Appendix A: Customizing the AddLayer Wizard

Overview

The Add Layer Wizard is a guided tool that assists users with adding layers of maps. It displays when you click on the Add button in the Layer Control dialog. For an example of the Add Layer Wizard behavior, run the Map Definition Manager and choose File > New Map Definition. From the Layer Control dialog, click the Add button to display the Add Layer Wizard. (The Map Definition Manager is discussed in Chapter 14).

The Add Layer Wizard initializes itself based on values for data sources stored in the **addlayerwizard.properties** file. This file is a text file, installed by the MapXtreme Java installer in your server directory beneath where you installed MapXtreme Java. Most lines in the file contain a key and a value to associate with that key, separated by a '='. You can modify the values in this file to change the configuration of the AddLayer Wizard, including:

- The list of data sources which should appear in the initial list of data sources.
- The list of "Named Resources" which appear when the "Named Resource" data source is selected from the initial list (this data source does not appear by default).
- Default and "most-recently-used" values for the data source information steps of the wizard.
- Default and / or "most-recently-used" values for the initial ("Select a Data Source") and final ("Specify How the Data Will Be Accessed") steps of the wizard.
- Whether passwords are saved in the properties file (default is not to save them.)

Changing the List of Data Sources

The addlayerwizard.properties file installed by MapXtreme Java is configured to offer the following list of data sources in the initial step of the wizard:

1. MapInfo TAB file
2. Oracle8i with Spatial Option
3. Oracle 7/8 with SpatialWare
4. DB2 with SpatialWare
5. IUS with SpatialWare
6. Any with X/Y Column Data
7. GeoTIFF Raster
8. ESRI Shape

If you open the addlayerwizard.properties file in your favorite text file viewer, you will see the following line near the top of the file:

```
DataSource1=MapInfo TAB file
```

The “1” indicates this line specifies that the "MapInfo TAB File" data source should be the first data source in the list. This is followed (in numerical order) by each of the other data sources in the list above.

Removing Data Sources

If you know that you (or the users of your applet/application) will never want to add a layer from one of data sources in the above list, you can simply remove it from the list in the addlayerwizard.properties file. If you remove a data source in the middle of the list, however, you must then adjust the numbers of any data sources that follow it in the list. For example, if you remove the following group of entries, because you don't need to add layers from a DB2 data source:

```
DataSource4=DB2 with SpatialWare

DataSource4_DPHelper_Page=DataSource4_Page2

DataSource4_DPHelper_Class=com.mapinfo.dp.jdbc.db2sw.Db2S
   pwDataProviderHelper

DataSource4_Page_Count=3

DataSource4_Page1=com.mapinfo.beans.addlayer.PageDB2p1
```

```
DataSource4_Page1_Description=Specify DB2 Data Source
   Information

DataSource4_Page2=com.mapinfo.beans.addlayer.PageDB2p2

DataSource4_Page2_Description=Specify DB2 Table or Query
   Information

DataSource4_Page3=com.mapinfo.beans.addlayer.PageDB2p3

DataSource4_Page3_Description=Specify Other DB2 Table or
   Query Information
```

You must then "move up" all of the data sources that follow it (numerically) in the properties file. So you would change the DataSource5 group of entries to be DataSource4 entries, the DataSource6 group of entries to be DataSource5 entries, etc.

We recommend that rather than removing a group of entries from the properties file, that you comment out that group of entries, by placing a '#' character before each line in the group of entries. For example, to comment out the DB2 entries, follow the entry below.

```
#DataSource4=DB2 with SpatialWare

#DataSource4_DPHelper_Page=DataSource4_Page2

#DataSource4_DPHelper_Class=com.mapinfo.dp.jdbc.db2sw.Db2
   SpwDataProviderHelper

#DataSource4_Page_Count=3

#DataSource4_Page1=com.mapinfo.beans.addlayer.PageDB2p1

#DataSource4_Page1_Description=Specify DB2 Data Source
   Information

#DataSource4_Page2=com.mapinfo.beans.addlayer.PageDB2p2

#DataSource4_Page2_Description=Specify DB2 Table or Query
   Information
```

By commenting out the lines, if you need to put the data source back into the list, you can just remove all of the '#'s.

Also of note is that if you remove all of the data sources except for one, the initial step of the AddLayerWizard, which displays the list of data sources, will not display, since there is only one data source from which to choose.

Re-ordering the Data Sources

The number associated with the each group of data source entries determines its place in the list of data sources that will appear in the initial step of the Add Layer Wizard.

