Table of Contents

Chapter 1: Introduction to MapXtreme .................................................. 19
  Overview of MapXtreme ................................................................. 20
    Key Features ............................................................................. 20
  Migrating to MapXtreme ................................................................. 22
  Learning to Use MapXtreme ........................................................... 24
    Support Resources .................................................................... 24

Chapter 2: Getting Started ................................................................. 27
  Installation Requirements ............................................................... 28
    System Requirements ................................................................ 28
    Minimum System Requirements ................................................. 29
    Additional Software Requirements .......................................... 29
    Supported Databases ................................................................ 29
  Types of Installations .................................................................. 30
    Development (SDK) Installations ............................................... 30
    Deployment (Runtime) Installations .......................................... 30
    Side-By-Side Installations and Use ........................................... 30
  MapXtreme Licensing ................................................................. 31
    Types of Licenses ....................................................................... 31
    Obtaining a License ................................................................ 32
    Locations of License Files ....................................................... 33
    Troubleshooting ....................................................................... 34
    Known Issues with Licensing .................................................. 36

Before You Install ........................................................................... 36
  Administrator Privileges .............................................................. 36
  Install .NET Framework and Visual Studio First. ......................... 36
  Windows 2000 Hotfix Package .................................................... 37
  IIS 7.0 on Windows Vista ............................................................. 37
  Default Install Directory for MapXtreme Under Wow64 ............... 38
  Additional Installation Features .................................................. 38
  Installing MapXtreme in Your Environment ............................... 39
<table>
<thead>
<tr>
<th>Chapter 3: Mapping Concepts</th>
<th>57</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping and MapXtreme</td>
<td>58</td>
</tr>
<tr>
<td>Maps</td>
<td>58</td>
</tr>
<tr>
<td>Tables</td>
<td>58</td>
</tr>
<tr>
<td>Layers</td>
<td>59</td>
</tr>
<tr>
<td>Features</td>
<td>59</td>
</tr>
<tr>
<td>Labels and Legends</td>
<td>59</td>
</tr>
<tr>
<td>Themes</td>
<td>60</td>
</tr>
<tr>
<td>Tools</td>
<td>61</td>
</tr>
<tr>
<td>Workspaces</td>
<td>61</td>
</tr>
<tr>
<td>Coordinate Systems and Projections</td>
<td>61</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Geocoding with MapXtreme</th>
<th>62</th>
</tr>
</thead>
</table>

| Routing with MapXtreme | 62 |

<table>
<thead>
<tr>
<th>Chapter 4: Understanding the MapXtreme Architecture</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapXtreme Architecture</td>
<td>64</td>
</tr>
<tr>
<td>Object Model Overview</td>
<td>65</td>
</tr>
<tr>
<td>MapInfo.Data Namespace</td>
<td>65</td>
</tr>
<tr>
<td>MapInfo.Data.Find Namespace</td>
<td>66</td>
</tr>
<tr>
<td>MapInfo.Engine Namespace</td>
<td>66</td>
</tr>
<tr>
<td>MapInfo.Geometry Namespace</td>
<td>66</td>
</tr>
<tr>
<td>MapInfo.Mapping Namespace</td>
<td>66</td>
</tr>
<tr>
<td>MapInfo.Mapping.Legends Namespace</td>
<td>66</td>
</tr>
<tr>
<td>MapInfo.Mapping.Thematics Namespace</td>
<td>66</td>
</tr>
<tr>
<td>MapInfo.Persistence Namespace</td>
<td>66</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>MapInfo.Raster Namespace</td>
<td>67</td>
</tr>
<tr>
<td>MapInfo.Styles Namespace</td>
<td>67</td>
</tr>
<tr>
<td>MapInfo.WebControls Namespace</td>
<td>67</td>
</tr>
<tr>
<td>MapInfo.Windows Namespaces</td>
<td>67</td>
</tr>
<tr>
<td>MapInfo.Tools Namespace</td>
<td>67</td>
</tr>
<tr>
<td>MapInfo.Geocoding Namespace</td>
<td>67</td>
</tr>
<tr>
<td>MapInfo.Routing Namespace</td>
<td>68</td>
</tr>
<tr>
<td>Application Architectures</td>
<td>68</td>
</tr>
<tr>
<td>Web Application Architecture</td>
<td>69</td>
</tr>
<tr>
<td>Desktop Application Architecture</td>
<td>71</td>
</tr>
<tr>
<td>Chapter 5: Web Applications, Controls, and Tools</td>
<td>73</td>
</tr>
<tr>
<td>Web Application Request/Response Lifecycle</td>
<td>74</td>
</tr>
<tr>
<td>Components of a MapXtreme Web Application</td>
<td>74</td>
</tr>
<tr>
<td>MapXtreme Session</td>
<td>74</td>
</tr>
<tr>
<td>Background Map</td>
<td>74</td>
</tr>
<tr>
<td>MapControl</td>
<td>75</td>
</tr>
<tr>
<td>Map Tools</td>
<td>75</td>
</tr>
<tr>
<td>State Management and Pooling Capabilities</td>
<td>76</td>
</tr>
<tr>
<td>MapXtreme Web Controls and Tools</td>
<td>77</td>
</tr>
<tr>
<td>Description of Web Controls and Tools</td>
<td>78</td>
</tr>
<tr>
<td>Web Control Architecture</td>
<td>80</td>
</tr>
<tr>
<td>Map Tools Architecture</td>
<td>81</td>
</tr>
<tr>
<td>How a Map Tool Works</td>
<td>83</td>
</tr>
<tr>
<td>Using the MapXtreme Web Controls</td>
<td>84</td>
</tr>
<tr>
<td>Managing Your Web Controls</td>
<td>85</td>
</tr>
<tr>
<td>Event Handling</td>
<td>85</td>
</tr>
<tr>
<td>Error Management</td>
<td>85</td>
</tr>
<tr>
<td>State Management</td>
<td>85</td>
</tr>
<tr>
<td>Creating a Custom Tool</td>
<td>86</td>
</tr>
<tr>
<td>Using and Distributing Custom Web Controls</td>
<td>88</td>
</tr>
<tr>
<td>Creating a Web Assembly</td>
<td>88</td>
</tr>
<tr>
<td>ASP.NET AJAX and MapXtreme Web Applications</td>
<td>89</td>
</tr>
<tr>
<td>Adding ASP.NET AJAX Controls to a MapXtreme Web Application</td>
<td>89</td>
</tr>
<tr>
<td>HTML/XHTML Validation Issues</td>
<td>91</td>
</tr>
<tr>
<td>Migrating from pre-v6.5 Web Controls to 2005/2008 Web Controls</td>
<td>91</td>
</tr>
<tr>
<td>Loading Data</td>
<td>92</td>
</tr>
<tr>
<td>Replacing Controls</td>
<td>92</td>
</tr>
<tr>
<td>State and Event Management</td>
<td>92</td>
</tr>
</tbody>
</table>
**Table of Contents**

Specialized Topics for Web Controls ........................................... 92
  Using Web Controls in Frames .............................................. 93
  Using the MapControl in Table Cells ..................................... 93
  Web Control Localization .................................................... 94

Chapter 6: Understanding State Management ................................ 95
  Overview .............................................................................. 96
  Terminology .......................................................................... 96
  What is State Management? .................................................. 98
    What State Management Options are Available? ....................... 98
    Questions to Ask Before Writing Your Application ................... 98
  InProc Development Model .................................................. 100
    Pros and Cons of the InProc Development Model ..................... 101
    InProc Management: A Walk-Through ................................... 101
    Configuring an Application to Use the InProc Development Model . 102
    Using the MapXtreme Template with the InProc Development Model . 103
  State Management For Pooled Objects .................................... 103
    What is Pooling? ............................................................. 103
    Pros and Cons of Pooling ............................................... 104
    Saving State for Pooled Applications .................................. 104
    Manual State Management: A Walk-Through .......................... 106
    Configuring a Pooled Application to Use Manual State Management . 107

A Detailed Look at Manual State Management ................................ 108
  Overview of the Thematics Sample .......................................... 108
  Application Settings ......................................................... 109
  Implementing a StateManager .............................................. 110
  Serializing MapXtreme Objects in the Proper Order ................... 111
  Automatically Deserializing MapXtreme Objects ....................... 112
  Handling Initial Requests ................................................. 112
  Handling Subsequent Requests ............................................ 113

A Closer Look at the MapXtreme Session .................................... 114
  Configuring Microsoft COM+ Object Pooling ......................... 114

Chapter 7: Web Application Performance .................................... 117
  Web Application Performance ............................................ 118
    How Many Users? ......................................................... 118
    Performance Tuning ...................................................... 118
    IIS Server Settings ..................................................... 120
<table>
<thead>
<tr>
<th>Chapter 8: Desktop Applications, Controls, Dialogs, and Tools</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning a Desktop Application</td>
<td>124</td>
</tr>
<tr>
<td>Best Practices for Desktop Applications</td>
<td>124</td>
</tr>
<tr>
<td>MapXtreme and COM</td>
<td>124</td>
</tr>
<tr>
<td>Sample Applications and Project Templates</td>
<td>125</td>
</tr>
<tr>
<td>MapInfo.Windows.Controls Namespace</td>
<td>126</td>
</tr>
<tr>
<td>Key Controls to Use in Desktop Applications</td>
<td>127</td>
</tr>
<tr>
<td>MapControl.</td>
<td>127</td>
</tr>
<tr>
<td>The MapToolBar</td>
<td>128</td>
</tr>
<tr>
<td>Layer Control</td>
<td>130</td>
</tr>
<tr>
<td>MapInfo.Windows/dialogs Namespace</td>
<td>132</td>
</tr>
<tr>
<td>Customizing Controls and Dialog Boxes</td>
<td>137</td>
</tr>
<tr>
<td>Overview of the MapInfo.Tools Namespace</td>
<td>137</td>
</tr>
<tr>
<td>MapXtreme Desktop Tools API</td>
<td>139</td>
</tr>
<tr>
<td>View Tools</td>
<td>140</td>
</tr>
<tr>
<td>Select Tools</td>
<td>140</td>
</tr>
<tr>
<td>Add Tools</td>
<td>141</td>
</tr>
<tr>
<td>Custom Tools</td>
<td>141</td>
</tr>
<tr>
<td>Shape Tools</td>
<td>142</td>
</tr>
<tr>
<td>Using InfoTips</td>
<td>143</td>
</tr>
<tr>
<td>Customizing Tools</td>
<td>144</td>
</tr>
<tr>
<td>Tool Events</td>
<td>145</td>
</tr>
<tr>
<td>Editing a FeatureGeometry with the Select Tool</td>
<td>146</td>
</tr>
<tr>
<td>Reshaping a Feature</td>
<td>146</td>
</tr>
<tr>
<td>Adding Nodes</td>
<td>147</td>
</tr>
<tr>
<td>Reshaping and Adding Nodes Programmatically</td>
<td>148</td>
</tr>
<tr>
<td>Chapter 9: Working with Core MapXtreme Classes</td>
<td>151</td>
</tr>
<tr>
<td>ISession Interface</td>
<td>152</td>
</tr>
<tr>
<td>Session Management</td>
<td>152</td>
</tr>
<tr>
<td>Using Session.Dispose Method</td>
<td>153</td>
</tr>
<tr>
<td>Serialization and Persistence</td>
<td>154</td>
</tr>
<tr>
<td>Serialization</td>
<td>154</td>
</tr>
<tr>
<td>Persistence</td>
<td>155</td>
</tr>
<tr>
<td>Selection Class</td>
<td>156</td>
</tr>
<tr>
<td>Using Selection Properties</td>
<td>156</td>
</tr>
<tr>
<td>Selection Highlighting and Exporting</td>
<td>157</td>
</tr>
<tr>
<td>SelectionChangedEventArgs</td>
<td>157</td>
</tr>
<tr>
<td>ISerializable Interface on Selection and Selections Classes</td>
<td>157</td>
</tr>
<tr>
<td>Selection Code Examples</td>
<td>157</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Selecting Features Within Another Feature</td>
<td>158</td>
</tr>
<tr>
<td>Checking a Table for Selections</td>
<td>158</td>
</tr>
<tr>
<td>Returning All Columns From a Table</td>
<td>158</td>
</tr>
<tr>
<td>Changing the Map View Following a Selection</td>
<td>159</td>
</tr>
<tr>
<td><strong>Event Arguments</strong></td>
<td>159</td>
</tr>
<tr>
<td><strong>Exceptions</strong></td>
<td>159</td>
</tr>
</tbody>
</table>

**Chapter 10: Working with Data** ........................................... 161

<table>
<thead>
<tr>
<th>Overview of MapInfo.Data Namespace</th>
<th>162</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Catalog and Tables</strong></td>
<td>163</td>
</tr>
<tr>
<td>Tables</td>
<td>163</td>
</tr>
<tr>
<td>Catalog</td>
<td>166</td>
</tr>
<tr>
<td><strong>Supported Table Types</strong></td>
<td>167</td>
</tr>
<tr>
<td><strong>Working with Catalog and Tables</strong></td>
<td>169</td>
</tr>
<tr>
<td>Locating Open Tables</td>
<td>170</td>
</tr>
<tr>
<td>Closing a Table</td>
<td>170</td>
</tr>
<tr>
<td>Packing a Table</td>
<td>170</td>
</tr>
<tr>
<td>Listening to Table and Catalog Events</td>
<td>171</td>
</tr>
<tr>
<td><strong>Table Metadata (TableInfo)</strong></td>
<td>172</td>
</tr>
<tr>
<td>Examining TAB File Metadata</td>
<td>172</td>
</tr>
<tr>
<td>Creating a New Table</td>
<td>173</td>
</tr>
<tr>
<td>Adding Expression Columns to a Table</td>
<td>175</td>
</tr>
<tr>
<td>Data Sources</td>
<td>176</td>
</tr>
<tr>
<td>Choosing the Correct Data Source</td>
<td>177</td>
</tr>
<tr>
<td>Methods for Accessing Data</td>
<td>177</td>
</tr>
<tr>
<td>Using an ADO.NET Data Provider</td>
<td>178</td>
</tr>
<tr>
<td>Data Binding</td>
<td>182</td>
</tr>
<tr>
<td>Making Tables Mappable</td>
<td>185</td>
</tr>
<tr>
<td><strong>MapInfo ADO.NET Data Provider</strong></td>
<td>187</td>
</tr>
<tr>
<td>MIConnection</td>
<td>187</td>
</tr>
<tr>
<td>MICommand</td>
<td>188</td>
</tr>
<tr>
<td>MIDataReader</td>
<td>190</td>
</tr>
<tr>
<td>MapInfo SQL</td>
<td>191</td>
</tr>
<tr>
<td><strong>Features and Feature Collections</strong></td>
<td>191</td>
</tr>
<tr>
<td>Feature</td>
<td>191</td>
</tr>
<tr>
<td>Feature Collections</td>
<td>192</td>
</tr>
<tr>
<td>Searching for Features</td>
<td>193</td>
</tr>
<tr>
<td>Catalog Search Methods</td>
<td>193</td>
</tr>
<tr>
<td>SearchInfo and SearchInfoFactory</td>
<td>194</td>
</tr>
<tr>
<td><strong>Analyzing Data</strong></td>
<td>198</td>
</tr>
<tr>
<td>Improving Data Access Performance</td>
<td>200</td>
</tr>
<tr>
<td>Chapter 11: Creating Expressions</td>
<td>201</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Expressions Overview</td>
<td>202</td>
</tr>
<tr>
<td>Creating Expressions</td>
<td>202</td>
</tr>
<tr>
<td>Where Clause – Boolean Expressions</td>
<td>203</td>
</tr>
<tr>
<td>Functions In Expressions</td>
<td>203</td>
</tr>
<tr>
<td>Expression Examples</td>
<td>204</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 12: Accessing Data from a DBMS</th>
<th>209</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessing Remote Spatial Data</td>
<td>210</td>
</tr>
<tr>
<td>Accessing Remote Tables Through a .TAB File</td>
<td>210</td>
</tr>
<tr>
<td>Accessing Remote Tables Without a .TAB File</td>
<td>210</td>
</tr>
<tr>
<td>Mapping DBMS Data with X/Y Columns</td>
<td>211</td>
</tr>
<tr>
<td>Accessing Data from Oracle</td>
<td>211</td>
</tr>
<tr>
<td>Geometry Conversion</td>
<td>211</td>
</tr>
<tr>
<td>Multi Dimension Geometry Support (Greater than two Dimensions)</td>
<td>212</td>
</tr>
<tr>
<td>SDO_GEOMETRY Arc and Circle Translation</td>
<td>212</td>
</tr>
<tr>
<td>Visualization of Non-translatable Oracle Objects</td>
<td>212</td>
</tr>
<tr>
<td>Centroid Support</td>
<td>213</td>
</tr>
<tr>
<td>Additional Information</td>
<td>213</td>
</tr>
<tr>
<td>Oracle Spatial Reference Support (SRID)</td>
<td>213</td>
</tr>
<tr>
<td>OCI Connection Dialog</td>
<td>214</td>
</tr>
<tr>
<td>DBMS Connection String Format</td>
<td>214</td>
</tr>
<tr>
<td>ODBC Connection String Format</td>
<td>214</td>
</tr>
<tr>
<td>ODBC Layers and Pooling in Web Applications</td>
<td>215</td>
</tr>
<tr>
<td>Oracle Spatial Connection String Format</td>
<td>216</td>
</tr>
<tr>
<td>Sample Connection Strings</td>
<td>216</td>
</tr>
<tr>
<td>Defining Mappable Tables in Server Table Queries</td>
<td>216</td>
</tr>
<tr>
<td>The Geometry Column</td>
<td>217</td>
</tr>
<tr>
<td>The Key Column(s)</td>
<td>217</td>
</tr>
<tr>
<td>Accessing Attribute Data</td>
<td>218</td>
</tr>
<tr>
<td>Performance Issues</td>
<td>219</td>
</tr>
<tr>
<td>Working with the Cache</td>
<td>219</td>
</tr>
<tr>
<td>What Is the Cache?</td>
<td>219</td>
</tr>
<tr>
<td>How the Cache Works</td>
<td>220</td>
</tr>
<tr>
<td>The TableInfoServer Object and the CacheSettings Property</td>
<td>220</td>
</tr>
<tr>
<td>The MapInfo MapCatalog</td>
<td>221</td>
</tr>
<tr>
<td>Loading Spatial Data to DBMS</td>
<td>222</td>
</tr>
<tr>
<td>Manually Creating a MapInfo MapCatalog</td>
<td>222</td>
</tr>
<tr>
<td>Adding Rows to the Map Catalog</td>
<td>223</td>
</tr>
</tbody>
</table>
# Table of Contents

