewImage <input type="checkbox"/> Clicks <input type="checkbox"/> RatingAverage <input type="checkbox"/> CreatedByUser <input type="checkbox"/> RatingTotal	4. Tabs: data e (id=134) 5. Modules: --- 2 modules, please select one ---

Select a module from the Modules dropdown, in this case “English Repository.” The query will be generated, selecting only the repository items for the module instance with the ID of 542. All records are attached with the address and geocode fields matched with their ItemID.

1. Module Table: grmRepositoryObjects <input checked="" type="checkbox"/> Tables with a ModuleId or PortalId column only	3. Portals: My personal site (id=0)
2. Key Columns: <input checked="" type="checkbox"/> ItemID <input type="checkbox"/> Image <input type="checkbox"/> Approved <input type="checkbox"/> ModuleID <input type="checkbox"/> Author <input type="checkbox"/> Name <input type="checkbox"/> AuthorEmail <input type="checkbox"/> PreviewImage <input type="checkbox"/> Clicks <input type="checkbox"/> RatingAverage <input type="checkbox"/> CreatedByUser <input type="checkbox"/> RatingTotal	4. Tabs: data e (id=134) 5. Modules: English Repository (id=542)

Query generated, see below.

The generated query:

```
SELECT t.*,
       ISNULL(m.TableName, 'grmRepositoryObjects') AS TableName,
       ISNULL(m.ID1, t.ItemID) AS ID1,
       m.Address, m.City, m.State, m.Zip, m.Country,
       m.Latitude, m.Longitude, m.Zoom, m.Iconfile
FROM   grmRepositoryObjects AS t
LEFT JOIN
       dbo.Netism_MapExtreme_AttachedLocations AS m
ON     m.TableName = 'grmRepositoryObjects'
       AND t.ItemID = m.ID1
WHERE  t.ModuleID = 542
```

You can now save the query, and enter BatchGeocode for the Match Parameters. Create another custom query, enter the same query and leave the Match Parameters blank. With this custom queries setup, you can use one to fill in location information for the repository items then batch geocode them on the BatchGeocoding page. And use another query to pull the items, supposedly, already geocoded, for the map display.

Optionally, you can add “and isnull(m.Latitude,0)=0 and isnull(m.Longitude,0)=0” for the BatchGeocoding query, so only records not geocoded would show up for geocoding. And you can add “and isnull(m.Latitude,0)<>0 and isnull(m.Longitude,0)<>0” for the default query, so only records already geocoded would show up for the map view.

Furthermore, you can add “SELECT TOP 100 t.*, ...” instead of “SELECT t.*, ...”, so you can geocode only 100 records at a time if you have a lot.

However, some modules, due to their flexibilities in design, don't have a database table structure that our Query Builder could support and properly build the queries with the intended fields. Some examples of those modules: XMod, Property Agent, Forms and Lists (or User Defined Table) and DNN User Profile etc. In those cases, you can either write the custom queries yourself, or you can always contact us.

If you write the query, it must return the TableName and ID1 columns as generated by the Query Builder. It optionally support up to three Key columns with ID2 and ID3. ID1, ID2 and ID3 can be string.

Below is a sample query to first group events by their locations, and then use the location as ID1 to join it to our table attaching the location information. This way, not only you can display individual records, you can also display computed, aggregated or statistical information like the total field from the below query. Ex: “31 events coming up in Soho, New York”, “162 photos in this album” and “31 people in this group” etc.

```
SELECT t.*,
       ISNULL(m.TableName, 'Events') AS TableName,
       ISNULL(m.ID1, t.location) AS ID1,
       m.Address, m.City, m.State, m.Zip, m.Country,
       m.Latitude, m.Longitude, m.Zoom, m.Iconfile
FROM
(
    select location, count(*) as total from Events group by location
) as t
LEFT JOIN
    dbo.Netism_MapExtreme_AttachedLocations AS m
    ON m.TableName = 'Events'
    AND t.location = m.ID1
```