It is generally a good idea to have those data sources that you use frequently towards the top of this list. For example, if you know that the users of your applet / application will most often need to add layers from an IUS data source to their map, you might choose to make that data source the first in the list. The default addlayerwizard.properties file indicates the IUS data source will appear as the fifth data source in the list. To make it the first you would just change its group of entries as follows:

```
DataSource1=IUS with SpatialWare

DataSource1_DPHelper_Page=DataSource5_Page2

DataSource1_DPHelper_Class=com.mapinfo.dp.jdbc.iussw.IusS
   pwDataProviderHelper

DataSource1_Page_Count=3

DataSource1_Page1=com.mapinfo.beans.addlayer.PageIUSp1

DataSource1_Page1_Description=Specify IUS Data Source
   Information

DataSource1_Page2=com.mapinfo.beans.addlayer.PageIUSp2

DataSource1_Page2_Description=Specify IUS Table or Query
   Information

DataSource1_Page3=com.mapinfo.beans.addlayer.PageIUSp3

DataSource1_Page3_Description=Specify Other IUS Table or
   Query Information
```

Of course, by doing this you now have two DataSource1 entries in your properties file.To remedy this, you can increment the numbers associated with the data sources that should now follow your new number 1 entry.

Specifying a Named Resources

If you've already looked at the contents of the addlayerwizard.properties file installed by MapXtreme Java, you may have noticed that it includes a ninth data source, Named Resource. You may also have noticed that this data source does not appear in the list of data sources to choose from in the initial step of the wizard. This is because the addlayerwizard.properties file, as installed, does not have any Named ResourceX entries from which a user might choose.

These named resource entries are the same named connections that are managed by the Connections Manager application (see Chapter 10) and stored in the **miconnections.properties** file. Named connections provide an easy way to refer to a particular database connection (or more accurately, a set of database connection properties).

To give the users of your applet / application the ability to add a layer based on one of these named connections without specifying any connection information, you add the named connection to the list of Named Resources in the addlayerwizard.properties.

For example, if you had previously defined a named connection called "Enigma" that defines a connection to an Oracle8i data source, you would add a pair of entries to the addlayerwizard.properties file, to make the Add Layer Wizard aware of the named connection / resource, as illustrated by the bold text below

```
# Named Resources
# Add any named resources that appear in
   miconnections.properties.
# Each named resource entry should include the name of the
   resource as well
# as the data source which is appropriate for that named
   resource.
# Sample named resource:
#NamedResource1=snoopy
#NamedResource1_DataSource=DataSource2
# In this sample, DataSource2 represents the data source
   which is appropriate
# for snoopy.
NamedResource1=Enigma
NamedResource1_DataSource=DataSource2
```

Add the entries right after the Named Resource comment section in the file, so you can manage all of your named resources in one place. The comment section also describes the convention to use when defining a named resource. The only thing which may not be obvious is this line:

```
NamedResource1_DataSource=DataSource2
```

This line indicates that DataSource2 (which by default is the Oracle8i data source) is the data source that corresponds to the named Enigma connection.

You can add as many of these named resources as you like, provided of course, that they coincide with named connections in your miconnections.properties file. Once there is at least one named resource in this list, the "Named Resource" data source will appear in the list of data sources in the initial step of the wizard.

See Chapter 10: Accessing Remote Data for more information on connections and named resources.

Specifying Default Values

It is possible to seed the various Add Layer Wizard controls with default values that will appear pre-selected or specified when the user encounters them. This is very useful, for example, when it is likely that the users of your applet / application will not know the connection information for a remote data source, and you would like to have it filled in automatically for them.

To specify a default value for a control, add a line to addlayerwizard.properties that indicates the page and control to be set (these are combined to form the "key") and the default value. For example, if you wanted to set the data source that would appear pre-selected on the initial DataSourcePage page of the wizard, add a line like this:

```
DataSourcePage_default_datasource=Oracle8i with Spatial
   Option
```

Note that addlayerwizard.properties is installed with this line already included, and MapInfo TAB file specified as the default data source.

The convention to use for specifying a default value in addlayerwizard.properties is:

```
<page>_default_<control name>=<default value>
```

Note: Do not include double quotes around the default values.

Setting Defaults for Initial and Final Page Controls

The initial page (step) of the wizard allows the user to select the data source from which they want to add a layer. The final page allows the user to specify how the data will be accessed, as well as some other miscellaneous values.

To specify default values for the controls in either of these pages, add lines to the properties file using the convention described above to specify the default values.

Page	Control Name	Control Description
DataSourcePage	datasource	The data source to pre-select from the "Available Data Sources" list (eg. "MapInfo TAB file")
DataSourcePage	useprevious	True to pre-select the "Use previous settings as the defaults" option or False to pre-select the "Use default property values as the defaults" option
DataAccessPage	servlet	True to pre-select the "Access Data Via Remote MapXtremeServlet" or False to pre-select the "Access Data Locally" option
DataAccessPage	url	URL - the URL of the MapXtremeServlet

Setting Defaults for Other DataSource Page Controls

Each data source in the Add Layer Wizard has one or more "pages" associated with it, which are used to query the user for the information necessary to add a specific layer from that type of data source. If you look at the contents of addlayerwizard.properties, you will notice the various DataSourceX_PageY entries within it. The default values you specify will be exclusive to one of these pages in the wizard.

The table below enumerates the controls of the various DataSourceX pages for which default values may be specified via the addlayerwizard.properties file.

The Data Source column specifies which data source(s) the value pertains to. "All remote" denotes that the value is pertinent to all remote database data sources,

namely Oracle8i, Oracle 7/8, DB2, IUS, and X/Y. Note the data source values are used here, since your data source numbers may be different due to re-ordering.