**Per-Record Styles**  ................................................. 225  
Symbol, Pen, Brush Clause Syntax  ........................................... 226  
Text Objects Limitation  .................................................. 227  
**Troubleshooting**  ....................................................... 227  

**Chapter 13: Adding Mapping Capability to Your Applications**  229  

**Introduction to the Mapping Namespace**  .................................. 230  
**Base Mapping Classes**  .................................................. 230  
MapExport  ........................................................................ 230  
Map  .................................................................................. 231  
MapFactory  ........................................................................ 231  
MapLoader  ........................................................................... 232  
MapViewList, MapView  ......................................................... 232  
MapControl  ......................................................................... 232  
**Layers**  ........................................................................... 233  
FeatureLayer  ........................................................................ 233  
Layers  ................................................................................ 233  
MapLayer  ............................................................................. 234  
UserDrawLayer  ....................................................................... 234  
ObjectThemeLayer  ................................................................. 234  
GroupLayer  ........................................................................... 234  
Layer Filters  ........................................................................ 234  
IVisibilityConstraint.  .............................................................. 234  
Code Example: Animation Layer  .............................................. 234  
**Labels**  ........................................................................... 237  
LabelLayer  ................................................................. 237  
LabelSource  ................................................................. 237  
LabelModifier  ............................................................... 238  
ILabelSourceFilter  ............................................................. 238  
LabelProperties  ............................................................... 238  
Generating Labels  .............................................................. 238  
Label Priorities  ................................................................. 239  
Label Layer Selectability  ....................................................... 239  
Code Example: Creating a LabelLayer  ................................. 240  
**Adornments**  ................................................................... 240  
Legends  ............................................................................. 241  
ScaleBar Adornment  ............................................................ 241  
Title Adornment  ................................................................. 242
# Table of Contents

**Developer Guide**

<table>
<thead>
<tr>
<th>Feature Style Modifiers</th>
<th>242</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureStyleModifier</td>
<td>243</td>
</tr>
<tr>
<td>FeatureStyleModifiers</td>
<td>243</td>
</tr>
<tr>
<td>FeatureOverrideStyleModifier</td>
<td>243</td>
</tr>
</tbody>
</table>

**Printing Your Map**                        | 244 |

**Chapter 14: Finding Locations**            | 245 |

**Functional Overview of Find**              | 246 |
| The Find Process                           | 246 |
| Matching Address Numbers                   | 248 |
| Matching with a Refining Boundary Table    | 248 |
| Find Results                               | 248 |

**Overview of the Data.Find Namespace**     | 248 |
| Find                                      | 249 |
| .FindAddressRange                         | 250 |
| FindCloseMatch                            | 251 |
| FindResult                                | 252 |

**Fine Tuning the Find Process**             | 254 |
| Editing the Mapinfow.abb File              | 255 |

**Chapter 15: Using Themes and Legends**    | 261 |

**Thematics Overview**                      | 262 |
| Mapping.Thematics Namespace                | 262 |
| Modifier Themes                            | 262 |
| Object Themes                             | 263 |

**GraduatedSymbolTheme**                     | 263 |
| When To Use a Graduated Symbol Theme       | 264 |

**PieTheme**                                | 265 |
| When To Use a Pie Theme                    | 265 |

**BarTheme**                                | 266 |
| When To Use a Bar Theme                    | 266 |

**RangedTheme**                             | 266 |
| When To Use a Ranged Theme                 | 267 |
| Types of Ranged Values                     | 267 |

**RangedLabelTheme**                        | 269 |
| When To Use a RangedLabelTheme Class       | 269 |

**IndividualValueTheme**                    | 269 |
| When To Use an IndividualValueTheme Class  | 269 |

**IndividualValueLabelTheme**               | 270 |
| When To Use an IndividualValueLabelTheme Class | 270 |
### Table of Contents

**DotDensityTheme** .......................................................... 271
   When To Use a DotDensityTheme Class .................................. 271
   Bivariate Thematic Maps .................................................. 271

**Legends Overview** ......................................................... 272
   Theme Legends ............................................................... 272
   Cartographic Legends ...................................................... 273
   Formatting a Legend ........................................................ 274

**Chapter 16: Stylizing Your Maps** .................................. 277

**Overview of the MapInfo.Styles Namespace** ......................... 278
   StyleFactory ............................................................... 279

**Style Descriptions** ...................................................... 279
   AreaStyle ................................................................. 279
   BitmapPointStyle .......................................................... 279
   CompositeStyle ........................................................... 280
   SimpleInterior ............................................................. 280
   Font ........................................................................ 280
   FontPointStyle ............................................................. 280
   GridStyle ................................................................. 280
   RasterStyle ............................................................... 280
   Hillshade ................................................................. 281
   Inflection ................................................................. 281
   SimpleLineStyle ........................................................... 281
   BasePointStyle ............................................................. 281
   BaseLineStyle ............................................................. 281
   BaselInterior ............................................................... 281
   StockStyles ............................................................... 281
   TextStyle ................................................................. 282
   SimpleVectorPointStyle .................................................. 282

**Pre-defined Styles and the StyleRepository Class** .................. 282
   StyleRepository Class .................................................... 282

**Using Styles** ............................................................. 283
   Styles and Layer Control ................................................. 283
   Creating a Custom Bitmap Style ......................................... 283

**Overriding Styles** ......................................................... 284
   FeatureOverrideStyleModifiers ......................................... 284
Chapter 17: Spatial Objects and Coordinate Systems ........................................... 287

Introduction to MapInfo.Geometry Namespace .............................................. 288

Geometries ................................................................. 288
   Geometry Objects .................................................... 289
   FeatureGeometry Objects ......................................... 289
   Geometry Objects .................................................... 293

Including Your FeatureGeometry in a Map ..................................................... 294

Checking for Points in Polygons ................................................................. 296

Coordinate Systems .............................................................. 297
   Creating a CoordSys Object ........................................ 297
   Changing the Coordinate System of a Geometry Object .......... 298
   Determining the Coordinate System of a Map in MapControl .................. 299
   Adding Coordinate Systems to MapXtreme .......................... 299

National Transformation v. 2 (NTv2) ......................................................... 302

Chapter 18: Working with Rasters and Grids .................................................... 307

Overview of the MapInfo.Raster Namespace .................................................. 308

Raster Images ................................................................. 308
   Raster Classes ....................................................... 309
   Raster Images and Coordinate Systems .............................. 310
   Raster Reprojection .................................................. 310
   Raster Image Limitations ............................................ 310
   Code Sample: Adding a Raster Image to a Map .................... 310

Raster Handlers ................................................................. 311

Configuring Custom Raster Handlers ......................................................... 312

Grid Images ................................................................. 313
   Grid Classes .......................................................... 314
   Code Sample: Adding a Grid Image to a Map ...................... 314
   Code Sample: Retrieving Data from a Grid Map .................. 314

Chapter 19: Geocoding ................................................................. 317

Overview of the MapInfo.Geocoding Namespace .......................................... 318

Main Geocoding Classes ................................................................. 318
   GeocodeRequest ....................................................... 319
   GeocodeResponse ..................................................... 319
   GeocodeClientFactory ............................................... 319
   GeocodingConstraints ............................................... 320
   AddressCandidates ................................................... 320
   BaseGeocodeMatchCode and GeocodeMatchCode .................. 320
   CandidateAddress .................................................... 320
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understanding the Geocoding Model</td>
<td>320</td>
</tr>
<tr>
<td>Geocoding Trade-offs</td>
<td>320</td>
</tr>
<tr>
<td>A Few Words About Addresses</td>
<td>321</td>
</tr>
<tr>
<td>What are Custom User Dictionaries?</td>
<td>322</td>
</tr>
<tr>
<td>What is Geocoding World?</td>
<td>322</td>
</tr>
<tr>
<td>Geocoding a Location</td>
<td>322</td>
</tr>
<tr>
<td>Street Address Geocoding</td>
<td>323</td>
</tr>
<tr>
<td>Street Intersection Geocoding</td>
<td>324</td>
</tr>
<tr>
<td>Postal Code Geocoding</td>
<td>325</td>
</tr>
<tr>
<td>Gazetteer Type Geocoding</td>
<td>325</td>
</tr>
<tr>
<td>Batch Geocoding</td>
<td>325</td>
</tr>
<tr>
<td>Using Constraints for Accurate Geocoding</td>
<td>325</td>
</tr>
<tr>
<td>What are the Match Constraints?</td>
<td>326</td>
</tr>
<tr>
<td>Impact of Relaxesing Match Constraints</td>
<td>328</td>
</tr>
<tr>
<td>Understanding Accuracy for Close Matches</td>
<td>328</td>
</tr>
<tr>
<td>Single Close Match (S Category)</td>
<td>329</td>
</tr>
<tr>
<td>Best Match from Multiple Candidates (M Category)</td>
<td>330</td>
</tr>
<tr>
<td>Postal Code Centroid Matches (Z Category)</td>
<td>330</td>
</tr>
<tr>
<td>Geographic Centroid Matches (G category)</td>
<td>330</td>
</tr>
<tr>
<td>Non-Match Codes</td>
<td>331</td>
</tr>
<tr>
<td>Chapter 20: Routing</td>
<td>333</td>
</tr>
<tr>
<td>Overview of MapInfo.Routing Namespace</td>
<td>334</td>
</tr>
<tr>
<td>Main Routing Classes</td>
<td>334</td>
</tr>
<tr>
<td>Calculating Routes</td>
<td>335</td>
</tr>
<tr>
<td>Point-to-Point Routing</td>
<td>335</td>
</tr>
<tr>
<td>Multi-Point Routing</td>
<td>336</td>
</tr>
<tr>
<td>Matrix Routing</td>
<td>337</td>
</tr>
<tr>
<td>Advanced Route Options</td>
<td>338</td>
</tr>
<tr>
<td>Routing Preferences</td>
<td>338</td>
</tr>
<tr>
<td>Driving Directions</td>
<td>339</td>
</tr>
<tr>
<td>Route Geometry</td>
<td>340</td>
</tr>
<tr>
<td>Avoiding Points, Features, and Segments</td>
<td>340</td>
</tr>
<tr>
<td>Time-Based Routing</td>
<td>341</td>
</tr>
<tr>
<td>Iso Routing (Drive-Time and Drive-Distance)</td>
<td>342</td>
</tr>
<tr>
<td>Creating an IsoChrone (Drive-Time)</td>
<td>343</td>
</tr>
<tr>
<td>Creating an IsoDistance (Drive-Distance)</td>
<td>345</td>
</tr>
<tr>
<td>Updating a Request Using Routing Data</td>
<td>347</td>
</tr>
<tr>
<td>Returning Segment Information</td>
<td>347</td>
</tr>
<tr>
<td>Transient Updates</td>
<td>348</td>
</tr>
</tbody>
</table>
Chapter 21: Web Map Service .......................................................... 353
  Introduction to MapXtreme's Web Map Service .......................... 354
  Understanding WMS Operations .................................................. 354
  Setting up a WMS Server ......................................................... 356
    Step 1: Create a Web.config File .......................................... 356
    Step 2: Create a Valid WMS Configuration File for Hosted Data ... 357
    Step 3: Configuring and Testing the WMS Server ....................... 359
  Configuring Layer Information for a WMS Server ....................... 362
    Requesting a WMS Layer ..................................................... 364

Chapter 22: Web Feature Service .................................................. 367
  Web Feature Service ............................................................... 368
  Understanding WFS Server Operations ..................................... 368
  Configuring a WFS Server ....................................................... 373
    Step 1: Create a Web.config File .......................................... 373
    Step 2: Create a Valid WFS Configuration File for Hosted Features 374
    Step 3: Configuring and Testing the WFS Server ....................... 376
  Using the MapXtreme WFS Client Programmatically ..................... 379
    Using Filters in WFS Queries .............................................. 379
    Creating a Map Layer from a WFS Response ............................ 382

Chapter 23: Workspace Manager .................................................. 385
  Features of the Workspace Manager ....................................... 386
  Workspace Format and Contents ............................................. 387
  Workspace Manager Menu Commands ...................................... 387
    File Menu Commands .......................................................... 387
    Map Menu Commands .......................................................... 389
    Tools Menu Commands ........................................................ 392
  Layer Control ................................................................. 394
    Layer Control Tools .......................................................... 394
    Layer Tree ........................................................................... 395
    Layer Control Tabs ............................................................. 396
    Layer Settings ..................................................................... 399
    Theme Layer Settings .......................................................... 401
    Label Layer Settings ............................................................ 401
    Graticule Layer Settings ...................................................... 404
  Using Workspace Manager Features ........................................ 406
    Enhanced Rendering with GDI+ Translucency and Anti-Aliasing ...... 406
    Creating Translucent Effects ............................................... 408
    Curved Labels ..................................................................... 411
    Graticule Layers ................................................................... 414
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapInfo Shields</td>
<td>506</td>
</tr>
<tr>
<td>MapInfo Real Estate</td>
<td>507</td>
</tr>
<tr>
<td>Map Symbols</td>
<td>507</td>
</tr>
<tr>
<td>MapInfo Symbols</td>
<td>507</td>
</tr>
<tr>
<td>MapInfo Transportation</td>
<td>508</td>
</tr>
<tr>
<td>MapInfo Weather</td>
<td>508</td>
</tr>
<tr>
<td>Custom Symbols</td>
<td>508</td>
</tr>
<tr>
<td>Appendix G: Defining the MapInfo Codespace</td>
<td>511</td>
</tr>
<tr>
<td>Defining the MapInfo Codespace</td>
<td>512</td>
</tr>
<tr>
<td>Appendix H: Elements of a Coordinate System</td>
<td>517</td>
</tr>
<tr>
<td>Projections and Their Parameters</td>
<td>518</td>
</tr>
<tr>
<td>Projection</td>
<td>519</td>
</tr>
<tr>
<td>Projection Datums</td>
<td>522</td>
</tr>
<tr>
<td>Units</td>
<td>531</td>
</tr>
<tr>
<td>Coordinate System Origin</td>
<td>532</td>
</tr>
<tr>
<td>Datum Conversion</td>
<td>533</td>
</tr>
<tr>
<td>Custom Datums</td>
<td>534</td>
</tr>
<tr>
<td>Defining Custom Datums</td>
<td>534</td>
</tr>
<tr>
<td>Information on Coordinate Systems and Projections</td>
<td>539</td>
</tr>
<tr>
<td>Appendix I: User-Defined Metadata</td>
<td>541</td>
</tr>
<tr>
<td>Metadata and the MapCatalog</td>
<td>542</td>
</tr>
<tr>
<td>User-Defined Metadata Support for TableInfoServer Queries</td>
<td>542</td>
</tr>
<tr>
<td>ColumnHints Property</td>
<td>542</td>
</tr>
<tr>
<td>Appendix J: Sample Data Copyrights</td>
<td>547</td>
</tr>
<tr>
<td>Appendix K: Glossary</td>
<td>557</td>
</tr>
<tr>
<td>Terms</td>
<td>558</td>
</tr>
<tr>
<td>Index</td>
<td>564</td>
</tr>
</tbody>
</table>
Welcome Developers to Pitney Bowes MapInfo’s latest offering in the world of .NET programming. In support of Microsoft’s .NET Framework for Windows, MapXtreme reflects a single object model for developing or extending mapping applications for the desktop, traditional client/server environments or the Web.