Below is a sample query to select the DNN users and their locations for BatchGeocoding.

```
declare @portalid int
select @portalid = 0

select ISNULL(m.TableName, 'Users') AS TableName,
       ISNULL(m.ID1, u.UserID) AS ID1,
       u.userid as LocationID, u.displayname as Title,
       '' as description, '' as link,
       m.Latitude, m.Longitude, m.Zoom, m.Iconfile,
       up1.PropertyValue as Address,
       up2.PropertyValue as City,
       up3.PropertyValue as State,
       up4.PropertyValue as Zip,
       up5.PropertyValue as Country
from   Users u
join   UserPortals p on u.UserID = p.UserID and p.PortalID = @portalid
left  join UserProfile up1
      on up1.UserID = u.UserID
      and up1.PropertyDefinitionID =
         dbo.GetProfilePropertyDefinitionID(@portalid, 'Street')
left  join UserProfile up2
      on up2.UserID = u.UserID
      and up2.PropertyDefinitionID =
         dbo.GetProfilePropertyDefinitionID(@portalid, 'City')
left  join UserProfile up3
      on up3.UserID = u.UserID
      and up3.PropertyDefinitionID =
         dbo.GetProfilePropertyDefinitionID(@portalid, 'Region')
left  join UserProfile up4
      on up4.UserID = u.UserID
      and up4.PropertyDefinitionID =
         dbo.GetProfilePropertyDefinitionID(@portalid, 'PostalCode')
left  join UserProfile up5
      on up5.UserID = u.UserID
      and up5.PropertyDefinitionID =
         dbo.GetProfilePropertyDefinitionID(@portalid, 'Country')
left  join Netism_MapExtreme_AttachedLocations AS m
      on m.TableName = 'Users'
      and u.UserID = m.ID1
```

Batch Geocoding

The Batch Geocoding page will find and execute a query with the match parameter BatchGeocoding. The returned column with the name Label, Title, Name, or the first text column from the result set will be used for labeling. It would also fill in the Address, City, State, Zip, Country, Latitude and Longitude if any of them have already saved in the database.

Note that this is not an automated or scheduled process. You will need to open and run the batch geocoding form manually.

The initial form, loaded with the query we previously built using with the Query Builder. The repository module has 4 items.

4 records returned. Geocode All! Save All!

#	Label	Address	City	State	Zip	Country	Latitude	Longitude	Geocoded Matches
1.	NY News	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>▼
2.	CT News	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>▼
3.	NJ News	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>▼
4.	PA News	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>▼

Geocode All! Save All!

You can simply fill in any parts of the location information, in this case, just the states.

4 records returned. Geocode All! Save All!

#	Label	Address	City	State	Zip	Country	Latitude	Longitude	Geocoded Matches
1.	NY News	<input type="text"/>	<input type="text"/>	NY	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>▼
2.	CT News	<input type="text"/>	<input type="text"/>	CT	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>▼
3.	NJ News	<input type="text"/>	<input type="text"/>	NJ	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>▼
4.	PA News	<input type="text"/>	<input type="text"/>	PA	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>▼

Geocode All! Save All!

Clicking the “Geocode All!” button would geocode the items based on the address information you fill in. If there are multiple matches, it would load all the matched locations in the Geocoded Matches dropdown. For only one match, it would fill in the Latitude and Longitude.

4 records returned.

#	Label	Address	City	State	Zip	Country	Latitude	Longitude	Geocoded Matches
1.	NY News	<input type="text"/>	<input type="text"/>	NY	<input type="text"/>	<input type="text"/>	42.903789281E	-75.570487976	New York <input type="button" value="v"/>
2.	CT News	<input type="text"/>	<input type="text"/>	CT	<input type="text"/>	<input type="text"/>	41.575179845E	-72.738311290	Connecticut <input type="button" value="v"/>
3.	NJ News	<input type="text"/>	<input type="text"/>	NJ	<input type="text"/>	<input type="text"/>	40.082771256E	-74.649916961	New Jersey <input type="button" value="v"/>
4.	PA News	<input type="text"/>	<input type="text"/>	PA	<input type="text"/>	<input type="text"/>	40.896690264E	-77.838889807	Pennsylvania <input type="button" value="v"/>

Clicking “Save All!” would save all the information to the database table with update and insert queries like the below. In this case, @TableName would be “grmRepositoryObjects” and the @ID1 would be the ItemID from grmRepositoryObjects. @ID2 and @ID3 would be null.