The Page column denotes which page the value is pertinent to.

The Control Name column denotes the internal name of the control that should be used to specify a default value.

The Control Description column describes which control in the Add Layer Wizard the key pertains to — in most cases it is the caption of the control.

DataSource	Page	Control Name	Control Description
MapInfo TAB	1	table	MapInfo Table – full path to.tab file
All remote	1	host	Host – name or IP address of the host database machine
Oracle8i, DB2, IUS	1	port	Port – port on which the remote database is listening
Oracle8i, Oracle 7/8	1	sid	Database Instance (SID)
Oracle8i, DB2, IUS, X/Y		user	User Name
Oracle8i, DB2, IUS, X/Y	1	pwd	Password
Oracle8i, DB2, IUS	1	drivertype	Driver Type – one of "use default", "thin", "thick", "thickbequeth", "kernel"
Oracle8i	1	rows	Number of Rows to Pre-fetch
Oracle 7/8	1	dsn	Data Source Name (DSN)
Oracle 7/8	1	dbuser	Database User Name
Oracle 7/8	1	dbpwd	Database Password
Oracle 7/8	1	hostuser	Host User Login
Oracle 7/8	1	hostpwd	Host Password
DB2, IUS	1	db	Database Name
IUS	1	server	Server Name
X/Y	1	driver	JDBC Driver Class Name

DataSource	Page	Control Name	Control Description
X/Y	1	url	JDBC Connection URL
GeoTIFF	1	file	GeoTIFF File – full path to .tif file
ESRI Shape	1	shapefile	ESRI Shape File – full path to .shp file
ESRI Shape	1	encoding	Char-Set Encoding
All remote	2	table	Table
Oracle8i, Oracle 7/8, DB2, X/Y	2	owner	Owner Name
All remote	2	quotes	Use Quotes – "true" or "false"
All remote	2	query	"true" to pre-select the "SQL Query" option, or "false" to pre-select the 'Table" option
All remote	2	querytext	SQL Query – the text of the SQL query
All remote	2	idcol	ID Column
All remote	3	specify	"true" to pre-select the "Use the following settings:" option, or "false" to pre-select the "Query the MAPINFO_MAPCATALOG for other settings"
Oracle8i, Oracle 7/8, DB2, IUS	3	spatialcol	Spatial Column
X/Y	3	xcol	X Column
X/Y	3	ycol	Y Column
Oracle8i	3	dimension	Geometry Dimension
All remote	3	userenditions	Per-feature Renditions – one of "Don't use", "MapInfo MapBasic format", "MapXtreme Java format", "Ask MAPINFO_MAPCATALOG"
All remote	3	renditioncol	Rendition Column

For example, when the user selects the "MapInfo TAB File" data source, you can define a default MapInfo table to appear filled in the Wizard.

Add a line like this:

```
DataSource1_Page1_default_table=d:\\maps\\world.tab
```

Most-Recently-Used (MRU) Values

The Add Layer Wizard can persist the set of values that were most recently used from session to session. If the Add LayerWizard has write access to the addlayerwizard.properties file, it will write out various MRU value lines each time the "Finish" button is clicked. These values will be pre-set in the initial page of the wizard if the user selects the "Use previous settings as the defaults" option. This feature avoids repetitive typing of the same or similar settings on the user's part.

Note: the Add Layer Wizard is **not** able to write to this file if deployed within an applet.

Saving Passwords

By default, the Add Layer Wizard is configured so that it does not save most-recently-used passwords to the properties file. Passwords are required as part of the connection information when adding a layer contained in a remote database. For obvious security reasons, these are not saved in the properties file by default. If, however, you would like to have your MRU passwords saved in the properties file, you need only change one line in addlayerwizard.properties. Change the very first line in the properties file so that it looks like:

```
Save_Passwords=true
```

Appendix B: Uploading TAB Data to Remote Databases

Introduction

EasyLoader v 6.0 is a Windows-only utility available from MapInfo Corporation that allows you to upload MapInfo .tab files to a remote database. The following databases are supported:

- Oracle8i (v 8.1.5 or higher)
- Informix Universal Server with SpatialWare DataBlade
- DB2 Universal Server with the SpatialWare Extender

A copy of EasyLoader is available for download from the MapInfo Professional area of the Try it / Buy It page on the MapInfo website at **testdrive.mapinfo.com/mipro**.

Note: Note: This version of EasyLoader does not create or update Rendition columns in the MAPINFO_MAPCATALOG to support MapXtreme Java's per-Feature Rendition feature. If you have already created such columns, EasyLoader will ignore them. Future versions of EasyLoader will support Rendition columns. Check the MapInfo Web site for updates . See Appendix C for more information on MAPCATALOG.