MapXtreme is an application development tool for organizations who recognize that data visualization and mapping can help you make better business decisions and manage assets and operations more effectively. MapXtreme is for organizations that need to incorporate location analysis or definition into desktop, client/server and web-based products. MapXtreme can be used as a powerful analysis toolkit to make critical business decisions such as optimal locations for sales offices, how to transport products most efficiently, and how to manage and protect assets. Developers can use MapXtreme to shorten their development time and improve performance, reliability, and security.

In this chapter:

- Overview of MapXtreme .............................. 20
- Migrating to MapXtreme .............................. 22
- Learning to Use MapXtreme .............................. 24
Overview of MapXtreme

MapXtreme is Pitney Bowes MapInfo’s premier Windows software development toolkit that allows .NET-experienced developers to create powerful location-enhanced desktop and client/server applications.

From this single SDK, you can develop applications using your favorite .NET programming language, share and reuse code between desktop and web deployments, access data from a wide variety of sources using standard protocols, and more.

This is all possible through MapXtreme’s object model, an API of 100 percent managed code that was developed on Microsoft’s .NET Framework. The Framework’s Common Language Runtime (CLR) provides the foundation that makes simplified development a reality.

The following components and features are included in MapXtreme:

- **Product framework**: The MapXtreme Object Model is built using the Microsoft .NET Framework 2.0. When developing MapXtreme applications you can choose to target .NET Framework versions 2.0, 3.0 or 3.5 and use any of their supported features. See MapXtreme Architecture in Chapter 4 on page 64 for more information.

- **Development Environment Tools**: A variety of templates, controls, sample code and tools help you develop Windows Forms and ASP.NET applications within Visual Studio. You can extend some of these components to provide more advanced functionality available through the object model. Two data management utilities are included for managing tables that you will use in your application (Geodictionary Manager) and manage workspaces for ease of use and portability (Workspace Manager). See Chapter 5: MapXtreme Web Controls and Tools and Chapter 8: Desktop Applications, Controls, Dialogs, and Tools.

- **Full Mapping and Analytical capabilities**: Map creation and display, data access, thematic mapping, raster and grid handling, object processing and presentation, and more.

- **Scalable Infrastructure**: Session object pooling and caching capabilities offer big performance gains for web applications. Maintain session and user information by saving information to MapXtreme’s XML-based workspace format. See Chapter 9: Working with Core MapXtreme Classes.

- **Runtime Deployment**: MapXtreme uses Windows Installer technology (merge modules) that developers can use to install or redistribute runtime components used in deployed applications. See Deploying Your Application in Chapter 2 on page 49.

- **Extensive Documentation**: Product documentation is at your fingertips as integrated components in the Visual Studio development environment. The MapXtreme Learning Resources page keeps you connected to all product resources, including what’s new and changed in this release. It’s available from the Start menu.

**Key Features**

MapXtreme is packed with features and conveniences to help you build your Windows Forms or ASP.NET Web applications efficiently. Regardless if you are making the map the cornerstone of your application or are adding some basic mapping functionality to support your existing application, the framework and tools you use are the same. Here is an overview of MapXtreme’s capabilities:

If you are new to Pitney Bowes MapInfo’s mapping products, be sure to also see Chapter 3: Mapping Concepts for additional help on the basics.
For developers upgrading to MapXtreme, see the Release Notes for what’s new and changed in the product. See also Migrating to MapXtreme on page 22 for a mapping of features in MapX to features in MapXtreme .NET.

<table>
<thead>
<tr>
<th>Feature *</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables, Layers, Features</td>
<td>Maps in MapXtreme consist of geographic features like point locations, boundaries and street networks. The feature information is stored in tables and display in the map as layers.</td>
</tr>
<tr>
<td>Data access</td>
<td>MapXtreme supports data from a wide variety of sources, including spatial and non-spatial RDBMS, MS Access, dBase and ASCII, as well as its own native type the MapInfo Table (.TAB). All data operations are carried out via the MapInfo.Data namespace. Operations include adding and removing tables, and inserting, updating and deleting records from a variety of data sources. .NET Dataset Provider support: allows any ADO.NET dataset provider to be treated as a table in MapInfo.Data. This will allow you to use external non-mappable data.</td>
</tr>
<tr>
<td>Web Services</td>
<td>MapXtreme provides clients and APIs for you to access several popular web services: geocoding, routing, WMS and WFS.</td>
</tr>
<tr>
<td>Selections and Searches</td>
<td>This common mapping operation allows you to find just the data that matches your criteria, by using attributes or spatial queries.</td>
</tr>
<tr>
<td>Thematic mapping</td>
<td>One of the most widely used ways of analyzing your data is to show the visual relationship and underlying data through theme maps. MapXtreme supports the creation and use of six themes: ranged, individual value, graduated symbol, dot density, and pie and bar charts.</td>
</tr>
<tr>
<td>Labeling</td>
<td>MapXtreme provides a sophisticated labeling capability for you to label features with names or other information (column data or expressions), and create ranged and individual value themes on the labels themselves to impart your message without relying on text alone.</td>
</tr>
<tr>
<td>Map Styling</td>
<td>Labeling is one form of map styling that you can control on your map in nearly infinite ways. Style also refers to the color, patterns, fonts, line styles and symbols of map features that is used in many areas of MapXtreme, including features, adornments (map titles), dialog boxes and text.</td>
</tr>
</tbody>
</table>
**Migrating to MapXtreme**

The following is a table that compares features and functionality from MapX and MapXtreme v3.0 with that of MapXtreme on .NET. As with any new architecture of a product, note that the equivalent may not be exact. Use the MapXtreme topics in the right column of this list to find further details elsewhere in this Developer Guide and in the online help and object model. A full list of the MapX object model and the equivalent functionality in MapXtreme is covered in *Migrating to MapXtreme* in Appendix A on page 423.

<table>
<thead>
<tr>
<th>MapX5.0 /MapXtreme3.0</th>
<th>MapXtreme</th>
</tr>
</thead>
</table>
| Map object            | Map class: holds collection of Layers.  
MapControl: way to view a map on a form.  
**MapInfo.Mapping namespace** |
| MapXBroker            | Session class: starting point for all MapXtreme-based applications.  
Related topics: MICommand, Catalog, Pooling  
**MapInfo.Engine namespace** |
| Layer-centric model   | Tablecentric model  
Related topics: Table metadata (TableInfo class), Feature class, Column class (MI_Geometry, MI_Style, MI_Key), MapInfo ADO.NET data provider.  
**MapInfo.Data namespace** |
<table>
<thead>
<tr>
<th>MapX5.0 /MapXtreme3.0</th>
<th>MapXtreme</th>
</tr>
</thead>
</table>
| Datasets, data binding | Add a temporary column to an Table using the Table.AddColumns() method.  
Related topic: Geodictionary Manager  
**MapInfo.Data namespace** |
| Geosets. Geoset Manager | Workspaces (.MWS): XML format. Geosets are supported.  
Related topics: Workspace Manager.  
**MapInfo.Persistence namespace** |
| Annotations | Adornments: a Legend, a Title, a Scalebar, or some other similar user-defined object in a single map.  
**MapInfo.Mapping namespace** |
| Thematic mapping | Same thematic map types. Themes are no longer layers.  
Related topics: ModifierThemes (graduated symbol, pie and bar themes), ObjectThemes (ranged, individual value, dot density themes).  
**MapInfo.Mapping.Thematic namespace** |
| Feature layer and Feature Collections | FeatureGeometry: all geometries are now objects. Includes point, multipoint, curve, multicurve, polygon, multipolygon ring.  
Geometries include rectangle, rounded rectangle, ellipse, legacy arcs, legacy text.  
Related topics: CoordSysFactory (registered coordinate systems), object processing (see FeatureProcessor, below)  
**MapInfo.Geometry namespace** |
| FeatureFactory | FeatureProcessor class: object processing Buffer, Combine, Intersection, ConvexHull.  
Related topic: Difference (formerly Erase) in FeatureGeometry class.  
**MapInfo.Geometry namespace** |
| Tools | Desktop tool SelectRegion can assign to mouse buttons and configure mouse wheel for zooming.  
**MapInfo.Tools namespace** |
Learning to Use MapXtreme

MapXtreme has a lot to offer beginning and experienced .NET developers alike. This section describes several support mechanisms we have created to get you up and running quickly and with minimal disruption in your development timetable.

Support Resources

MapInfo is committed to supporting new MapInfo developers as well as our long time customers. We provide a wide variety of tools to help you make the transition or get up and running quickly in the Visual Studio environment with the following resources.

<table>
<thead>
<tr>
<th>MapX5.0 /MapXtreme3.0</th>
<th>MapXtreme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label objects and Label collections</td>
<td>LabelLayer: allows the separate ordering of labels and layers. LabelSource: information from the data source that is used for labeling.</td>
</tr>
<tr>
<td>Selection object</td>
<td>Selection class: a multi-feature collection of lists of features that are tied to a table.</td>
</tr>
<tr>
<td>Spatial Server connectivity</td>
<td>MI ADO.NET data providers, MapInfo SQL language.</td>
</tr>
<tr>
<td>Style Object</td>
<td>Style class: new object model. Styles are now objects, not properties of other objects; information is stored in column MI_Style. Related topics: FeatureStyleModifiers, FeatureOverrideStyleModifiers in the Mapping namespace.</td>
</tr>
</tbody>
</table>

MapInfo.Raster namespace

MapInfo.Mapping namespace

MapInfo.Engine namespace

MapInfo.Data namespace

MapInfo.Styles namespace
Chapter 1: Introduction to MapXtreme

MapXtreme Learning Resources Page
Accessible from the Start Menu after you install MapXtreme, the Learning Resources Page brings together a wide variety of information about MapXtreme, including best development practices, code samples, tutorial web applications, links to all documentation and online resources on the MapInfo website, and much more. Refer to this page often to get comfortable with MapXtreme and as you continue to develop mapping applications that match your business needs.

Documents and Help Systems
This MapXtreme Developer Guide provides an overview of the MapXtreme development environment and namespaces. The Visual Studio-integrated Help System provides the more specific API-level information you need to use these tools to develop integrated Windows desktop and web-based applications with the MapInfo powerful mapping components. If you have comments regarding the documentation, you can send them to documentation@mapinfo.com.

Note The above-mentioned email address should not be used for questions specific to the software or clarification about subjects contained in the documentation. Please send those questions to Technical Support (see below).

If you are new to using or developing with MapInfo mapping products, be sure to see Chapter 3: Mapping Concepts.

To get the latest release information, be sure to download a copy of the MapXtreme Release Notes from the MapInfo website.

Technical Support
Pitney Bowes MapInfo offers unparalleled technical support for users of MapInfo software products. Our Technical Support department provides technical assistance to registered users of MapInfo software – so you don't need to be an expert in all aspects of our products in order to get results. See the MapInfo Web site at www.mapinfo.com for information on the tech support offerings.
Getting Started

This chapter provides all the information you need to install, configure, and deploy your first MapXtreme application.

In this chapter:

- Installation Requirements ........................................... 28
- Types of Installations..................................................... 30
- MapXtreme Licensing .................................................... 31
- Before You Install .......................................................... 36
- Installing MapXtreme in Your Environment ...................... 39
- Upgrading to MapXtreme ............................................... 39
- Creating Applications in Visual Studio ............................ 42
- Building ASP.NET Web Applications Without a Template .... 48
- Deploying Your Application ........................................... 49
Installation Requirements

The following sections describe MapXtreme system requirements for installation.

System Requirements

Pitney Bowes MapInfo has tested and supports MapXtreme on the following operating systems, .NET Framework, development environments, and web browsers.

| Operating Systems* | • Windows 2000 Professional SP 4  
|                    | • Windows 2000 Server SP 4  
|                    | • Windows 2000 Advanced Server SP 4  
|                    | • Windows XP Professional SP 2  
|                    | • Windows XP Professional Edition x64 †  
|                    | • Windows Server 2003 Standard Edition SP2  
|                    | • Windows Server 2003 Standard Edition x64  
|                    | • Windows Server 2003 Enterprise Edition SP2  
|                    | • Windows Server 2003 Enterprise Edition x64†  
|                    | • Windows Server 2003 Web Edition SP2  
|                    | • Windows Server 2008 Standard Edition (x86, x64)†  
|                    | • Windows Server 2008 Enterprise Edition(x86, x64)†  
|                    | • Windows Vista Ultimate (x86, x64)†  |

| Development Framework and IDE Support ‡ | • Microsoft .NET Framework 2.0 SP1  
|                                          | • Visual Studio 2005 SP1 (Professional and Standard Editions)  
|                                          | • Visual Studio 2008 (Professional and Standard Editions)  |

| Browsers | • Internet Explorer 6.0††  
|          | • Internet Explorer 7.0††  
|          | • Firefox 2.0  
|          | • Netscape 8.1.2  |

* 32-bit systems only  
† 32-bit emulation mode only via Microsoft's Windows-on-Windows (WoW64) technology  
‡ Recommended development environments (IDE). Others can be used, however, the MapXtreme installer will not integrate its templates, samples, and help system.  
†† The MapXtreme Learning Resources displays in Internet Explorer automatically, regardless of your default browser setting. This will not change your default browser setting.
The .NET Framework 2.0 must be installed on the development or deployment system before installing MapXtreme. We recommend that the development environment (IDE) appropriate for your framework be installed as well, so that the MapXtreme templates, samples and help system can be integrated.

**Note** MapXtreme does not support the Borland Delphi development framework, nor does it support the Cold Fusion or HAHT Web development frameworks.

The MapXtreme installer will check for the 2.0 Framework. If the 2.0 Framework is not on the computer, you will receive a message that you must install the Framework. The MapXtreme installation will exit. The Microsoft .NET Framework installer is provided on the MapXtreme product CD. To install the Framework, go to the MapXtreme CD Browser main page and choose Install>Microsoft .NET Framework.

MapXtreme does not support the Express Editions of Microsoft Visual Studio.