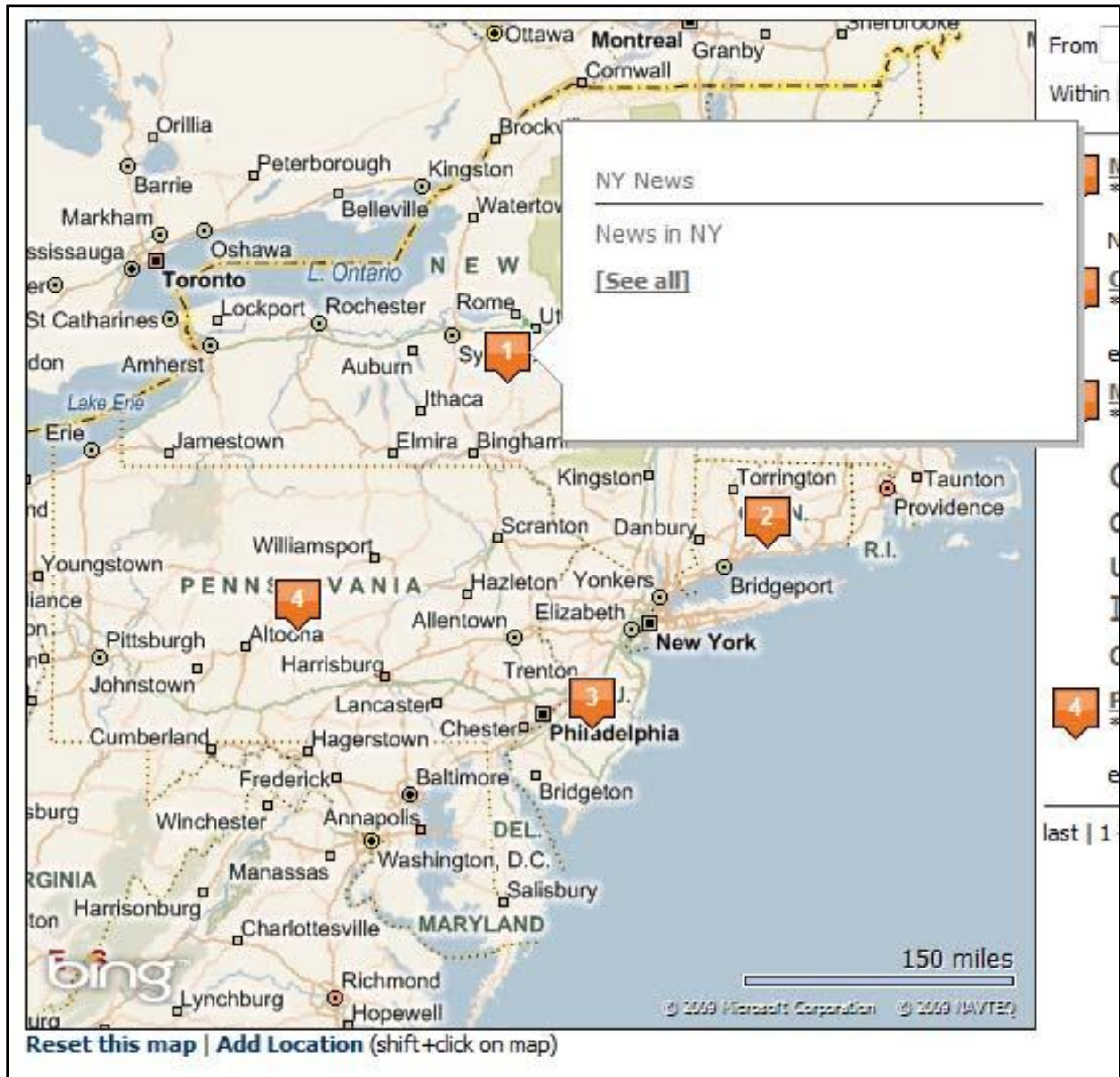
```

update Netism_MapExtreme_AttachedLocations
set   Address = @Address, City = @City, State = @State, Zip = @Zip, Country = @Country,
      Latitude = @Latitude, Longitude = @Longitude, Zoom = @Zoom, Iconfile = @Iconfile
where TableName = @TableName and ID1 = @ID1 and ID2 = @ID2 and ID3 = @ID3

insert Netism_MapExtreme_AttachedLocations
(TableName, ID1, ID2, ID3, Address, City, State, Zip, Country,
 Latitude, Longitude, Zoom, Iconfile)
values (@TableName, @ID1, @ID2, @ID3, @Address, @City, @State, @Zip, @Country,
 @Latitude, @Longitude, @Zoom, @Iconfile)