Running EasyLoader

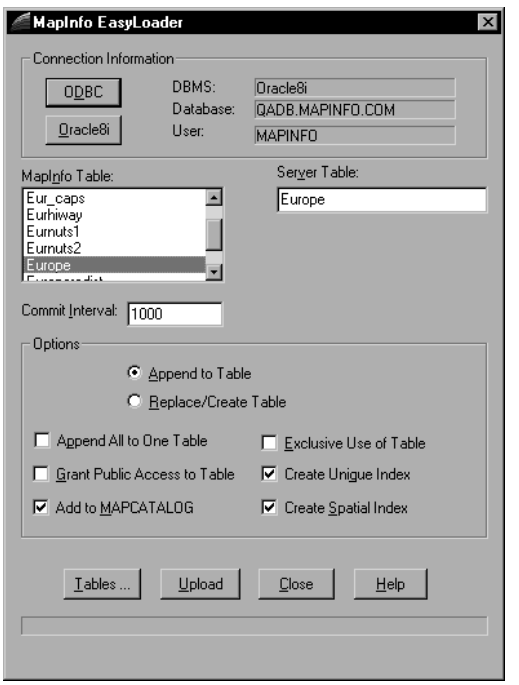
To upload MapInfo .tab files using EasyLoader: Run the EasyLoader executable from Windows. The EasyLoader dialog displays.

1. At the EasyLoader dialog in the Connection group, click the appropriate Connection button (Oracle8i or ODBC) to connect your database. Provide the necessary connection information (e.g., User ID, password and server name). Click OK to return to the EasyLoader dialog.
2. Click the Tables button to display a list of MapInfo tables from a single directory. When tables are selected for uploading, the names will display in the MapInfo Tables list box.
3. Under the Options group, choose Create / Replace table. Uncheck Create Unique Index and Create Spatial Index if you wish to create them after uploading.
4. Click the Upload button to start the upload process. Close EasyLoader once the upload process is finished.

5. If you haven't already created the Spatial Index during the upload process, do so now by either executing a create index statement or re-uploading the table, making sure this time to create the Spatial Index and replace the table (see steps 1-4).

EasyLoader Options

The Upload button becomes available after you have selected the tables to upload. Be sure to specify your table options before you upload the tables. The Options are described below:



Append toTable

The MapInfo table will be appended to the server table if the server table exists and the structure of the two tables match. Otherwise, you will get an error and the table will not be uploaded.

Replace/Create Table

The server table of the same name is first dropped if it exists, then a new table is created to match the MapInfo table being uploaded.

Append All to One Table

All MapInfo tables listed are uploaded to a single server table. The server table name is the one visible in the Server Table box. This feature is meant to be used to upload tables with the same structure and symbology to one table. For example, instead of creating a new table for each street layer, check the Append All to One box, and only one table will be created. Then all of the tables will be appended to this table. Note: It is possible that some tables will not be appended if their structure differs.

When this option is used with the Append toTable option, the tables will all be appended to the existing server table.

When used with the Replace / Create Table option, the server table will be dropped, a new table created, and all tables listed will be appended to that one.

Note: All tables should have the same Projection.

Grant Public Access to Table

PUBLIC is granted all access to the server table.

Add to MAPCATALOG

After the table is uploaded, an entry is made in the MAPINFO.MAPINFO_MAPCATALOG, if it exists.

If the map catalog does not exist, EasyLoader will attempt to create it and issue the proper grants to it. When using ODBC it will not issue public grants on the MAPINFO.MAPINFO_MAPCATALOG, which need to be done by other means. If a user does not have adequate permissions then the creation of the MAPINFO.MAPINFO_MAPCATALOG will not succeed, and no entry will be made into the MAPINFO.MAPINFO_MAPCATALOG.

If the table is made up of a single type of object, then the server object type is restricted to that type, otherwise the type is ALL. Also the symbol clause generated is based on the server type. Example: after uploading table States.tab the server type will be X.2 (polygons), and the symbol clause will have only the information for a polygon.

Exclusive Use of Table

You can speed up load time on large tables significantly if you know that you will be the only one attempting to update the table. Note however, that specifying this option does not guarantee that loader will obtain exclusive use; you must guarantee that to the loader.

The loader checks on the current maximum value of the primary key column (prinx) after each commit, to ensure that it detects any other entries that may have been made by other processes. This option will prevent that check from occurring, which can make a significant change to the run time for large tables.

Create Unique Index

A unique index is created on the column sw_member for SpatialWare or mi_prinx for Oracle Spatial. The mi_prinx column is a sequential number that is generated by the loader.

Create Spatial Index

For Oracle Spatial tables the spatial index is created on the geometry column and is called <table_name>_SX. The index tiling level is based on the SDO_TUNE.ESTIMATE_TILING_LEVEL function. For tables with fewer than 7500 rows, the tiling level is restricted to 8. After the index is built theANALYZE table function is run on the index table.

For SpatialWare tables the index is created on the column geometry column called hg<table_name>ind. A spatial index is created and Update Statistics is executed after an rtree index is created for SpatialWare.

You may also build your own spatial index to suit your specific needs. If you choose to do this, clear this check box to save time in loading.

For more information on how to run EasyLoader, please view the online help provided with this utility, "EasyLoader.HLP".

Appendix C: MAPINFO_MAPCATALOG

Introduction

The MAPINFO_MAPCATALOG is a registry table for databases that stores metadata about geometry tables in the database. Using the tablename and ownername as the key, the MAPINFO_MAPCATALOG identifies the geometry column, geometry type, projection, projection bounds, and table and feature level rendition information The MAPINFO_MAPCATALOG is not specific to MapXtreme Java; it is used by several MapInfo products that access map data from databases.