### Minimum System Requirements

<table>
<thead>
<tr>
<th>Memory</th>
<th>Windows Server 2003: 256 megabytes (MB) RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Windows Server 2003 Web: 256 MB RAM</td>
</tr>
<tr>
<td></td>
<td>Windows XP: 256 MB RAM</td>
</tr>
<tr>
<td></td>
<td>Windows 2000: 256 MB RAM</td>
</tr>
<tr>
<td></td>
<td>Windows 2000 Server: 256 MB RAM</td>
</tr>
<tr>
<td></td>
<td>Windows 2000 Advanced Server: 256 MB RAM</td>
</tr>
<tr>
<td></td>
<td>Windows Vista: 1 gigabyte (GB) RAM</td>
</tr>
<tr>
<td>Processor</td>
<td>600-MHz Pentium III-class processor</td>
</tr>
<tr>
<td>Video Card</td>
<td>Graphics card that supports at least 256 colors</td>
</tr>
</tbody>
</table>

### Additional Software Requirements

- MDAC 2.7 or later (MDAC 2.8 is located on the MapXtreme product CD.)
- IIS 5.0, IIS 6.0, or IIS 7.0 (for web application development and deployment)

### Supported Databases

Supported databases include:

- IBM Informix Dynamic Server 9.3
- IBM Informix Dynamic Server 9.4
- MapInfo SpatialWare® 4.5.1 for IBM Informix Dynamic Server 9.3
- MapInfo SpatialWare® 4.5.2 for IBM Informix Dynamic Server 9.4
- Microsoft Access 97
- Microsoft Access 2000
Types of Installations

MapXtreme provides two installation types: one for Development (SDK) and one for Deployment (Runtime). Each is selectable from the product CD Browser.

Development (SDK) Installations

The Development Installation installs the MapXtreme Software Development Kit (SDK) on your computer. Choose this installation to develop your desktop and web applications. Upon installation, this SDK is automatically integrated with the Microsoft Visual Studio IDE, and works in conjunction with the .NET Framework. The SDK provides C# and VB application templates for simplified development.

For instructions on how to install the SDK, see Installing MapXtreme in Your Environment on page 39. Instructions are also available from the Help buttons on the installation dialog boxes.

Deployment (Runtime) Installations

The Deployment installation option installs the Location Runtime Environment which lays down the MXTRuntimeNCP.exe (or MXTRuntimeSCP.exe for software-copy protected versions of MapXtreme). For instructions on installing the Runtime installer see Deploying With the Runtime Installer on page 49.

Side-By-Side Installations and Use

You may have more than one version of MapXtreme installed on your system at the same time. Each version of MapXtreme installs into its own directory.

You may also build a desktop and web application against an earlier version of MapXtreme and run it against a later version.

1. Assumes user accounts have associated schemas bearing the same name. This is necessary due to behavior in Microsoft SQL Server 2005 that allows for the separation of owners and schemas for enhanced security management.
You may not, however, run ASP.NET applications built with different versions of MapXtreme in the same process space, as in the case of IIS 5.0 on Windows XP and Windows 2000 operating systems. Each application would require its own version of MapXtreme to be loaded into IIS 5.0’s single process space.

IIS 6.0 on Windows Server 2003, however, can maintain multiple process spaces. To run your ASP.NET applications on the same computer, create an application pool for each version of MapXtreme and place the appropriate ASP.NET application in it. Restart IIS 6.0. When an application runs in its own process space, it will load the appropriate version of MapXtreme.

This does not affect desktop applications created with different versions on MapXtreme. Each desktop application always runs in its own process space.

**MapXtreme Licensing**

All MapXtreme installations require a license (.lic) file. Some licenses are provided in the installation, others must be obtained from Pitney Bowes MapInfo Corporation. The following topics describe the types of licenses and how you can obtain one from Pitney Bowes MapInfo.

**Types of Licenses**

Depending on your purpose and needs, three types of licenses are available for MapXtreme—SDK (developer), runtime (deployment), and trial licenses.

**SDK License**

The SDK (Software Development Kit) license (MapXtremeSDK.lic) is for developers to create desktop and web applications. Applications developed using this license will display a small watermark in the upper-left corner of the map window.

**Runtime License**

Runtime, or deployment, licenses are required for all runtime desktop and web installations.

- For desktop runtime deployments built using the NCP version of MapXtreme, the license (e.g., MapXtremeDesktop.lic) can be used on multiple installations.
- For desktop runtime deployments built using the SCP version of MapXtreme, a license (e.g., MapXtremeDesktopIntl.lic, for example) is required for every installation.
- For web runtime deployments built using the NCP or SCP version of MapXtreme, a license (e.g., MapXtremeWeb.lic) is required for every installation.

See [Runtime Licenses for Deployment on page 32](#) for information on packaging licenses in your distribution.
Trial License

The trial license (MapXtremeTrial.lic) is provided in the SDK and the Evaluation version of MapXtreme. It expires 60 days from the date of installation. Applications that are developed using the trial license display a watermark on the map that also indicates the number of days remaining for the trial license.

Obtaining a License

SDK Licenses for Development

Users of the non-copy protected Software Development Kit (NCP SDK) automatically receive a perpetual SDK license (MapXtremeSDK.lic), as well as a trial license when they install. With the perpetual license you can continue to use the SDK as long as you want. No other SDK license is necessary.

Users of the copy-protected SDK (SCP SDK) receive a trial license only. To use the SDK permanently, you must obtain a perpetual SDK license from PB MapInfo Corporation. To do this, run MILicenseNodeID.exe (found under Program Files\Common files\MapInfo\MapXtreme\6.x.x\License Tool) to generate a unique machine-specific hardware ID contained in a text file called MILicenseNodeID.txt. PB MapInfo Corporation uses this information to generate a custom license for your particular installation. For complete details on obtaining a license, see the Software Copy Protection Notice in the product box or installed under Program Files\MapInfo\MapXtreme\6.x.x\Documentation\PDF\SoftwareCopyProtectionNotice_SCP.pdf. Contact PB MapInfo Corporation (or its representative) at the appropriate email or fax number for your region as provided on the notice. When you receive your permanent license, copy it to the Program Files\Common Files\MapInfo\MapXtreme\6.x.x folder prior to first use of the deployed application.

Evaluation users will be required to purchase the product before receiving a permanent license.

Runtime Licenses for Deployment

MapXtreme requires that you include a runtime license in every deployment. To do so, you must either provide the license in your package or include a licensing tool for your customers to obtain the license themselves. Runtime licenses are obtained from Pitney Bowes MapInfo Corporation. This license type also removes all watermarks from the map.

The type of runtime license you need depends on what type of application you are building and the version of MapXtreme you use. Find your situation in the table below:

<table>
<thead>
<tr>
<th>Deployment Type</th>
<th>SDK Version of MapXtreme</th>
<th>Node-Locked?</th>
<th>How to Obtain Runtime License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop</td>
<td>Non-copy-protected (NCP)</td>
<td>No</td>
<td>Contact Customer Service for a runtime license for one or more seats.</td>
</tr>
<tr>
<td>Desktop</td>
<td>Copy-protected (SCP)</td>
<td>Yes</td>
<td>Generate a node ID file for each seat and submit to Customer Service.</td>
</tr>
</tbody>
</table>
To generate a node ID for the deployment machine, run the licensing tool MILicenseNodeID.exe and submit the resulting text file to Customer Service. The license tool is installed with the MapXtreme SDK into Program Files\Common Files\MapInfo\MapXtreme\6.x.x\License Tool. See the Software Copy Protection license application document in your MapXtreme package or under Program Files\MapInfo\MapXtreme\6.x.x\Documentation\PDF.

The license tool is included in the merge module MapInfoMXTConfig_6.x.x.msm. Your users will get this tool automatically when you include the MapInfoCoreEngine.msm in your distribution. See Steps for Deploying an Application Using the Runtime Installer in Chapter 2 on page 49.

Note  Generally, web licenses cannot be used for desktop deployments and vice versa. If it is an SCP version, then the license (web or desktop) is node-locked to the machine on which you are deploying.

### Locations of License Files

Use the following chart to determine where your license is located or where you should place it:

<table>
<thead>
<tr>
<th>Deployment Type</th>
<th>SDK Version of MapXtreme</th>
<th>Node-Locked?</th>
<th>How to Obtain Runtime License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>NCP</td>
<td>Yes</td>
<td>Generate a node ID file for each seat and submit to Customer Service.</td>
</tr>
<tr>
<td>Web</td>
<td>SCP</td>
<td>Yes</td>
<td>Generate a node ID file for each seat and submit to Customer Service.</td>
</tr>
</tbody>
</table>

To generate a node ID for the deployment machine, run the licensing tool MILicenseNodeID.exe and submit the resulting text file to Customer Service. The license tool is installed with the MapXtreme SDK into Program Files\Common Files\MapInfo\MapXtreme\6.x.x\License Tool. See the Software Copy Protection license application document in your MapXtreme package or under Program Files\MapInfo\MapXtreme\6.x.x\Documentation\PDF.

The license tool is included in the merge module MapInfoMXTConfig_6.x.x.msm. Your users will get this tool automatically when you include the MapInfoCoreEngine.msm in your distribution. See Steps for Deploying an Application Using the Runtime Installer in Chapter 2 on page 49.

Note  Generally, web licenses cannot be used for desktop deployments and vice versa. If it is an SCP version, then the license (web or desktop) is node-locked to the machine on which you are deploying.

### Locations of License Files

Use the following chart to determine where your license is located or where you should place it:

<table>
<thead>
<tr>
<th>License Type</th>
<th>License Name</th>
<th>Location of License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>MapXtremeSDK.lic</td>
<td>Program Files\Common Files\MapInfo\MapXtreme\6.x.x</td>
</tr>
<tr>
<td>Desktop NCP</td>
<td>MapXtremeDesktop.lic</td>
<td>In the same folder as the application executable to avoid potential conflicts with other applications</td>
</tr>
<tr>
<td>Desktop SCP</td>
<td>MapXtremeDesktopIntl.lic</td>
<td>In the same folder as the application executable to avoid potential conflicts with other applications</td>
</tr>
<tr>
<td>Web NCP</td>
<td>MapXtremeWeb.lic</td>
<td>Program Files\Common Files\MapInfo\MapXtreme\6.x.x</td>
</tr>
</tbody>
</table>
### Troubleshooting Licensing Problems

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Possible Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>“No valid license found for</td>
<td>Could occur when using Remote Desktop. Need a new license with a format that</td>
<td>Reset the server, place new license in the appropriate directory, then</td>
</tr>
<tr>
<td>MapXtreme 200x”</td>
<td>has Terminal Services activated.</td>
<td>restart IIS and the ASP.NET worker process.</td>
</tr>
<tr>
<td>Corrupt license file</td>
<td></td>
<td>Remove license from machine, regenerate license, place new license in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>appropriate directory, then restart IIS.</td>
</tr>
<tr>
<td>Permissions needed</td>
<td>Ensure the ASP.NET user account has Read/Write access to the Temp directory</td>
<td>Ensure the ASP.NET user account has Read/Write access to the Temp directory</td>
</tr>
<tr>
<td></td>
<td>and Framework directories (see Permissions to Temp Directory for Deployed Web</td>
<td>and Framework directories (see Permissions to Temp Directory for Deployed</td>
</tr>
<tr>
<td></td>
<td>Applications on page 54).</td>
<td>Web Applications on page 54).</td>
</tr>
<tr>
<td>COM+ Activation error</td>
<td>Usually a bad or corrupt license file. It stops the COM+ object from</td>
<td>Remove license from machine, regenerate license, place new license in the</td>
</tr>
<tr>
<td></td>
<td>initializing and therefore causes a low-level OS fault in COM+.</td>
<td>appropriate directory, and then restart IIS.</td>
</tr>
</tbody>
</table>
## Troubleshooting Licensing Problems (continued)

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Possible Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>A license exception occurs when using a trial version downloaded from the Pitney Bowes MapInfo website (System.ComponentModel.LicenseException' occurred in mapinfo.coreengine.dll.)</td>
<td>The MapXtreme trial license is invalid due to a detected system clock setback.</td>
<td>Install the trial on a different machine or purchase a valid license.</td>
</tr>
<tr>
<td>The merge modules (MSM) in the trial version are missing (\Common files\Merge Modules).</td>
<td>The merge modules are left out of the trial version because it is not meant to be deployed.</td>
<td>Purchase a runtime license.</td>
</tr>
<tr>
<td>The Volume Shadow Copy Service prevents licenses from being found.</td>
<td>Related to a Veritas backup. There is a possibility of a network change based on the backup software. The backup process is locking files and possibly not allowing access.</td>
<td>Check whether the backup process is running under the same ID as the Exponare process. If yes, then there is a possibility of Terminal Services being the problem. A new license will fix the problem.</td>
</tr>
<tr>
<td>A MapXtreme folder (from a previous) prevents valid licenses from being found.</td>
<td>An invalid license file may have been present in the old directory and would have caused the failure.</td>
<td>Find and delete old invalid licenses. Place new license in the appropriate directory.</td>
</tr>
<tr>
<td>The MapXtreme license distributed with Exponare unlocks some sites but fails to unlock others.</td>
<td>Multiple network cards are present in the system.</td>
<td>Obtain the correct license file. Place new license in the appropriate directory.</td>
</tr>
<tr>
<td>A license file will not unlock MapXtreme unless it is both in the MapXtreme\6.x.x folder and the web application folder.</td>
<td>Permission errors in how the server is run and which user ID is used.</td>
<td>Set the correct permissions (see Permissions to Temp Directory for Deployed Web Applications on page 54).</td>
</tr>
<tr>
<td>MapXtreme license files appear to expire after approximately 15 days.</td>
<td>Terminal Services</td>
<td>Make sure you are using the latest build of the license generator that accommodates terminal services.</td>
</tr>
</tbody>
</table>
Before You Install

Troubleshooting Licensing Problems (continued)

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Possible Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapXtreme license node ID changes and causes Exponare license to revert to trial license when a Virtual PC is moved.</td>
<td>Moving a Virtual PC changes the network node ID and thus invalidates the license.</td>
<td>Obtain a new license</td>
</tr>
<tr>
<td>Directly after installation and first use, user gets the error message “Your MapXtreme trial license has expired”.</td>
<td>Installing a generated runtime (deployment) license instead of using the NCP SDK license they already had.</td>
<td>Use the perpetual SDK license that comes with the NCP SDK CD.</td>
</tr>
</tbody>
</table>

Known Issues with Licensing

The following are known issues with MapXtreme licensing. Contact Technical Support to assist you in dealing with these problems (see Technical Support on page 25).

- Virus software strips content from license email, leaving Null in the license file.
- License generator occasionally creates corrupt license files.

Before You Install

The following are things to be aware of prior to installing MapXtreme.

Administrator Privileges

To install MapXtreme, you must be an Administrator on the machine or the current user must be a member of the group Administrator. This applies to both types of installation (SDK and Runtime).

Note For Windows Vista, regular administration rights are not sufficient because the installer must register the COM+ object. Vista users must right-click Setup.exe and choose: ‘Run as administrator’ to obtain system privileges for the installer.

Install .NET Framework and Visual Studio First

Before you install MapXtreme, be sure that you have the .NET Framework and the Visual Studio environment appropriate for the framework installed.

Note You may use a different development environment than Visual Studio, however, the templates, samples, and the online help system will not be integrated.
Windows 2000 Hotfix Package

If you are running Windows 2000 and intend to use MapXtreme in a web-based application, you must obtain the Microsoft Windows 2000 Post-Service Pack 4 (SP4) COM+ 1.0 rollup package 27. For more information, see the Microsoft KnowledgeBase Article - 822618 at http://support.microsoft.com/?id=822618.

IIS 7.0 on Windows Vista

Microsoft Internet Information Services 7.0 is included (although not necessarily installed) with Windows Vista. If the IIS 7.0 feature is installed on Vista, you must do the following to ensure a successful installation of MapXtreme:

   a. Go to Administrator Tools, right-click IIS and choose to “Run As Administrator”.
   b. Select Default Web Site.
   c. Double-click Authentication.
   d. Right-click Windows and Anonymous Authentication and choose Enabled.