```

You can set to "Use custom query" under map setting, make sure you have the same query for the default/fallback query. And you should be able to get a map view similar to the below.



Custom Templates

With custom templates, except for the map tile images, you can virtually change all layout and style display you can see with your own layout and styles. The template engine recognizes some special instructions in the templates text which are called “tokens.” It will replace the recognized tokens with values that could be specific and unique to each map view and search. Tokens are enclosed with the { and } characters. If you need to display the { or } character, then use {{ for { and }} for }. Unrecognized tokens would be removed. Enable “use custom template” under Map Setting.

There are 3 main types of tokens:

Map Tokens

Or standard tokens, unique only for each map/search view.

{Total}	Total number of locations returned.
{PortalId}	ID of the current portal.
{TabId}	ID of the current tab (or page.)
{UserId}	ID of the logged in user, or empty if not available.
{ModuleId}	ID of the module.

URL Parameters – All parameters on the URL Querystring can also be referenced. So for the URL like: /map.aspx?zip=12345&type=dealers&speak=Spanish&rate=5 You can reference the parameters by {zip}, {type}, {speak} and {rate}.

Field Tokens

Or location tokens, unique for each map locations. The values would be retrieved from the database unchanged. There are 2 types of field tokens:

Map Extreme Default Query – fields from the Netism_MapExtreme_Locations table.

{LocationID}, {Title}, {Link}, {Description}, {Address}, {City}, {State}, {Zip}, {Country},
{Latitude}, {Longitude}, {Zoom}, {Iconfile}, {Username}, {Ext1}, {Ext2}, {Ext3}, {Ext4},
{Ext5}, {SortOrder}, {IsActive}, {CreatedDate}

Custom Queries – fields from a query defined in the Custom Queries page.

All fields returned could be referenced by tokens, the fields can be physical fields directly from a table column, or dynamically aggregated, computed or aliased fields.

So for the query: “select categoryid, count(*) as articles, avg(rating) as average ...”
You can reference the fields by {categoryid}, {articles} and {average} respectively.

Processed Tokens

They can be unique to each map view or locations. They are results of either combining other tokens result or mixing with some HTML/Javascript code. Note that not all processed tokens are available for all templates.

Module view tokens

{map}	Display code for the map view. It processes values and results from the Map Setting, Permissions and Icon templates.
{list}	Display code for the location list. It combines all list templates results.
{pager}	Display code for the pager bar. It combines all pager templates results.
{search}	Display code for the search form. Code results from the Search Form.

Pager tokens

{last-link}	JS function call to activate the last page for the location item list.
{next-link}	JS function call to activate the next page for the location item list.
{last}, {next}	Display code for the last and next links, it uses {last-link} and {next-link}. You can change to use your own text or image in the templates.
{from}, {to}	The index of the locations on the current page.