While many MapInfo products require the MAPINFO_MAPCATALOG to be present, MapXtreme does not require it if all the data that it would otherwise obtain from the MAPINFO_MAPCATALOG is provided to it in the TableDescHelper constructors for table defined layers. Specifically it needs to know the geometry column name, coordinate system and rendition column information. The absence of any of this data will cause MapXtreme to query the MAPINFO_MAPCATALOG and to throw an exception if the data is not available.

If you are familiar with MapInfo Professional, creating a MAPCATALOG is handled by the MapBasic application MIODBCAT.MBX. It creates a table with MAPINFO as the user.

MapXtreme Java SQL Scripts

If your database does not have a MAPCATALOG and you do not have access to MapInfo Professional, we provide SQL scripts that create the MAPCATALOG. The following files are located in \sampleapps\scripts after installation:

- ORA_MAPCAT.SQL - for Oracle
- IUS_MAPCAT.SQL - for Informix Universal Server
- DB2_MAPCAT.SQL - for DB2

These scripts create a MAPCATALOG that contains three string columns that are new to MapInfo products: RENDITIONCOLUMN, RENDITIONTYPE and RENDITIONTABLE. These columns support MapXtreme Java's ability to store rendition information on a per-Feature basis.

RENDITIONCOLUMN contains the name of the column in the geometry table that holds the style information, or null if there is no style column in the geometry table.

RENDITIONTYPE column contains an enumeration flag to indicate how the RENDITIONCOLUMN is to be interpreted by MapXtreme Java. These options are supported:

- No Per Feature Column Present (1)
- MapBasic style rendition (2)
- MapXtreme Java 3.0 style rendition (3)

The RENDITIONTABLE column is reserved for future use.

The following shows the general create table statement and table structure. For more specific information for your database, view the SQL script.

```
Create Table MAPINFO_MAPCATALOG (
                    SPATIALTYPE        Float,
                    TABLENAME          Char,
                    OWNERNAME          Char,
                    SPATIALCOLUMN      Char,
                    DB_X_LL            Float,
                    DB_Y_LL            Float,
                    DB_X_UR            Float,
                    DB_Y_UR            Float,
                    COORDINATESYSTEM   Char,
                    SYMBOL             Char,
                    RENDITIONCOLUMN    Char,
                    RENDITIONTYPE      Integer,
                    RENDITIONTABLE     Char,
                    XCOLUMNNAME        Char,
                    YCOLUMNNAME        Char
                      )
```

MAPCATALOG and EasyLoader v 6.0

If you are using MapInfo's EasyLoader utility to upload MapInfo .tab data into your database for use in MapXtreme Java, note that EasyLoader creates a MAPCATALOG that does NOT contain the new Rendition columns. Also note that EasyLoader will ignore those columns, if they exist, for example, if you created the MAPCATALOG using the SQL scripts described above. For more on EasyLoader, see Appendix B.

Appendix D: Geocoding Resources

Overview

When creating your own data, such as store locations, ATM machines, or cell towers, for use with MapXtreme, it must first undergo a process of spatial codification to turn those records into geographic information. This section describes this process known as geocoding and presents several products developed by MapInfo Corporation that can assist you in preparing your data for map display.

Using maps to illustrate geographic relationships within business data is important for businesses that want to maximize the hidden potential of their data. Spatial analysis reveals the trends, patterns, and opportunities that otherwise are lost when sifting through huge databases of information. To take advantage of the power of mapping and spatial analysis, data must first be enhanced through geocoding.

Geocoding is the process of assigning geographic coordinates (latitude/longitude) to data. Most business data contain a geographic component such as a street address or a ZIP Code. A geocoder simply codifies that component to allow spatial analysis or visual display of the data on a map.

This chapter presents several products offered by MapInfo Corporation that geocode your data in preparation for displaying it on a map. Consider these when designing your mapping application with MapXtreme Java Edition. One solution, MapMarker J Server, has been specifically designed as a Java-based extension to MapXtreme. Other geocoders include MapMarker for Windows and Sun, MapMarker Plus with GDT's premier data set Dynamap 2000, Oracle Geocoding Cartridge, MapInfo Geocoding DataBlade module for IUS, and MapInfo Professional's geocoding feature.

MapMarker and MapMarker Plus

MapMarker is MapInfo Corporation's premier geocoding product that provides batch and single-record address matching to U.S. street level or ZIP Code centroids. It can be installed as a stand-alone workstation or on a network server for multi-user access. MapMarker's primary function is to match your records to addresses in its Address Dictionary and assign latitude and longitude coordinates to them. MapMarker accesses records locally from.TAB or .DBF tables or remotely via ODBC from Oracle, SpatialWare, Informix, Access, and Excel tables.