2. Enable the IIS 6 Metabase and IIS 6 Configuration Compatibility web management tools.
   a. In Windows Vista, open the Control Panel and then click Programs and Features.
   b. In the right pane, click Turn Windows features on or off.
      The Windows Features dialog box opens.
   c. Select the Internet Information Services checkbox.
   d. Double-click (or expand) Web Management Tools, double-click IIS 6 Management Compatibility, and then select the IIS 6 Metabase and IIS 6 Configuration Compatibility check box.

To develop web applications in Visual Studio on Vista Systems, you will also need to enable the following IIS 7.0 features (either before or after you install MapXtreme):

• Internet Information Services: World Wide Web Services: Application Development Features
  • .NET Extensibility
  • ASP.NET
  • ISAPI Extensions
  • ISAPI Filters
• Internet Information Services: World Wide Web Services: Security
  • Windows Authentication

In order to run the MapXtreme web sample applications in IIS 7.0, you will also need to switch to IIS 7.0 to use “Classic .NET AppPool”.

If you create a new MapXtreme web application using the template through HTTP, you will need to go into IIS and explicitly set the Application Pool setting to “Classic .NET AppPool” for the new application. Otherwise, you will get an ambiguous error when trying to debug in Visual Studio.
Default Install Directory for MapXtreme Under Wow64

MapXtreme is a 32-bit application that runs on 64-bit systems through Microsoft’s Wow64 (Windows-on-Windows 64-bit) technology. Wow64 is an emulation layer that handles the differences between 32-bit applications (x86) and 64-bit operating systems (x64).

One effect of Wow64 is the location of the default installation directory for MapXtreme. 64-bit systems have two Program Files directories, one for native 64-bit installations (C:\Program Files), and one for 32-bit applications running under 64-bit emulation mode. For 32-bit applications, such as MapXtreme, the default installation directory is:

C:\Program Files(x86)\MapInfo\MapXtreme\6.x.x\n
where 6.x.x is the version of MapXtreme.

Both default paths are included in the Web.config files of the MapXtreme sample applications. If you have installed MapXtreme to another location, you must edit the Web.config files to point to that location for the samples to run properly.

Note For non-English US (ENU) installations of Windows, the default installation directory C:\Programmers\ is considered a custom install location by MapXtreme. You must edit the samples Web.config files to point to your install directory, as the example below shows.

<configuration>
  <appSettings>
    <add key="MapInfo.Engine.Session.Workspace"
      value="C:\Programmers\MapInfo\MapXtreme\6.x.x\Samples\Data\World.mws"
    />
  </appSettings>
</configuration>

Additional Installation Features

The MapXtreme CD provides online installation instructions to follow. You can also access the instructions via the Help button on the install dialogs during installation.

MapXtreme provides free sample data for a variety of world locations. To install the data, choose Install Sample Data from the CD Browser. You can control how much of the data you wish to install by choosing the Custom option. The Complete option (default) will install about 450 MB of world data sets under Program Files\MapInfo\MapXtreme\6.x.x\Samples\Data. For a listing of available data sets, see Appendix J: Sample Data Copyrights.

Note You do not need to run this data installer in order to use the sample applications that ship with MapXtreme. Basic sample data is automatically installed to the \Data folder for this purpose.

The MapXtreme CD Browser also provides a link to the PDF version of this Developer Guide.
Installing MapXtreme in Your Environment

To install MapXtreme:

1. Place the MapXtreme CD in the CD drive.
2. At the CD Browser main page, click Install. The Install Description page appears.
3. Choose either Development Install to install the SDK or Deployment Install to install the runtime version.
   
   **Note** The Deployment installation requires a runtime license. See Runtime Licenses for Deployment on page 32.

4. Choose Install SDK or Install Runtime. At the Welcome dialog box, click Next to proceed. For deployment installations skip to step 10.
   
   **Note** You may also review the installation instructions and install sample data from this page.

5. At the Installer Welcome dialog box, read the information in the panel and click Next to proceed.
6. Choose to accept the License Agreement. Click Next. The Customer Information dialog box appears.
7. At the Customer Information dialog box, enter your user name and company name in the appropriate fields and, if necessary, the product serial number which is located on the package. Click Next. The Setup Type dialog box appears.
8. At the Setup Type dialog box, select Complete or Custom. Choose Custom if you want the features to be installed or to install to a location other than the default (C:\Program Files\MapInfo\MapXtreme\6.x.x). Click Next. If you chose Complete proceed to step 10. If you chose Custom continue to step 9.
9. If you chose Custom in the previous step, select the components that you want to install, or click the Change button to specify a new installation path. Click Next.
10. At the Ready to Install the Program dialog box, click Install.
11. The Installing MapXtreme dialog box launches.
12. At the InstallShield Wizard Completed dialog box, uncheck the Launch Learning Resources checkbox if you do not wish to view the MapXtreme Learning Resources page, and then click Finish to leave the software installer.

The MapXtreme Learning Resources page is available anytime from the Windows Start menu.

Upgrading to MapXtreme

MapXtreme installs into its own directory using the form \<installdir>\MapInfo\MapXtreme\6.X.x.x, where X.x.x is the current release. It will not overwrite a previous version. This allows you to maintain side-by-side installations of different releases of the product.

A side-by-side installation requires that you copy your old license file to the location of the newest installation (Program Files\Common Files\MapInfo\MapXtreme\6.x.x.).

Note, when using a side by side install, you must close Visual Studio before opening a project of a different type. For web applications, you will also need to reset Internet Information Services (IIS). Use the iisreset.exe command in a console window or from the Start > Run menu option.
Upgrading to MapXtreme

Updating Existing Web Sites

In order to run a web site created with a previous release of MapXtreme, you must edit your application’s Web.config file to point to the new versions of the assemblies. The example below shows the four MapXtreme assemblies that must be included. One of the assemblies, MapInfo.CoreEngine.Wrapper, is new beginning with the MapXtreme v6.8.0 release. Any web application created with MapXtreme v6.7.1 or earlier, must include this assembly.

Following the illustration above, edit your Web.config file to include the version number and PublicKey token of the MapXtreme assemblies. To determine version number and PublicKey token, open the Run dialog from the Start menu and type assembly. The Global Assembly Cache (GAC) lists all assemblies installed on your system.

You can also reach the Assembly window by navigating from Start > All Programs > Control Panel > Administrative Tools > .NET Framework 2.0 Configuration and choose Manage the Assembly Cache.

When you are finished editing, save your Web.config file and rebuild your web site.
Updating Existing Desktop Applications

Desktop applications created with a previous release of MapXtreme can be recompiled to work in the current release. Controls may have to be re-added to the form.

First, however, you must redirect your application to use the new assemblies. For applications built using MapXtreme v6.7.1 or earlier, you must also add two assemblies: MapInfo.CoreEngine.Wrapper and MapInfo.CommonDialogExt.

Redirecting MapXtreme Assemblies To Newer Versions

Microsoft provides several mechanisms to redirect assemblies. For more detailed information, see Microsoft’s .NET Framework Developer Center.

Application Configuration File

It is recommended that you use an application configuration file to accomplish assembly redirection. The configuration file must be located in the same directory as the application and is named after the application. For example, the configuration file for `myApp.exe` must be named `myApp.exe.config`.

The application configuration file overrides settings in the publisher's policy file.

To redirect assemblies, you must identify the version numbers and PublicKey tokens for the current release and add them to your application configuration file. The version number is in the form X.x.x.x, for example, 6.8.0.532.

To find the information, open the Run dialog from the Start menu and type `assembly`. The Global Assembly Cache (GAC) lists all assemblies installed on your system. You can also reach the Assembly window by navigating from Start > All Programs > Control Panel > Administrative Tools > ..NET Framework 2.0 Configuration and choose Manage the Assembly Cache. This tool will also create an application configuration file, if one does not exist.

The illustration below highlights the two assemblies that must be added to desktop applications created with MapXtreme v6.7.1 or earlier.

Publisher Policy File

A publisher policy file containing redirection settings could be installed in the GAC with the assembly. However, this is not a supported MapInfo configuration.
**Machine Configuration File**

The Microsoft .NET Framework 2.0 Configuration tool can be used to update the machine configuration file, which is located in `\Microsoft.NET\Framework\v2.0.50727\Config` under your Windows installation directory.

Specifying redirection settings here will cause *all* applications referencing the assembly to use the upgraded version. Use this method of redirection carefully, since the machine configuration file overrides settings in both the application configuration file and the publisher’s policy file.

**Creating Applications in Visual Studio**

With MapXtreme, it's easy to add a map to your application. Visual Basic.NET and Visual C# project templates are provided that allow you to create simple mapping applications without writing any code.

Sample desktop and web applications are also provided for you to review, experiment with and adapt to your own situation. For a step-by-step tutorial on how to use these sample applications, see Appendix B: How to Create and Deploy a MapXtreme Application.

MapXtreme also ships with a collection of tutorial applications to help you understand how to include useful mapping functionality in a web application. Each tutorial application is accompanied by documentation that explains how the application was built. See the Learning Resources page for the documentation. The tutorial applications are included in a single Visual Studio solution called MapXtremeTutorials.sln, located in the `\Tutorials` folder of your MapXtreme installation (default location is `c:\Program Files\MapInfo\MapXtreme\6.x.x\Tutorials`, where 6.x.x is the release version).

The following procedure outlines the steps to make a simple desktop mapping application. For steps to create a web application, see [ASP.NET Web Applications on page 45](#).

**Map Applications**

- **Note** This example is specific to Visual Basic.NET. To create a Visual C# Map application substitute Visual C# for Visual Basic in the following steps.

1. From the Visual Studio File menu, select **New Project**. The New Project dialog box appears.
2. In the Project Types frame of the New Project dialog box, under the Visual Basic folder, click **MapXtreme 6.x.x**.
3. In the Templates frame of the New Project dialog box, select **MapXtreme 6.x.x MapForm Application**.
4. Choose an appropriate name and click **OK**. MapXtreme creates the application.
5. Under Solution Explorer double-click **MapForm.vb** and MapForm.vb [Design] appears.
6. On the View menu, click **Code** to examine the code that was written to create the application.

7. On the Debug menu, click **Start Debugging** to run the application.

8. Click the **Open Table** icon and load your data. The default location for sample data is **Program Files\MapInfo\MapXtreme\6.x.x\Samples\Data**.
9. Use the controls in the toolbar to manipulate the map as you would with any other Pitney Bowes MapInfo mapping application.

**ASP.NET Web Applications**

The following procedure outlines the steps to make a simple ASP.NET Web Application. For a more detailed discussion on the myriad considerations when planning a web application, see Chapter 5: Web Applications, Controls, and Tools.

This example is specific to Visual Basic.NET. To create a Visual C# ASP.NET map application substitute Visual C# for Visual Basic in the following steps.

**Note** If you are working on Windows Vista with IIS 7.0, you must first switch the Application Pool in IIS 7.0 to "Classic .NET AppPool".

1. From the Visual Studio File menu, click **New Web Site**. The New Web Site dialog box appears.

2. Select the **MapXtreme 6.x.x Web Application** template and click **OK**. MapXtreme creates the application.

3. Under Solution Explorer double-click **MapForm.aspx** to see the design view.
4. Click **Source** at the bottom of MapForm.aspx to view the HTML for the application.
5. On the Debug menu, click **Start Debugging** to run the application.

6. Use the controls in the toolbar to manipulate the map as you would with any other Pitney Bowes MapInfo mapping application.

**MapXtreme Controls**

Once you have created a basic application using one of our templates, enhancements are possible using a variety of MapXtreme controls provided in the Toolbox. For simple map applications built using Windows Forms, MapXtreme Windows Controls are available. Any of the controls found in the MapXtreme Windows Controls tab of the Visual Studio Toolbox can be added to your form.

For MapXtreme ASP.NET web applications, MapXtreme Web Controls are available. Any of the controls found in the MapXtreme Web Controls tab of the Toolbox, can be added to your form. See **MapXtreme Web Controls and Tools in Chapter 5 on page 77** for more information.
You may also build your ASP.NET map application without using the MapXtreme templates. For example, to create a Visual Basic web application:

1. Choose **File > New > Web Site** from the Visual Studio menu. The New Web Site dialog appears.
2. Under Visual Studio Installed Templates, choose the generic ASP.NET Web Site template and click **OK**.
3. From the MapXtreme Web Controls group in the Toolbox, choose a control and drag it onto the form. This will add three MapXtreme assemblies as references in your project: MapInfo.CoreEngine, MapInfo.CoreTypes, and MapInfo.WebControls. This will also update the Web.config file to add the assembly information as well as elements for httpModules.

4. In this situation, you will notice that the MapXtreme web controls and tools will display red X’s in the Designer, instead of their icons. To display the icons properly, copy the MapXtremeWebResources folder from one of the MapXtreme sample applications and paste it into your project where your Web.config and default.aspx files are located. Close and re-open the web page to see the icons.
Deploying Your Application

There are essentially two strategies for installing the MapXtreme components on the server machine where you wish to host your application: either 1) use the included runtime installer or, 2) create your own installer and add the appropriate merge modules (MSM).

The MapXtreme SDK ships with the MXTRunNCP.exe runtime installer for the NCP version. The SCP version of the SDK ships with MXTRunSCP.exe.

Deploying With the Runtime Installer

Use the runtime installer (MXTRunNCP.exe or MXTRunSCP.exe) as part of your custom installation process. This will install the MapXtreme assemblies and create the needed registry entries. It will also include the required .NET Framework v2.0. Runtime assemblies and files are included for both Web and Desktop applications. Additionally, you will need to install your runtime license file which you get from Pitney Bowes MapInfo Corporation or your reseller in the path Program Files\Common Files\MapInfo\MapXtreme\6.x.x. See Runtime Licenses for Deployment on page 32.

For web-based applications, using the runtime installer is the better strategy, as it also creates the necessary virtual directory. All of the runtime components are installed in their default locations. There are no user-configurable options. Another advantage of using the runtime installer is that you can apply patches without rebuilding your installer.

One disadvantage of using the runtime installer executable is that, depending on your application, unnecessary files may be installed. Additionally, if you expect to install more than one MapXtreme-enabled application, you will need to maintain your own reference counts when using the runtime installer. By using the MSMs directly with your own installer, references are maintained automatically.

Steps for Deploying an Application Using the Runtime Installer

1. From the MapXtreme CD, choose Install. The Install options page appears.
2. Choose Deployment Installation and choose Install Runtime Environment. The install wizard opens.
3. Follow the prompts to proceed with the installation.
   
   **Note** You must have Administrator permissions on the computer in order to run the installer. You also need IIS permissions to install to the Web server.

4. Deploy your Windows or Web application as you wish (for example, create a deployment project in Visual Studio and add your application.)
5. Install your runtime license file in the path C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x.
Deploying Your Application

If you wish to run the runtime installer from a command line, follow these instructions. The executable is located in the [CD root]\Install\InstallRuntime folder on the CD and in the folder Program Files\MapInfo\MapXtreme\6.x.x\Redistributables.

- To run the runtime installer with minimal UI, execute the runtime installer like this:
  MXTRunNCP.exe /v"/qb"
- To run the runtime installer silently, execute the runtime installer like this:
  MXTRunNCP.exe /v"/qn"

If you are using a software copy protected version of MapXtreme, the runtime executable is called MXTRunSCP.exe.

Deploying With Your Own Installer

Create your own installer using Windows Installer technology (MSI) and include the MapXtreme merge modules. A merge module (MSM file) is a single package that contains all files, resources, registry entries, and setup logic necessary to install a component. Merge modules cannot be installed alone; they must be assimilated into an MSI file. Use this strategy if you want to fine-tune exactly which components are installed, or if you want to create your own MSI-based installer.

One disadvantage of using the MSMs is that you will have to create your own patch or updated installer if Pitney Bowes MapInfo provides updates to this product.