Location tokens

{Index}	Index of the location.
{link-directions}	Display code for the direction link.
{link-edit}	Display code for the location edit link.

{zoom-satellite-js}	JS function call to activate the satellite view for the location.
{zoom-street-js}	JS function call to activate the street level zoom.
{zoom-city-js}	JS function call to activate the city level zoom.
{zoom-region-js}	JS function call to activate the region level zoom.

By using the above 4 tokens, you can use your own display code to trigger the functions.

{zoom-satellite}	Display code for the satellite view link.
{zoom-street}	Display code for the street level zoom link.
{zoom-city}	Display code for the city level zoom link.
{zoom-region}	Display code for the region level zoom link.
{icon-idle}	Icon image URL for the location.
{icon-active}	Icon image URL for the location for when it's selected.
{popup-show}	JS function call to show the location popup inside the map.
{popup-hide}	JS function call to hide the location popup inside the map.
{distance}	Place holder for the calculated distance from a location search.

There are 4 main types of templates:

Module View Template

The overall layout of the module is customizable by placing the tokens in different places. If the search form is not needed, just take out {search}. If the pager is not needed, just take out {pager}. If you like the search form to be on the left, on the top, or the list to be on the top or bottom, just change the HTML table structure accordingly.

Available tokens

All map tokens, {map}, {list}, {search} and {pager}

Default template code

```
<table><tr>
<td valign="top">{map}</td>
<td valign="top">{search}<hr/>{list}<hr/>{pager}</td>
</tr></table>
```

Pager Templates

Result of all pager templates is referenced by {pager} in the module view template. If a search returns no matches, {pager} will use the No Match template. If there are results, it will use the Pager template. On any given page, if there's a last page available, {last} would use the Last Active template, otherwise it would use the Last not Active templates, same idea for {next}.

Pager	{last} {from} - {to} of {total} {next}
Last Active	last
Last not Active	last
Next Active	next
Next not Active	next
No Match	no match

Icon Templates

Icon idle and active path are image URLs to use for custom or default icons. They are used as pins in the map display and referenced by {icon-idle} and {icon-active} in the listing templates. The Popup Box template is only for display inside the popup box that appears when a pin is active, or mouse over. It is only used inside the map view.

Icon Idle Path

Available tokens

All map tokens, field tokens and {index}.

Default template code

```
/DesktopModules/NetismMapExt/Images/Pushpin.aspx?id={index}
```

Icon Active Path

Available tokens

All map tokens, field tokens and {index}.

Default template code

```
/DesktopModules/NetismMapExt/Images/Pushpin.aspx?id={index}&selected=1
```

Popup Box

Available tokens

All map tokens, field tokens and processed tokens.

Default template code

```
{title}<hr/>{description}
```

Listing Templates

Result of all listing templates is referenced by {list} in the module view template. If List Alternate Item template is empty, then it would use the List Item template.

List Header

Available tokens

All map tokens.

Default template code

```
{total} location(s) in total<hr/>
```

List Item

Available tokens

All map tokens, field tokens and processed tokens.

Default template code

```
<table><tr> <td valign="top">
  <a href="#" {popup-show} {popup-hide}>
    
  </a>
</td>
<td valign="top">
  <a href="{link}" {popup-show} {popup-hide}>{title}</a>
  {link-edit}<br/> *{distance}*
  {address} {city}, {state} {zip} {country}
  {description}
</td>
</tr></table>
```

List Alternate ItemAvailable tokens

All map tokens, field tokens and processed tokens.

Default template code

empty

List SeparatorAvailable tokens

All map tokens.

Default template code

```
<hr/>
```

List FooterAvailable tokens

All map tokens.