MapMarker and MapMarker Plus for Windows offer a wizard-like user interface that walks you through the column selection phase prior to geocoding. Choose from automatic or interactive geocoding (where you choose a match from a list of candidates), street-level or ZIP Code centroid matching, and geocoding all records in a table or only unmatched. Other features include the ability to append attributes from another table to your matched records either during the geocoding pass or as a separate operation. During interactive geocoding, when MapMarker finds a record it cannot match and offers you some possible matches, known as candidates, you can visually check their viability by displaying them on a map before making your choice. MapMarker's operations are controlled by a wide variety of system and geocoding preferences that you can easily control from the interface.

MapMarker and MapMarker Plus are differentiated by their address dictionaries. The standard MapMarker product includes a U.S. Address Dictionary that combines U.S. Census Bureau TIGER 1998 files, current USPS ZIP+4 addresses, and GDT's ZIP+4 centroids. The Address Dictionary is updated twice a year to include recent changes in address information.

MapMarker Plus provides all the power and features of the standard MapMarker product, plus an enhanced address dictionary to increase the potential for matches. This Address Dictionary is updated bimonthly to comply with USPS CASS geocoding requirements. Instead of straight TIGER 1998 files, the Plus Address Dictionary contains GDT's premium Dynamap 2000 database of enhanced address ranges and streets. Use MapMarker's CASS geocoding mode to standardize your data according to USPS postal standards and prepare it for bulk-mailing discounts. CASS geocoding is the strictest level of address matching available from MapMarker.

MapMarker supports the creation and use of customized address dictionaries to augment or replace the MapMarker Address Dictionary. For geographic areas that are expanding faster than MapMarker's bimonthly data sets can accommodate, you can build your own table of streets and addresses to increase the geocoding hit rate of your data.

On the development side, MapMarker and MapMarker Plus for Windows come with an OCX for integrating MapMarker into your own applications. For even more control in application development, the underlying OLE Automation API is provided as well. For C programmers who wish to build a geocoding application from scratch, the MapMarker Geoengine API and RPC Server API are included.

MapMarker and MapMarker Plus are available on Sun Solaris in the form of a C API, RPC Server, and U.S. street-level Address Dictionary.

Also available are geocoding datablades for Informix Universal Server on Solaris, one for storing X,Y coordinates in separate columns, another for storing the point as a SpatialWare data type. These datablades are add-on products to MapMarker for Sun and HP.

MapMarker J Server

This product is the latest development tool in the MapMarker family, well-suited to intranet or Internet environments where multiple platforms and multiple users are of concern. MapMarker J Server offers a Java API for creating client geocoding applications that can be run on the platform of your choice. Because it uses a Java API, the geocoding client can reside on any platform, such as Windows NT, Sun, or Macintosh. The client sends geocoding requests to the MapMarker Server, which is currently limited to running on Windows 95/98, Windows NT, or Sun Solaris or HP.

Use this tool in applications where you need to send geocoding requests over a corporate intranet via TCP/IP or the Internet via HTTP. The HTTP method requires that the web server supports servlets, a server-side application that communicates with the MapMarker Server and geoengine when a firewall blocks the client from accessing the server via TCP/IP.

The MapMarker J Server product consists of the Java Server, the Java API for creating the client, sample client and administrator programs, and documentation in HTML format. MapMarker or MapMarker Plus for Windows or Sun/HP is a required additional purchase.

Also available as a separate purchase is the MapInfo Geocoding Cartridge that allows Oracle8*i* users to access MapMarker and store geocoded records with an Oracle8*i* database.

Geocoding with MapInfo Professional

MapInfo Professional provides an all-purpose geocoding function that geocodes records to street level or boundary centroid. For street-level geocoding, this requires county-level tables of StreetInfo or StreetPro. It is suitable for localized geocoding. Boundary centroid geocoding is appropriate for broad analyses, such as thematic shading across territories where street-level accuracy for each record in unnecessary.

MapInfo Professional's geocoder allows you to restrict matching to certain towns or cities, by specifying a boundary that limits the search area. For example, use a refining county boundary table when trying to geocode a record on Main St. in a town where there may be other Main St.'s in towns within the same county. Without the refining boundary MapInfo would match to the first Main St. it found in the county, not necessarily the one in the correct town.

Comparing MapMarker with MapInfo's Geocoding Feature

Both MapMarker and MapInfo Professional's geocoding processes assign geographic coordinates and create points for your data so that you can display them in a Map window. Both geocode to street level accuracy, spotting your records to the exact side of the street if possible.

The difference between the two geocoding methods is that the MapMarker family of products can do this using a single nationwide matching table (Address Dictionary) that allows you to geocode records anywhere in the United States in a single pass.

MapInfo Professional's geocoder uses county StreetInfo or StreetPro files as its search table. Each county level file must be opened and searched individually, so geocoding nationwide cannot be done in one pass.

MapMarker can differentiate among streets with the same name, thus eliminating the need to use a refining boundary, as is necessary when using MapInfo's geocoder.

MapMarker provides a lot of control with the accuracy with which it makes a match. You can set or relax matching conditions to increase the likelihood of a match. If MapMarker cannot match on street level, it can fall back to ZIP Code centroid in the same pass. When using MapInfo Professional's geocoder, geocoding to street level and ZIP Code centroids must be carried out in separate passes with different data sets.