An advantage of using the MSMs is that you control which components get installed and reference counts are maintained automatically. Assemblies of a particular version will be copied into the Global Assembly Cache (GAC) only once, and reference counts will be maintained for each application using those assemblies. If one application is subsequently removed, the reference count (which then decreases by one) will preserve the assemblies in the GAC. When the last application using those assemblies is removed, then the assemblies themselves will be removed.

Several developer tools are available to help you create an MSI installer. InstallShield Developer (InstallShield Corporation), Visual Studio (Microsoft), and Wise for Windows Installer (Wise Solutions) are some examples. See http://msdn.microsoft.com/library/en-us/msi/setup/windows_installer_start_page.asp for more information on the Windows Installer.

MapXtreme Merge Modules

The following are the merge modules for MapXtreme. Include them as necessary in your installer MSI. See Packaging Your Desktop Application in Appendix B on page 445 for more information on using merge modules.
Note the 6.x.x in each MSM filename represents the current version of the product. MSMs are located in \Program Files\Common Files\Merge Modules.

### MapXtreme Merge Modules

<table>
<thead>
<tr>
<th>Name of Merge Module</th>
<th>Included Assemblies</th>
<th>Purpose</th>
<th>When Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Custom Symbols</strong></td>
<td>none</td>
<td>Contains custom symbols</td>
<td>Required when the application expects to use the stock bitmap symbols</td>
</tr>
<tr>
<td>MapInfoCustSymb_6.x.x.msm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Desktop</strong></td>
<td>MapInfo.Windows</td>
<td>Contains assemblies with .NET controls for use in C# and VB .NET desktop applications</td>
<td>Required when the application uses the 'Desktop' tools</td>
</tr>
<tr>
<td>MapInfoDesktop_6.x.x.msm</td>
<td>MapInfo.Windows.Dialogs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MapInfo.Windows.Framework</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MapInfo.CommonDialogExt</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fonts</strong></td>
<td>none</td>
<td>Contains MapInfo fonts such as Symbols, Cartographic, Real Estate, Arrows, Miscellaneous, Oil&amp;Gas, Transportation, Weather, and Shields.</td>
<td>Required when the application expects to use the stock TrueType ® fonts</td>
</tr>
<tr>
<td>MIFonts_6.x.x.msm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mapping</strong></td>
<td>MapInfo.CoreEngine</td>
<td>Provides the core mapping functionality. They also install the common configuration and default reference files for these assemblies.</td>
<td>Required</td>
</tr>
<tr>
<td>MapInfoCoreEngine_6.x.x.msm</td>
<td>MapInfo.CoreEngine.Wrapper</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MapinfoMXTConfig_6.x.x.msm</td>
<td>MapInfo.CoreTypes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MapInfoCoreEngineIntl.msm</td>
<td>MapInfo.WMS.Client</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MapInfoCoreResJPN_6.x.x.msm</td>
<td>MapInfo.Windows.Printing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MapInfoCoreResCHN_6.x.x.msm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### MapXtreme Merge Modules (continued)

<table>
<thead>
<tr>
<th>Name of Merge Module</th>
<th>Included Assemblies</th>
<th>Purpose</th>
<th>When Required</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Web Controls</strong></td>
<td>MapInfo.Web</td>
<td>Installs the .NET assembly MapInfo.Web, and a number of 'web resources' to be used in web-based applications.</td>
<td>Required for any application that uses the stock web controls</td>
</tr>
<tr>
<td></td>
<td>MapInfo.WebControls</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>It also starts the ASPNetState service, and creates a virtual directory for the 'web resources' (if IIS is installed).</td>
<td></td>
</tr>
<tr>
<td><strong>Web Services Clients</strong></td>
<td>MapInfo.Services</td>
<td>Contains the assemblies for geocoding and routing</td>
<td>Required if your application requires geocoding and routing</td>
</tr>
<tr>
<td></td>
<td>MapInfoServices_6.x.x.msm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 2: Getting Started

Steps to Deploying an Application With Your Own Installer

Here are the tasks you must do to get MapXtreme to install with your installer:

1. Include the MSMs you need.
   
   By default, Visual Studio and InstallShield look in C:\Program Files\Common Files\Merge Modules for merge modules to include. This allows you to build installers immediately, without having to re-configure your development environment.

2. Include your runtime license file.
   
   A virtual directory that points to the folder Program Files\Common Files\MapInfo\MapXtreme\6.x.x\MapXtremeWebResources is automatically created.

Deploying a Web Site

If you create your own deployment for a Web site, you will need to add the MSMs manually. The web setup project does not detect the assemblies that are referenced within the Web.config file; thus the assemblies (and the corresponding MSMs) are not detected as dependencies. To add the MSMs manually, right-click on your Web setup project and choose Add>Merge Module. Select MapInfoCoreEngine_6.x.x, MapInfoMXTConfig_6.x.x, MapInfoWeb_6.x.x and any other MSMs you need.

MapXtreme Merge Modules (continued)

<table>
<thead>
<tr>
<th>Name of Merge Module</th>
<th>Included Assemblies</th>
<th>Purpose</th>
<th>When Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS</td>
<td>MapInfo.WFS.Server</td>
<td>Contains the assemblies needed for WFS.</td>
<td>Required if your application uses a Web Feature Service for data transformation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WMS</td>
<td>MapInfo.WMS.Server</td>
<td>Contains the assemblies needed for WMS.</td>
<td>Required if your application uses a Web Map Service for retrieving digital images.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Include if you are deploying applications built with MapXtreme SCP.
† Include if you are deploying applications built with MapXtreme JPN.
‡ Include if you are deploying applications built with MapXtreme CHN.
Deploying Applications that Access Data

Any Visual Basic or Visual C# application that includes data access has a dependency on Microsoft Data Access Components (MDAC) version 2.7 or later. MDAC must be installed on a target computer prior to installing your application or the application will fail. MDAC 2.8 is located on the MapXtreme product CD.

MapXtreme Web Applications Behind Proxy Servers

If you are deploying your web application behind a proxy server, be sure to add the MapXtreme Server URL to the proxy server’s bypass cache list. This will allow MapXtreme to deliver dynamic maps for every request, which it is designed to do.

Proxy servers rely on cached images for display. MapXtreme’s images, however, are built for every request, so in the case of the web application located behind the proxy server, no images are sent to the cache. The web application displays a red X in place of the map image.

Permissions to Temp Directory for Deployed Web Applications

For deployed web applications, if you are using integrated security, ensure all users who will access the site have permissions on the temp directory and any other resources. MapXtreme executes in the ASP.NET process space and this process executes with the security token passed from the IIS process. You must grant access to any user who will log in access to this directory. If you are using anonymous access then you must grant access to the temp directory to the IUSR_LocalMachineName system account. MapXtreme gets the temp directory from the current TEMP environment setting.

Application Data Files

Application data files are nonexecutable files used by an application. MapXtreme installs and uses the following set of application data files. For information about customizing these file locations, see Location of Application Data Files in Appendix C on page 464:

<table>
<thead>
<tr>
<th>File Type</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviation file</td>
<td>MAPINFOW.ABB</td>
</tr>
<tr>
<td>Pen file</td>
<td>MAPINFOW.PEN</td>
</tr>
<tr>
<td>Projection file</td>
<td>MapInfoCoordinateSystemSet.xml</td>
</tr>
<tr>
<td>Vector symbol file</td>
<td>MapInfow.fnt</td>
</tr>
<tr>
<td>Custom symbol directory</td>
<td>CustSymb</td>
</tr>
<tr>
<td>Nadcon files</td>
<td>*.las, *.los</td>
</tr>
<tr>
<td>jgd2000 files</td>
<td>jgd2000.*</td>
</tr>
</tbody>
</table>
By default, MapXtreme applications look in the following directories for data files:

- Program Files\Common Files\MapInfo\MapXtreme\6.x.x—This is the directory the MapXtreme installer places these files.
- The directory where your application is located. For a Windows application, this is the directory where the .exe file is located. For web applications, this is the directory where the Web.config file is located.
- MapInfo.CoreEngine assembly folder—This directory is the last place any application looks.
Before you create a mapping application, it’s helpful to understand basic mapping concepts and how these concepts are implemented in MapXtreme. This chapter discusses the common concepts you will come across as you learn MapXtreme.

At the end of this Developer Guide, we have provided Appendix K: Glossary containing mapping and programming terms that you will also find useful.

In this chapter:

- Mapping and MapXtreme .................................................. 58
- Geocoding with MapXtreme ............................................. 62
- Routing with MapXtreme .................................................. 62
Mapping and MapXtreme

The central element to a mapping application is the map. This chapter presents a short overview of the most important mapping terms that you will likely encounter while building your application with MapXtreme. The introductions also point you to the appropriate namespace in the MapXtreme object model so that you can quickly get the technical information you need. The topics include:

- Maps
- Tables
- Layers
- Features
- Labels and Legends
- Themes
- Tools
- Workspaces
- Coordinate Systems and Projections

Maps

A map displays the spatial relationship among map features, such as town boundaries, customer locations, or power lines. The map visually orients you to where those features are and what they represent. In addition to features, elements on the map can include labels, titles, legends, and themes. Themes are created based on some action taken involving the features and information on the map.

The map is contained in a MapControl. The MapControl also provides basic tools for viewing the map (pan, zoom in, zoom out, center).

You create a map in a variety of ways:

- Use the MapXtreme Workspace Manager to build and save a map workspace. (See Features of the Workspace Manager on page 386).
- Use a MapXtreme template that provides a MapControl that you are drag and drop onto a Visual Studio form (See Chapter 8: Desktop Applications, Controls, Dialogs, and Tools for desktop applications and Chapter 5: MapXtreme Web Controls and Tools for web applications.
- Use the MapXtreme Object Model to programmatically build mapping into your application (See MapFactory on page 231 and the MapInfo.Mapping namespace in the Programmers Reference (online help).

Tables

Tables contain the data you wish to display on the map. Tables hold rows and columns of information that describe the features, including their geometry, style, and attributes. MapXtreme supports tables from a wide variety of sources including, native tables (MapInfo .TAB), relational database management systems (RDBMS), dBase, MS Access, ASCII files, and ESRI ShapeFiles. Speciality tables include raster, grid, seamless, views, WMS, and ADO.NET. The type of table is available through the TableInfo class. Tables are opened and closed via the Catalog in the Data namespace. See Chapter 10: Working with Data.
Chapter 3: Mapping Concepts

Layers

Maps are made up of layers. Layers contain map features, such as postal code boundaries, schools, or streets networks. It is important to understand the order of the layers. The bottommost layer is drawn first and the topmost layer drawn last. Layers containing features that would obscure the features of other layers should be placed lower, respectively. For example, a layer of boundary regions should be placed beneath a layer of points.

Layers in MapXtreme can represent more than features. Layers can be raster or grid images, seamless maps (joined maps), contain labels or user-drawn features, or contain an object theme, such as a pie theme. Layers can be grouped for easier positioning and to facilitate animation of their features. The main interface is IMapLayer. For more information see Layers on page 233.

Features

Features are described by their geometry, style, data source, key and attributes. Typically a feature is a row in a table. Supported geometries include closed objects that cover a given area (Polygons, MultiPolygons, Rings, Rectangle, RoundedRectangles, and Ellipses); point objects that represent single locations of data (Points, MultiPoints); and line objects that cover a given distance (Curves, MultiCurves and LegacyArcs).

One of the main uses of computerized maps is to gather information about the features. In MapXtreme features are returned in FeatureCollections by any of several methods, either created from scratch using a schema, selected using selection tools or methods or by searching the Catalog for those that meet a specific set of criteria.

Feature classes are located in the MapInfo.Data namespace.

Labels and Legends

Maps without elements to describe what is displayed are not very useful. Maps need text such as labels and legends. Labels have been mentioned above as belonging to a type of layer called a LabelLayer. This allows you to control every aspect of a label’s visibility, position, style, and content. MapXtreme classes for working with labels include LabelSource, LabelProperties, and LabelModifiers. See Introduction to the Mapping Namespace on page 230.
Other text elements can also be used in a map to help deliver its message properly. Legends are cartographic elements that describe the features in a coded manner. For example, the legend may describe the boundaries as school districts, the lines as a power line network, or points as corporate office locations. Legends also contain a title to describe collectively what the map represents.

In MapXtreme, legends are part of the Adornments class, along with map titles and scalebars. Adornments reside in the MapInfo.Mapping namespace.

Themes

Computer maps are not only useful for visibly showing spatial relationships among the map features, but you can analyze the underlying data that is associated with the features to learn more about what you see. A common analytical technique is to create a theme based on a feature layer in which the data is ranked in specific ways. For example, a ranged theme shows color blocks where each color represents features on the map that meet the same criteria. A graduated symbol theme is useful for showing distributions of populations for example, with the largest symbol representing the largest population.

Themes can also be created for labels. For example, use a ranged label theme to show the relative population size among cities. The largest labels represent the cities with the largest populations.

The MapInfo.Mapping.Thematics namespace contains classes that implement themes as style overrides on Feature layers and as Object themes. Modifier themes change the style, while object themes add a new layer. All themes implement the ITheme interface.
Chapter 3: Mapping Concepts

Tools

Most mapping applications provide an assortment of toolbar buttons (tools) to aid with common drawing tasks (such as drawing a line on the map) and navigation tasks (such as zooming in). MapXtreme provides a number of mapping tools, plus you can also create your own custom tools.

The tools are divided into desktop tools and web tools, the API for each contained in their own namespace (MapInfo.Tools for desktop and MapInfo.WebControls for web tools.)

For more information on desktop tools in MapXtreme see MapXtreme Desktop Tools API in Chapter 8 on page 139. For more information on web tools see Chapter 5: Web Applications, Controls, and Tools.

Workspaces

While not strictly a mapping concept, workspaces are included here because they will make working with all the mapping elements easier. MapXtreme supports an XML-based workspace format that uses the .MWS extension. In it are all the settings for your maps. The format for the workspace is explained in Appendix D: Understanding the MapInfo Workspace. MapXtreme provides a utility called Workspace Manager to help you construct workspaces and save them for later use. See Chapter 23: Workspace Manager.

Coordinate Systems and Projections

Coordinate systems and projections are two important mapping concepts about which you should have a basic understanding. Projection refers to how a map is displayed on a flat surface such as a paper map or computer screen, while a coordinate system describes how map features are spatially arranged. Both are important considerations when developing applications, especially those where spatial precision and accuracy are important.

A projection is a method of reducing the distortion that occurs when objects from a spherical surface are displayed on a flat surface. There are two main trade-offs to be considered: the preservation of equal area, and the preservation of the true shape of a feature. There are many different types of projections, each designed to reduce the amount of distortion for a given area. Some projections preserve shape; others preserve accuracy of area, distance, or direction.

A coordinate system is a set of parameters that tells you how to interpret the locational coordinates for objects. One of those parameters is projection. Coordinates can be of two types: Spherical or Cartesian. Spherical relates to locations on the curved surface of the Earth, while Cartesian describes flat surface locations in two dimensions. Both are represented by x and y coordinates. The difference comes when calculating distance or area of features that represent real Earth locations such as streets or rivers (Spherical), or relative locations, such as a map of brain anatomy or a chess board (Cartesian).

Knowing which coordinate system your map uses is an important consideration when developing applications. Analytical operations involving distance and area calculations, such as buffering, routing, and querying use the coordinate system and projection to yield the correct results.

Coordinate system and projection classes are part of the Geometry namespace. For more information see Chapter 17: Spatial Objects and Coordinate Systems.
Geocoding with MapXtreme

The maps discussed above all use data that provide additional information beyond what you can see on the map. For example, a table of store locations not only includes geographic coordinates to place the stores in the correct map location, it may contain data about the locations, such as store hours, customer service phone numbers and manager name. This gives the application the power to analyze and yield information that would otherwise be lost in rows and columns of tables.

Typically a table of custom data is included on a map along with reference layers, such as streets, town boundaries and water features that represent the true environment of the area. These references layers are usually purchased ready to display on a map. Pitney Bowes MapInfo sells a wide variety of reference data for locations around the world.