Default template code

```
<br/><hr/>  
<a target="_blank" href="http://www.netismsolutions.com/">Netism Map  
Extreme</a>
```

Search Form

You can define a search form with “Search Tokens” to filter locations dynamically with Ajax, so a full page load is not required. Search tokens can be wrapped with any HTML, Javascript and CSS, allowing you to virtually define a form to filter any fields, with any values, any style, and any layout. Search fields can be set to trigger searches as the user types on a text field, selects an item on a dropdown list, clicks on a checkbox or radio button. It can also be configured to search only when clicking on the button.

There are 10 types of search tokens, each have their own sets of properties.

Config

`type=config` Included on the top of the search form to specify some search behaviors.
`presearch=1` The only property currently supported, search when the map first loads.

Example

```
{type=config, presearch=1}
{type=config}
```

Text

`type=text` A text input field.
`field` The name of the field to search the text for.
`value` Default value, which can be loaded from querystring.

Example

```
{type=text, field=title}
{type=text, field=name, value=store}
```

Location

`type=location` A text input field to be used for specifically for location.
 It can be a full address, just a zip code, just a city name, just a state name, just a country name, or even the name of an attraction or a point of interest etc.
`autocomplete=1` Display location suggestion/corrections as location is inputted.
 It starts on the 3rd entered character.
`value` Default value, which can be loaded from querystring.

Example

```
{type=location, autocomplete=1, value=}
{type=location, value=$zip}
```

Distance

`type=distance` A dropdown menu containing numeric distance values.
 The selected value is used to filter the calculated distance based on the entered location.
`values` List of predefined values separated by |.
`selected` Default value, which can be loaded from querystring.

Example

```
{type=distance, values=0.1|0.5|1|2|5, selected=0.5}
{type=distance, values=10|50|100, selected=$max}
```

Check

type=check	A checkbox.
field	The name of the field to search for the checkbox value.
value	The value of the checkbox.
selected	1 for pre-checking it, 0 otherwise.

Example

```
{type=check, field=country, value=usa, selected=1}
{type=check, field=hasReview, value=true, selected=0}
```

Radio

type=radio	A radio button.
field	The name of the field to search for the radio button value.
value	The value of the radio button.
selected	1 for pre-checking it, 0 otherwise.

Example

```
{type=radio, field=state, value=ny, selected=1}
{type=radio, field=isApproved, value=true, selected=0}
```

Drop List

type=droplist	A dropdown list.
field	The name of the field to search the selected value for.
load	The only supported value for this property is “unique”, which populates the dropdown with a set of unique values from the query field.
first	Insert an item on top of the dropdown list.
values	Append a list of predefined values separated by , to the dropdown list.
selected	Default value, which can be loaded from querystring.

Example

```
{ type=droplist, field=city, load=unique,
  values=Others, selected=$city }
{ type=droplist, field=state,
  first = - pick your state -, values=NY|NJ|PA|CT }
```

Check List

type=checklist	A list of checkboxes.
field	The name of the field to search the checked value for.
load	The only supported value for this property is “unique”, which generates a list of checkboxes with a set of unique values from the query field.
values	Append a list of predefined values separated by , to the checkbox list.
selected	Default value, which can be loaded from querystring.

direction horizontal for displaying the checkboxes from left to right.
vertical for displaying the checkboxes from top to bottom.

Example

```
{ type=checklist, field=city, load=unique,
  values=Others, selected=$city, direction=horizontal }
{ type=checklist, field=state,
  values=NY|NJ|PA|CT, direction=vertical, selected=NY }
```

Radio List

type=radiolist A list of radio buttons.

field The name of the field to search the checked value for.

load The only supported value for this property is “unique”, which generates a list of radio buttons with a set of unique values from the query field.

values Append a list of predefined values separated by |, to the radio button list.

selected Default value, which can be loaded from querystring.

direction horizontal for displaying the radio buttons from left to right.
vertical for displaying the radio buttons from top to bottom.

Example

```
{ type=radiolist, field=city, load=unique,
  values=Others, selected=$city, direction=horizontal }
{ type=radiolist, field=state,
  values=NY|NJ|PA|CT, direction=vertical, selected=NY }
```

Button

type=button A button that triggers a search.

value The label or text of the button.