Appendix E: Custom Symbols

These custom symbol GIF images can be found in the \server\fonts\custom directory.


AMBU1-32.gif

BADG1-32.gif

BADG2-32.gif

BANK1-32.gif


BANK2-32.gif


BOOK1-32.gif


CAMP1-32.gif


CAR1-32.gif


CAUT1-32.gif

CHUR1-32.gif

COMP1-32.gif


DB-CON.gif


FARM1-32.gif


FAST1-32.gif


FIRE1-32.gif


GLOB1-32.gif


GOLF1-32.gif


HOSP1-32.gif


HOUS1-32.gif


HOUS2-32.gif


HOUS3-32.gif


HYDR1-32.gif


INTE1-32.gif


LITE1-32.gif


LITE2-32.gif


MAIL1-32.gif


MBOX1-32.gif


MBOX2-32.gif


MOSQ1-32.gif


ONEW1-32.gif


ONEW2-32.gif


PENC1-32.gif


PIN1-32.gif


PIN2-32.gif


PIN3-32.gif


PIN4-32.gif


PIN5-32.gif


PIN6-32.gif


POLI1-32.gif

RAIL1-32.gif

RAIL2-32.gif

RAIL3-32.gif

REST1-32.gif

STAT1-32.gif

STOP1-32.gif

SYNA1-32.gif

TARG1-32.gif

TAXI1-32.gif

TEMP1-32.gif

TOWE1-32.gif

TOWE2-32.gif

TRAF1-32.gif

TRUC1-32.gif

TRUC2-32.gif

YIEL1-32.gif

YIEL2-32.gif

Appendix F: Map Definitions and Geosets

Map Definitions

Map Definitions are XML-based text files that describe a set of map layers in MapXtreme Java. Because they are in XML format, the information can be stored as a file (extension .MDF) or as a record in a database.

Map Definitions are created using the Map Definition Manager. Any layer that can display in MapXtreme Java can be saved as a Map Definition using the MDM. For example, any sample geoset that ships with MapXtreme Java can be saved as a Map Definition. For instructions on using Map Definition Manager, see Chapter 14: Managing Your Data.

Geosets

MapXtreme Java can also display map layers that are saved as geosets. Geosets are files ending in .GST that contain information about a set of layers, and can be loaded at one time. A geoset is loaded by specifying one at design time (as a property), or using the **AddGeoset** method of the Layers object. Additionally, geosets can be loaded via the Map Definition Manager.

Geosets are limited to MapInfo .tab format files and as a result, cannot be stored as a record in a database. Additionally, you cannot change the renditions on a per-feature basis as you can for a Map Definition. Therefore, we strongly recommend that you use Map Definitions in order to have the most flexibility and control when loading and displaying layers in MapXtreme Java.

MapXtreme Java ships with a wide variety of data in geosets. The following Appendix has a complete listing.We recommend that you open any of these sample geosets to learn more about map layers, then save the geoset as a Map Definition. See the instructions in Chapter 14.

A geoset file is an ASCII file containing strings that consist of keys and values. The keys correspond to properties in MapXtreme—properties for the main Map object, as well as for Layer objects.

Keys are hierarchical in nature, and are specified as quoted strings. Key values are also quoted values—even numbers are quoted. The following is a sample showing some keys and values:

```
"\geoset\name" = "Europe"
"\geoset\projection" = "3,0,0,25.0,35.0,40.0,65.0,0.0,0.0"
"\geoset\center" = "-534.315701,1476.882519"
"\geoset\mbr\lowerleft" = "-2973.525440,92.890436"
"\geoset\mbr\upperright" = "1904.894038,2860.874602"
"\geoset\zoomlevel" = "4992.000000"
"\table\6\file" = "eurnuts2.tab"
"\table\6\autolabel" = "true"
"\table\6\zoom\min" = "0.000000"
"\table\6\zoom\max" = "300.000000"
"\table\6\display\brush\forecolor" = "16777215"
"\table\6\display\brush\backcolor" = "16777215"
"\table\6\display\brush\transparent" = "0"
"\table\6\display\pen\linestyle" = "1"
"\table\6\display\pen\linewidth" = "1"
"\table\6\label\font\style" = "0"
"\table\6\label\font\size" = "7"
"\table\6\label\font\forecolor" = "8421504"
"\table\6\label\font\backcolor" = "13696976"
"\table\7\file" = "europe.tab"
"\table\7\autolabel" = "true"
"\table\7\label\font\style" = "1"
"\table\7\label\font\description" = "Arial"
"\table\7\label\font\size" = "8"
"\table\7\label\font\forecolor" = "8388608"
"\table\7\label\font\backcolor" = "13696976"
"\table\7\label\linetype" = "0"
"\table\7\label\position" = "0"
```

The first 8 lines show keys beginning with \geoset.' These are keys that set properties for the entire map, or properties for the Map object. Notice that some keys, like \geoset\mbr are multi-level—there is a lowerleft and an upperright to describe the extend of the map.