But the custom data, like your store locations, may not be ready to display on a map. The table must contain geographic coordinates so the mapping engine knows where to draw the objects. The process of assigning coordinates to data is called geocoding. Any table of data that contains locational information, such as address or postal code, can be geocoded. The process involves matching the custom table against an already geocoded table covering the same location. If an address match is made, the coordinates from the geocoded table are assigned to the custom data. Then the custom data is ready to be viewed on a map.

Geocoding is a typical early step in the process of creating a map. As a developer of mapping applications, you will need to consider the type of data you wish to display on the map and its need to be geocoded.

The MapXtreme framework provides classes for using a geocoding client that can access Pitney Bowes MapInfo’s server geocoding products. For more information about geocoding see Chapter 19: Geocoding and the MapInfo.Geocoding namespace in the online Help (accessible via Visual Studio).

Routing with MapXtreme

Another component available to developers of MapXtreme is routing. Driving direction applications and those involved in planning routes for deliveries or laying cable, for example, utilize routing. Typically the goal is to locate the route by shortest distance or shortest travel time.

MapXtreme provides four types of routing: point-to-point routing, multi-point routing, matrix routing, and isogram routing. Each type offers numerous options for creating the appropriate routing network for your needs.

Like the provision for geocoding, MapXtreme allows developers to use a pre-built routing client in their application that interacts with Pitney Bowes MapInfo’s routing server products. See Chapter 20: Routing and the MapInfo.Routing namespace in the MapXtreme Online Help.
Understanding the MapXtreme Architecture

This chapter focuses on the design of the MapXtreme architecture so you can make informed choices for your development needs. Understanding the architecture of the product will help you to create applications that efficiently use the features and capabilities of the product.

In this chapter:

- MapXtreme Architecture ........................................... 64
- Object Model Overview ............................................ 65
- Application Architectures ......................................... 68
- Web Application Architecture ................................. 69
- Desktop Application Architecture ......................... 71
MapXtreme Architecture

MapXtreme is built on top of Microsoft’s .NET framework and utilizes the functionality that its infrastructure includes. This Pitney Bowes MapInfo initiative enables you to leverage the power and adaptability of developing your applications on the .NET framework. We have also worked hard to combine the power and ease of our Windows products into one comprehensive object model. The object model is the basis for Pitney Bowes MapInfo’s partners and customers, as well as Pitney Bowes MapInfo itself, for developing Windows-based products for the foreseeable future.

With similar code, you can develop an application that can be deployed on either a desktop or the Web. If you develop an application for the desktop, you can then adjust the application for subsequent web deployment with only minimal code changes.

The following figure illustrates the MapXtreme architecture. The MapInfoCoreEngine.dll assembly and MapInfo.CoreTypes.dll assembly contain most of the core mapping and data access functionality. On top of the Core Engine are the MapInfo.Windows and MapInfo.Web namespaces that contain controls, tools, and other functionality specific to each deployment environment. Any application that gets developed from the MapXtreme object model is built atop of the MapInfo.Windows or MapInfo.Web namespace.
Object Model Overview

The MapXtreme Object Model is made up of many namespaces. A .NET namespace is a type classification system that differentiates specific classes, methods, and properties from others with the same name. By utilizing namespaces, .NET developers can avoid collisions between names of objects and their methods and properties.

MapXtreme ships with a full-color poster of the key interfaces and classes and shows how they relate to each other via namespace segregation. A PDF version of the poster is viewable from the Learning Resources page, which is accessible from the Windows Start menu. Specifically, the Learning Resources page can be accessed from: Start > All Programs > MapInfo > MapXtreme > Learning Resources.

The list below contains several of the namespaces implemented in the MapXtreme Object Model. A broad overview of each namespace is included in the following sections. Each description contains a reference to the part of this manual that contains detailed information about it and its use.

- MapInfo.Data
- MapInfo.Data.Find
- MapInfo.Engine
- MapInfo.Geometry
- MapInfo.Mapping
- MapInfo.Persistence
- MapInfo.Raster
- MapInfo.Styles
- MapInfo.Tools
- MapInfo.Windows.Controls
- MapInfo.WebControls
- MapInfo.Geocoding
- MapInfo.Routing

The complete object model is organized by namespace in the MapXtreme Programmer’s Reference, which is integrated into Visual Studio.

If you have used MapX or the non-.NET version of MapXtreme (MapXtreme for Windows v3), be sure to review Appendix A: Migrating to MapXtreme for a comparison of the two product’s object models.

MapInfo.Data Namespace

The MapInfo.Data namespace contains the classes and interfaces that implement the MapInfo Data Provider. The object model has several different classes to access data. Depending on the format in which your data is stored, there are specific classes to use to access it. Additionally, we now implement ADO.NET access to any data that is contained in formats not covered by any other class. For details on the MapInfo.Data namespace, see Chapter 10: Working with Data and Chapter 12: Accessing Data from a DBMS.
MapInfo.Data.Find Namespace

The MapInfo.Data.Find namespace contains the classes used for searching through data. The namespace facilitates the search for an object by specifying a mappable table and column (it must be indexed) on which to perform the search. For details on the MapInfo.Data.Find namespace, see Chapter 14: Finding Locations.

MapInfo.Engine Namespace

The MapInfo.Engine namespace contains all classes directly related to the core functionality that drives all applications based on MapXtreme. This includes the core Session class which is the starting point for all MapXtreme applications. For details on the MapInfo.Engine namespace, see Chapter 9: Working with Core MapXtreme Classes.

MapInfo.Geometry Namespace

The MapInfo.Geometry namespace is an extensible hierarchy based on OGC (Open GIS Consortium) standards, coordinate system interoperability, and object processing. The MapInfo.Geometry namespace contains classes, interfaces, and enumerations for creating and editing Geometry objects. For details on the Geometry namespace, see Chapter 17: Spatial Objects and Coordinate Systems.

MapInfo.Mapping Namespace

The MapInfo.Mapping namespace contains classes, interfaces, and enumerations for creating, displaying, and exporting maps, layers, modifiers, and labels. For details on the MapInfo.Mapping namespace, see Chapter 13: Adding Mapping Capability to Your Applications.

MapInfo.Mapping.Legends Namespace

The MapInfo.Mapping.Legends namespace contains classes, interfaces, and enumerations for creating and displaying Cartographic and Thematic Legends. For more information, see Legends in Chapter 13 on page 241 and Chapter 15: Using Themes and Legends.

MapInfo.Mapping.Thematics Namespace

The MapInfo.Mapping.Thematics namespace contains classes that implement themes as styles of layers and as layers themselves. Themes can be applied to Modifier themes to change the style, while object themes add a new layer. All themes implement the ITheme interface. For details on the MapInfo.Mapping.Thematics namespace, see Chapter 15: Using Themes and Legends.

MapInfo.Persistence Namespace

The MapInfo.Persistence namespace contains classes that support the reading and writing of XML-based workspaces to enable the saving and retrieval of mapping workspaces. See Appendix D: Understanding the MapInfo Workspace.
MapInfo.Raster Namespace


MapInfoYLES Namespace

The MapInfo.Styles namespace highlights the Styles object model. The Styles class is the base class of all styles. For details on the Styles namespace, see Chapter 16: Stylizing Your Maps.

MapInfo.WebControls Namespace

The MapInfo.WebControls namespace provides support for using Visual Studio templates for a MapXtreme ASP.NET application. There are MapControl and LayerControl design-time enhancements available from this namespace, as well as web tools. For details on the MapInfo.WebControls namespaces, see Chapter 5: Web Applications, Controls, and Tools.

MapInfo.Windows Namespaces

The MapInfo.Windows namespace contains classes that implement various windows controls and their requisite components for use with developing forms in Windows applications. The Windows.Dialogs namespace contains classes that implement various dialog boxes and dialog box components to be used in Windows applications. For details on the MapInfo.Windows namespace, see Chapter 8: Desktop Applications, Controls, Dialogs, and Tools.

MapInfo.Tools Namespace

The MapInfo.Tools namespace contains classes for creating and implementing many types of tools to use in your desktop map application. For details on the MapInfo.Tools namespace, see Overview of the MapInfo.Tools Namespace in Chapter 8 on page 137.

MapInfo.Geocoding Namespace

The MapInfo.Geocoding namespace contains the classes, interfaces and enumerations that define the MapXtreme client for geocoding. Geocoding using either the MapInfo geocoding server or the MapInfo Location Utility service is supported. The URL of a running geocoding server or Location Utility service must be available in order to perform geocoding. The interface of the geocoding server and Location Utility service are similar, since they both use the same classes for geocode requests, constraints, responses, result codes, and input and candidate addresses. See Chapter 19: Geocoding.
MapInfo.Routing Namespace

The MapInfo.Routing namespace contains classes, interfaces, and enumerations which comprise the .NET client for Routing. The MapInfo.Routing namespace contains classes that support point-to-point, multi-point, matrix and isogram routing. optimized for shortest time or shortest distance. It interacts with other MapInfo routing server products. The ability to avoid certain points is also available. Routing results can include step by step directions and/or a route geometry which can be displayed on a map. See Chapter 20: Routing.

Application Architectures

Now that you have had an overview of the MapXtreme namespaces, the next step is to consider the architecture of the application you plan to build.

Using MapXtreme, you can build both web and desktop applications. The follow sections illustrate possible architectures for web and desktop applications. The designs are based on the Model-View-Controller paradigm that separates an application's data model, user interface, and control logic into three distinct components. This allows for modifications to one component with minimal impact to the others. Chapter 5: Web Applications, Controls, and Tools and Chapter 8: Desktop Applications, Controls, Dialogs, and Tools provides important additional information on key design elements and decisions you need to consider when planning your MapXtreme application.
Web Application Architecture

Architecture Description

The Web application illustrated above takes into account the following components and capabilities:

- MapXtreme Web Application
- Microsoft .NET Infrastructure
- Map-building Tools
- Data Access

MapXtreme Web Application

A typical MapXtreme web application contains views (presentation layer), a model (to interact with data source and application internal data model), and controller (the business logic that controls the flow of the application).

MapXtreme provides web templates that are integrated into Microsoft Visual Studio to help you create your initial web application. For a tutorial on building a web application, see Appendix B: Building a Web Application.
Build views by dragging and dropping MapXtreme web controls onto a Visual Studio web form. Build your internal data structures and interact with external data sources, base maps and dynamic content by using objects under MapInfo.Engine and MapInfo.Data namespaces. Use the controller code to tie the views and data together and provide the user with an engagement sequence to effectively use the application to resolve a business need or problem.

**Microsoft .NET Infrastructure**

MapXtreme runs under Microsoft .NET 2.0 Framework. An application built using MapXtreme runs as an ASP.NET application under the worker process of IIS.

The Microsoft ASP.NET framework provides COM+ object pooling for developers of high performance enterprise applications so that objects such as workspaces can be preloaded. MapXtreme’s object model operates very efficiently under this framework. The framework also provides application state management tools such as StateServer and SQL Server, as well as automatic and manual state management control. Chapter 6: Understanding State Management presents important information on these topics.

**Map-building Tools**

Use the MapXtreme Workspace Manager to create your application’s base maps. Here you can manage each layer of a map and control its zoom level, labeling, styles, themes and adornments to give you exactly the presentation you need for your application. The information is saved to an XML-based workspace for easy retrieval at a later date. See Chapter 23: Workspace Manager.

**Data Access**

A key element of this architecture is the ability to access to dynamic data content. Objects that exist within the MapInfo.Data namespace provide this data access. Data content can come from a number of sources, such as WMS, WFS, remote database management systems, live feeds from GPS or driving directions from the MapInfo Routing service. To make the most of disparate data, you can concurrently use information from different data sources. See Chapter 10: Working with Data.
Desktop Application Architecture

The desktop application architecture is similar to the web application architecture in its Model-View-Controller design.

Separate components are used for the presentation layer, application model and business logic. Use the Workspace Manager to build any base maps that you need. Use Windows controls and dialogs to give your application a rich user experience. You can also concurrently use dynamic data content from a variety of sources, and control the flow and logic of the application.

See Chapter 8: Desktop Applications, Controls, Dialogs, and Tools for more information.
Web Applications, Controls, and Tools

This chapter brings together a host of information related to building ASP.NET web applications using the web controls, tools, and conveniences provided with MapXtreme.

In this chapter:

- Web Application Request/Response Lifecycle ..................74
- Components of a MapXtreme Web Application ..................74
- MapXtreme Web Controls and Tools ..........................77
- Web Control Architecture ......................................80
- Using the MapXtreme Web Controls ..........................84
- Managing Your Web Controls .................................85
- Creating a Custom Tool .........................................86
- Using and Distributing Custom Web Controls ..................88
- ASP.NET AJAX and MapXtreme Web Applications ..........89
- HTML/XHTML Validation Issues ..............................91
- Specialized Topics for Web Controls ..........................92
In order to plan and build an effective web application, you need a solid understanding of the behind the scenes interactions between the client (browser) and the server (web application). In its simplest form, a web application is a software application that is accessed through a web browser over an Internet or intranet connection. The capabilities of the application are presented to a user as an HTML page, and through user interaction with the elements on the web page, HTTP requests are sent to a web server for processing. The web server sends back a response that satisfies the user’s request.

A MapXtreme web application typically presents the user with an image of a map and some tools to interact with the map. A single request/response cycle could be as simple as the user clicks a Zoom-In tool to display a different view of the map. Behind the scene, the request to zoom in is sent to the server. The server processes the request and responds with a refreshed image of the map showing the new view.

For information on the architecture of a MapXtreme web application, see Chapter 4: Understanding the MapXtreme Architecture.

For a tutorial on building a MapXtreme application, see How to Create and Deploy a MapXtreme Application in Appendix B on page 435.

For a discussion of creating ASP.NET web applications, see Creating ASP.NET Web Applications on Microsoft’s MSDN.

Components of a MapXtreme Web Application

The following sections cover the major components that make up a typical MapXtreme web application, including:

- MapXtreme Session
- Background Map
- MapControl
- Map Tools
- State Management and Pooling Capabilities

MapXtreme Session

The MapXtreme Session is the starting point for all MapXtreme applications. It manages the initialization of resources needed for a MapXtreme application. The MapXtreme Session also provides access to other important objects such as Catalog, MapFactory, CoordSysFactory, Selections, and others.

To access the MapXtreme Session, call the MapInfo.Engine.Session.Current() method. Each thread in your process has a MapXtreme Session object available. There can be only one MapXtreme Session per thread and this session cannot be shared by multiple threads. See Chapter 9: ISession Interface.
Web applications can have one MapXtreme Session object per user, or pooled MapXtreme Sessions available to many users. Both development models are discussed in Chapter 6: Understanding State Management.

Background Map

The background map, in the form of an image, is the most visual component of a MapXtreme web application. It provides the user of the application with information about map features and shows their relationship to other map features. Typically, a background map is made up of reference layers, such as administrative boundaries, street networks, and points of interest. Custom data related to the application, such as points representing office locations, cell towers, or ATM machines, are additional layers. The reference layers and the custom layers do not change based on the user’s interaction with the application. What typically changes is the display of the map. A user may zoom into a particular location or create a thematic shading that overlays the map based on criteria the user submitted in a web request.

The background map is presented to the user in its initial, or "clean" state. This state is an important consideration when you design a pooled application since your application must handle changes in user state.

The background map is pre-loaded into the application from an XML-based workspace (.mws). The workspace is identified in the Web.config file of the application. See What Should the Initial Map Look Like? in Chapter 6 on page 99. The MapXtreme web template and sample applications provide Web.config files that include hard-coded paths to sample data. If you base your web application on one of these, be sure to adjust the path to your own data.

The Workspace Manager utility provided with MapXtreme is an example of a desktop application that assists you with building a background map. See Workspace Manager in Chapter 23 on page 385.

MapControl

MapControl is a MapXtreme web server control containing an instance of a Map object. The MapControl is the main web control in a MapXtreme application; it displays the background map and responds to tool interactions on the map. Behind the scenes, the Map object is obtained from the MapFactory using the MapAlias property of the MapControl. The map is rendered and exported as an image and returned to the browser in the control's <img> tag.