Note:

- All search tokens can be quickly and temporarily disabled by adding # in the front.
Example: `{#type=check ... }`
- Except for config, checklist and radiolist, all search tokens can include other properties, which will be kept in the generated HTML element as their attributes.
Example: `{type=text, class=bigbox, style=width:100px ... }`

The power of search tokens combined with your imaginations, you can easily and quickly create flexible and powerful search forms to match all your needs, from searching locations, to filtering categories, to complex combinations. Below are 3 quick examples.

Search Form Example 1 – Default Form

```
{type=config, presearch=1}
<table><tr><td colspan="2">
  From
  {type=location, autocomplete=1, value=$zip, style=width:200px}
</td></tr>
<tr><td>
  Within
  {type=distance, values=0.1|0.5|1|2|5|20|50|100, selected=5}
  Miles
</td><td align="right">
  {type=button, value=Search!}
</td></tr></table>
```

The screenshot shows a search form with a text input field containing "From jfk". Below it is a dropdown menu labeled "Within 5 miles" with a downward arrow. To the right of the dropdown is a button labeled "search!".

Search Form Example 2 – Simple Form

```
{type=config, presearch=1}
<table><tr><td>City:</td><td>
  {type=radiolist, field=city, load=unique, direction=horizontal}
</td></tr>
<tr><td>Location Type:</td><td>
  {type=radiolist, field=ext2, load=unique, direction=horizontal}
</td></tr></table>
```

The screenshot shows a search form with two rows of radio button options. The first row is labeled "City:" and has three radio buttons for "Brooklyn", "Manhattan", and "Queens". The second row is labeled "Location Type:" and has three radio buttons for "Office", "Service Center", and "Store".

Search Form Example 3 – Complex Form

```

{type=config, presearch=1}
<table><tr><td>From:</td><td>
  {type=location, autocomplete=1, value=$zip, style=width:200px}
</td></tr>
<tr><td>Within:</td><td>
  {type=distance, values=0.1|0.5|1|2|5|10|20|50|100, selected=2} miles
</td>
<tr><td>Name:</td><td>
  {type=text, field=title, value=, style=width:200px}
<td></tr>
<tr><td>Certified:</td><td>
  {type=radiolist, field=ext1, values=Yes|No, direction=horizontal}
<td></tr>
<tr><td>Type:</td><td>
  {type=checkboxlist, field=ext2, load=unique, direction=horizontal}
<td></tr>
<tr><td valign="top">Speak:</td><td>
  {type=checkboxlist, field=ext3, direction=vertical,
    values=Chinese|Japanese|English|Spanish|German|Italian }
<td></tr>
<tr><td>In Business:</td><td>
  {type=droplist, field=ext4,
    values=1+ Years|3+ Years|5+ Years|10+ Years}
<td></tr>
<tr><td colspan="2" align="right">
  {type=button, value=Search!}
</td></tr></table>

```

The screenshot shows a search form with the following elements:

- From:** Text input field containing "11219, NY".
- Within:** Dropdown menu showing "2" and "miles".
- Name:** Text input field.
- Certified:** Radio buttons for "yes" (selected) and "no".
- Type:** Checkboxes for "Office", "Service Center" (checked), and "Store".
- Speak:** Checkboxes for "chinese" (checked), "japanese", "english" (checked), "spanish", "german", and "italian".
- In Business:** Dropdown menu showing "1+ years" (selected) with a list of options: "1+ years", "3+ years", "5+ years", and "10+ years".
- Search:** A "search!" button.

License

This section only applies to the Portal License level, which allows ME to run on only one portal within a DNN installation at a time. Clicking on it and select a portal to enable ME for the DNN installation. For the Host license level, Enterprise license level and Enterprise + Source license level, it is enabled for all portals.

Permission

This section only applies to the module when it's set to NOT use custom query. Users other than admin and host will need to be in a role checked on the permission page in order to have access to manipulate map locations.