The remaining lines show some key settings for two layers—europe.tab and eurnuts2.tab. All layer keys begin with \table. The next word refers to the layer number. The number corresponds to the position in the map, as seen from top to bottom in Layer Control. The next level describes properties for a layer, including its display, labeling, and zoom. In the next level the properties that describe particular aspects of the layer properties, such as the font used for labeling or pen and brush used for the display. Every element of how the layer looks is defined in a geoset key.

The following is a summary of the supported geoset keys in MapXtreme Java:

Key	Description
\GEOSET\NAME	Friendly name of geoset.
PROJECTION	Projection clause
CENTER	Number, Number – center of the map – MapJ.get/setCenter()
\MBR\LOWERLEFT	Number, Number – lower left corner – MapJ.get/setBounds()
\MBR\UPPERRIGHT	Number, Number – upper right corner – MapJ.get/setBounds()
ZOOMLEVEL	Number – Zoom level of the map – MapJ.get/setZoom()
MAPUNIT	Number - corresponds to LinearUnit. 0 = mile, 1 = kilometer, 2 = inch, 3 = foot, 5 = millimeter, 6 = centimeter, 7 = meter, 8 = survey foot, 9 = nautical mile, 10 = TWIP, 11 = point, 12 = pica, 30 = link, 31 = chain, 32 = rod.
TABLE\NUMBER	
FILE	String – name of .TAB file – Layer.get/setName(). TAB file must be in same directory as .GST. This key preferred over Description key.
DESCRIPTION	String – name of .TAB file – Layer.get/setName(). TAB file must be in same directory as .GST.
ISVISIBLE	TRUE or FALSE whether layer is visible – Layer.is/setVisible()
AUTOLABEL	TRUE or FALSE whether layer is autolabeled – Layer.is/setAuto-label()
SELECTABLE	TRUE or FALSE – Corresponds to Layer.is/setSelectable()
ZOOM\MIN	Number - minimum zoom value to display layer –Layer.get/set-ZoomMin()
ZOOM\MAX	Number – maximum zoom value to display layer -Layer.get/set-ZoomMax()

Key	Description
DISPLAY\BRUSH	
PATTERN	Number – corresponds to Rendition.FILL
FORECOLOR	Number – corresponds to Rendition.FILL
BACKCOLOR	Number – corresponds to Rendition.FILLS
TRANSPARENT	TRUE or FALSE – corresponds to Rendition.FILTER_EFFECTS – use Rendition.FilterEffects.NONE for transparent or Rendition.FilterEffects.BOX for opaque.
DISPLAY\PEN	
LINEWIDTH	Number – corresponds to Rendition.STROKE_WIDTH
LINESTYLE	Number – corresponds to Rendition.STROKE
COLOR	Number – corresponds to Rendition.STROKE
DISPLAY\LINEPEN	
LINEWIDTH	Number – corresponds to Rendition.STROKE_WIDTH
LINESTYLE	Number – corresponds to Rendition.STROKE
COLOR	Number – corresponds to Rendition.STROKE
DISPLAY\SYMBOL	
TYPE	Number – Corresponds to Rendition.SYMBOL_MODE
CODE	Number – Corresponds to Rendition.SYMBOL_STRING
COLOR	Number – Corresponds to Rendition.SYMBOL_FOREGROUND
POINTSIZE	Number – Size of symbol in points. Corresponds to Rendition.FONT_SIZE
LABEL	
ZOOM\MIN	Number – minimum zoom value to label layer. LabelProperties.get/setZoomMin()
ZOOM\MAX	Number – maximum zoom value to label layer. LabelProperties.get/setZoomMax()
FONT	This is the label font. It has the same sub-keys as DISPLAY\FONT above
DUPLICATE	TRUE or FALSE – permit duplicate labels. Corresponds to LabelProperties.get/setDuplicationAllowed()
PARALLEL	TRUE or FALSE – Corresponds to LabelProperties.get/setLineLabelHorizontal(). APARALLEL value of TRUE means that horizontal is false, while a value of FALSE means that the label is horizontal.
OVERLAP	TRUE or FALSE – Corresponds to LabelProperties.get/setOverlapAllowed()
OFFSET	Number – Corresponds to LabelProperties.get/setOffset()
POSITION	Number – Corresponds to LabelProperties.get/setHorizontalAlignment AND LabelProperties.get/setVerticalAlignment

Key	Description
LABEL\FONT	
STYLE	Number – Bit Mask (BOLD 0x01), ITALIC 0x02, UNDER 0x04, STRIKEOUT 0x08, OUTLINE 0x10, SHADOW 0x20, INVERSE 0x40, BLINK 0x80)
EXTSTYLE	Number – Bit Mask (HALO 0x01, ALLCAPS, 0x02, DBLSPACE 0x04)
DESCRIPTION	String – font name. Corresponds to Rendition.FONT_FAMILY
SIZE	Number – size in points
FORECOLOR	Number – Corresponds to Rendition.SYMBOL_FOREGROUND
BACKCOLOR	Number – Corresponds to Rendition.SYMBOL_BACKGROUND
OPAQUE	TRUE or FALSE – Corresponds to Rendition.SYMBOL_OPACITY