The sample applications that ship with MapXtreme have a MapControl built in, as does the MapXtreme Web Application template. You will see, however, that in design mode, the map is not rendered. Only runtime rendering is supported.

Note If you start from a Visual Studio Visual Basic or C# template, you must manually add the MapControl and tools to your web form in order for the MapXtreme resources to be included in your project.

For more information, see MapXtreme Web Controls and Tools on page 77.
Map Tools

MapXtreme provides a number of map tools to assist you in navigating and interacting with the background map. These tools are contained in the Visual Studio toolbox. Use them by dragging and dropping the tool onto a web form.

Many of these tools are built into the MapXtreme web template and sample applications, including:

- Basic tools for Center, Pan, Zoom-in, and Zoom-out
• ZoomBarTool with pre-set zoom levels between 500 and 12,500 map units
• Directional tools (N, NE, E, SE, S, SW, W, NW) for direct line panning in 45-degree increments

The LegendControl, Distance tool and selection tools are located in the MapXtreme portion of the Visual Studio toolbox.

Tools are discussed in Description of Web Controls and Tools on page 78.

A generic WebTool, the base tool for all MapXtreme map tools is also located in the toolbox. Use this tool if you wish to add custom behaviors to a tool. See Creating a Custom Tool on page 86.

State Management and Pooling Capabilities

One of your goals in building a MapXtreme web application is to make it stable, scalable and offer a satisfying experience for your users. MapXtreme provides configurations to help you manage the state of your application, and allow it to grow with the numbers of users.

The MapXtreme Web Application template and the sample applications are configured to manage state manually. This means that only the changes your users make during their interaction with your application are saved and restored. You accomplish this by writing code that is specific to your application and save/restoration needs.

To more efficiently serve the potentially growing number of users of your application, the template and sample applications are configured to use the Microsoft COM+ pooling service in which MapXtreme Session objects are available and shared across multiple requests.

State management and pooling need careful consideration when you are planning your web application. See Chapter 6: Understanding State Management.

MapXtreme Web Controls and Tools

MapXtreme’s web controls and tools are embedded in a web page. Web controls respond to interactions by web tools. Information that is captured using a tool is processed on the server by the web page and returned to the client, typically as a new map image.

MapXtreme provides three web controls (MapControl, LayerControl, and LegendControl) and a number of map tools. The web controls show content, such as a map, a list of map and label layers and their properties, or a legend to identify what the layers represent. The tools interact with the MapControl to change the view of the map, select features on the map for further analysis, and more. For a description of the web controls and tools see Description of Web Controls and Tools on page 78.

The Web controls and tools are available from several places within Visual Studio:

• Visual Studio toolbox under the MapXtreme heading. Drag and drop these controls onto your Web Form to add mapping functionality to your project.
• Web Site Application template: Start with the MapXtreme Web Application template (Visual Studio File > New > Web Site menu) to build a mapping prototype with no coding.
• Sample Applications: Task-oriented samples are located in the \Samples\Visual Studio 200x\Web\Features folder. Source code is also provided for you to learn from or extend for your own needs.
JavaScript-Enabled for Partial-Page Updates

MapXtreme web controls and tools use a combination of a web page and JavaScript to tell the MapXtreme server what needs to be done. Each map tool specifies a JavaScript ClientInteraction that defines what must happen (for example, click, draw a rectangle, line, or polygon) and a JavaScript ClientCommand that sends a URL request to the server to process the command (for example, pan, zoom, or select an object).

These JavaScript-enabled tools do not trigger a full page postback with each use. Typically, only the image of the map is refreshed after each tool use. See Map Tools Architecture on page 81 for an understanding of this development model. See Managing Your Web Controls on page 85 for information on event handling, error management and state management.

Previous MapXtreme controls (pre-v6.5 releases) required a postback that called Page_Load and Page_Unload every time a tool was used. These tools are provided for backward compatibility and are not recommended for new development projects. See Migrating from pre-v6.5 Web Controls to 2005/2008 Web Controls on page 91.

Customizable

If the provided tools do not offer you the functionality to meet your needs, consider modifying them. This can be as simple as changing a built-in tool property or as complicated as writing your own JavaScript and server-side class to extend its functionality. Source code for the web controls and tools is provided in the Samples\MapXtremeWebControlsSourceCode folder. See Creating a Custom Tool on page 86.

MapXtreme provides an ASP.NET AJAX sample application that demonstrates how to use Microsoft's ASP.NET AJAX controls in a MapXtreme web mapping application. See ASP.NET AJAX and MapXtreme Web Applications on page 89.

Description of Web Controls and Tools

The following are the available MapXtreme Web controls and tools.

<table>
<thead>
<tr>
<th>Web Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapControl</td>
<td>Allows you to display an instance of a Map object. Each Map object is referred to by a MapAlias, such as Map1. At runtime, the MapControl displays the map which is obtained from a MapFactory by using the map’s MapAlias property. The map is drawn by exporting the map image, and referencing this image in the HTML &lt;img&gt; tag. If the MapAlias property is not specified or is invalid, the first map from the MapFactory is chosen. Set the MapControl MapAlias property at design time.</td>
</tr>
<tr>
<td>LayerControl</td>
<td>Allows you to display map feature layers and label layers in a tree view structure. This control can turn the visibility on or off for a particular layer and displays a read-only current zoom value. The visibility changes are persistent only for the active life of the application using the LayerControl.</td>
</tr>
</tbody>
</table>
Chapter 5: Web Applications, Controls, and Tools

LegendControl
The LegendControl allows you to display a legend for a given MapControl. The legend that is returned is a non-interactive image. The default export format is .GIF.

The legend to display can be specified at design time using its LegendAlias or its index in the map's legend list. This can also be set using JavaScript on the page. JavaScript can also be used to show and hide the legend. You can arrange for the legend to be scrollable.

Both thematic and cartographic legends are supported in the LegendControl.

The Legend Control web sample application supports this control. It demonstrates how to create and use a customized LegendControl based on our current web control architecture, and how to create a theme and display a legend by sending requests to the server using JavaScript without needing to refresh the whole page. For more information, see the MapXtreme Learning Resources browser on your Program menu or from the MapInfo website under Support & Training. Expand the Learning Resources link and click on Sample Applications.

<table>
<thead>
<tr>
<th>Map Tools</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CenterTool</td>
<td>Allows you to recenter the map by clicking on the map.</td>
</tr>
<tr>
<td>DistanceTool</td>
<td>Allows you to get the distance between two or more points by clicking on the map.</td>
</tr>
<tr>
<td>NavigationTools</td>
<td>Allow you to pan the map by fixed directions: North, South, East, West, NorthEast, NorthWest, SouthEast, and SouthWest.</td>
</tr>
<tr>
<td>PanTool</td>
<td>Allows you to reposition the map within its window by dragging in any direction.</td>
</tr>
<tr>
<td>PointSelectionTool</td>
<td>Allows you to select a feature (nearest to the point) when clicked on the map.</td>
</tr>
<tr>
<td>PolygonSelectionTool</td>
<td>Allows you to select all features whose centroids lie within the polygon. The selection area is drawn on the map using mouse clicks representing the nodes of the polygon. A double click closes/ends the polygon.</td>
</tr>
<tr>
<td>RadiusSelectionTool</td>
<td>Allows you to select all features whose centroids lie within the radius. The radius is drawn on the map using mouse clicks representing the center and boundary of the circle. The selection radius is drawn on the map using a click and drag operation.</td>
</tr>
<tr>
<td>RectangleSelectionTool</td>
<td>Allows you to select all features whose centroids lie within the rectangle. The rectangle selection area is drawn on the map using a click and drag operation.</td>
</tr>
</tbody>
</table>
Web Control Architecture

The MapXtreme Web control architecture follows the ASP.NET model for creating Web applications. The general architecture is a Model-View-Controller (MVC) design pattern, in which the web application represents the Model, the web page (HTML, JavaScript) represents the View, and the MapXtreme Server that responds to requests for information represents the Controller.

The user interacts with the web application through web controls and tools that capture data and send instructions and commands to the server which processes and returns the information.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZoomBarTool</td>
<td>Allows you to zoom a map to a series of preset levels between 500 and 12,500 map units.</td>
</tr>
<tr>
<td>ZoomInTool</td>
<td>Allows you to zoom a map by either a single click or by selecting a rectangular area</td>
</tr>
<tr>
<td>ZoomOutTool</td>
<td>Allows you to draw a rectangle representing the view to zoom out of the map.</td>
</tr>
<tr>
<td>InfoTool Sample Web Application</td>
<td>An example of an InfoTool is included in the sample Web applications. It demonstrates how to create and use a customized map tool based on the MapXtreme Web control architecture.</td>
</tr>
</tbody>
</table>

In MapXtreme, controls and tools are rendered when the web page is rendered at initialization time. After initialization time, when a tool is used, only the map image is rendered. These JavaScript-enabled tools are an improvement over the pre-v6.5 web controls that performed a full page postback with every tool operation.

MapXtreme controls provide the following behaviors and functionality:

- A background map is loaded via a pre-defined workspace. At design time, set the MapControl MapAlias property to the map alias of the map defined in the preloaded workspace. At runtime, the corresponding map is loaded into the MapControl, ready for users to interact with it using map tools.
• The first time a page is rendered, Page_Load and Page_Unload are called. Page_Load initializes the state of the application, either to a default state for a new session, or restores state if the session is not new. Page_Unload stores the state of any changes, in anticipation of another request.
• The StateManager is implemented in an application as a class, and an instance of that class is put in the MapXtreme Session. SaveState and RestoreState methods are called from this object. The SaveState and RestoreState methods are called every time a tool is used. If Manual mode is used for state management, a StateManager class instance must be in the session. (MapInfo.Engine.Session.State is set to Manual in your Web.config file.) For more on state management, see State Management on page 85.
• Error processing is done in the global_asax.cs/.vb file in the application_error event handler. See Error Management on page 85.
• Events are handled through client-side JavaScript commands that send a request to the server. A server-side command class does server-side processing. See Event Handling on page 85.

The MapXtremeWebResources folder included with the Visual Studio MapXtreme Web templates contains dependencies for the web controls and map tools. Make sure to include these files when deploying an application.

Map Tools Architecture

MapXtreme map tools are used to interact in some way with the map, such as panning to a different view or selecting an area of the map to collect data for further analysis. A basic set of navigational tools are provided in the MapXtreme web application template. These tools, along with others for selecting map features and creating a legend, are located in the Visual Studio toolbox.

The map tools are made up of client-side and server-side components. On the client side, the tools have a JavaScript interaction component and a JavaScript client command component. On the server side, the tools have server command class component.

The client-side map tool components are responsible for:
• Drawing and mouse operations (for example, rubber band rectangle, mouse click)
• Collecting data from mouse operations (for example, getting the screen coordinates for a zoom-in operation)
• Sending the url request to the server

The server-side map tool components are responsible for:
• Carrying out the business logic of the tool (for example, calculating the distance between two points)

Client Side Map Interaction

The client side map tool interaction is implemented with JavaScript classes. These are generic classes that can be used by any user interface to perform an interaction on any HTML element. The base Interaction class, located in Interaction.js, is extended to create all of the individual interactions such as ClickInteraction and RectInteraction. The constructor for Interaction is:

```
Interaction(elementID, onComplete)
```

where `elementID` is the IMG tag of the map and `onComplete` is the function which is called when the interaction is complete.
For example, the Interaction class can draw a rubber band rectangle over the map and collect all the point features that are contained within it.

**Client Side Command Execution**

A map tool has a client side JavaScript command object that is responsible for performing a specific task. The base Command class, located in Command.js, is extended to create all of the individual commands such as PanCommand and ZoomCommand. After the map tool interaction is complete, the tool executes the client command. The Command class constructor is:

```
Command(name, interaction)
```

where `name` is the server side Command class name and `interaction` is the data gathered during the client interaction.

The client Command generates a URL request that is sent to the server, which then processes the response to display the new map.

The interaction object can be `null`.

**Server Side Command Architecture**

On the server, the MapInfo.WebControls.MapControlModel class handles the URL request sent from the client. This class contains methods for carrying out basic map navigation commands such as pan and zoom, as well as selection commands for selecting points, polygons, and features within drawn rectangles and radii. The InvokeCommand method locates the specified tool command and calls its Execute method. Execute calls the RestoreState, then Process, and then finally SaveState.

To perform commands other than those provided in this class, you must write a Command class that extends the WebControls.MapBaseCommand class.
How a Map Tool Works

The following describes a typical request/response cycle for a web map tool, in this case, a Zoom-In tool. You can create custom map tools using this same architecture. See Creating a Custom Tool on page 86.

The numbers in the diagram refer to the stages described below.

1. A user draws a rectangle around the general area of Europe displayed in a MapControl using a Zoom-In tool.
2. The tool's ClientInteraction property called RectInteraction collects the screen coordinates that define the rectangle. RectInteraction is defined in Interaction.js.
3. The tool's ClientCommand property MapCommand creates a URL request and assigns it to the MapControl's image.src property. ClientCommand is defined in Command.js. The URL request looks like:
   ```
   MapController.ashx?Command=Zoom&Width=300&Height=300&ZoomLevel=1200&Points=1,50,100&MapAlias=Map1
   ```
4. The URL request is sent to the server.
5. The MapController receives the request and invokes the MapControlModel. The MapController derives from System.Web.IHttpHandler.
6. The MapControlModel parses the URL request and invokes the ZoomInCommand class.
7. The ZoomInCommand gets the map from the MapFactory and carries out the operation to zoom in on the map. The map image is updated to reflect the new view.
8. The map image is exported to an image, and written to the HTTP response as a stream, and returned to the client.
Using the MapXtreme Web Controls

To use the MapXtreme Web controls in your web application:

1. Do one of the following:
   a. Create a MapXtreme web application from Visual Studio File > New > Web Site and choose the MapXtreme Web Application template. The MapControl, LayerControl and map navigation tools are built in.
   b. Open one of the sample web applications and modify it for your needs.
   d. Create a MapXtreme web application from Visual Studio File > New > Web Site and choose the generic ASP.NET Web Site template.

2. In your application’s Web.config file, specify the path and name of a workspace that contains map layers for pre-loading into the MapControl. This is necessary regardless of which choice you made in step1.

   The Web.config file for the template and samples contains a path to the default location of the installed sample data, as shown below:

   ```xml
   <configuration>
     <appSettings>
       <add key="MapInfo.Engine.Session.Workspace" value="c:\Program Files\MapInfo\MapXtreme\6.x.x\Samples\Data\World.mws" />
     </appSettings>
   </configuration>
   
   If your application requires multiple maps, they can exist in one or more workspaces, each with a unique MapAlias property. To preload more than one workspace in the Web.config file, use a semicolon between the full path of each workspace.

3. Make sure that MapAlias property of the MapControl is set. The default MapAlias for the MapControl in the template is Map1. If this property is not set, MapXtreme will render the first map from the session which may not be the map you expect.

   To determine the MapAlias, open the .mws file in a text editor and look for the MapDefinition element. The MapAlias is stored as an alias attribute.

   You can also find the MapAlias from Workspace Manager. When you hover the mouse pointer over the map node (top node in the layer list), the alias appears in a ToolTip.

4. Set the MapControlID property for all map tools, LayerControl, LegendControl and custom controls (if any) to point to the appropriate MapControl.

---

1. In this situation, you will notice that the MapXtreme web controls and tools display red X’s in the Designer instead of their icons. To display the icons properly, copy the MapXtremeWebResources folder from one of the MapXtreme sample applications and paste it into your project where your Web.config and default.aspx files are located. Close and re-open the web page to see the icons. To avoid this manual step, choose option “a” instead.
Managing Your Web Controls

An important part to using Web controls is managing them effectively. This section explains how to perform:

• Event Handling
• Error Management
• State Management

Event Handling

Previously, in the pre-v6.5 MapXtreme postback controls, event handlers were written in the Page class. After the tool was used, the postback executed the code in the event handlers.

The current JavaScript-enabled map tools do not perform a postback with every use. Map tool events are handled through client-side JavaScript commands that send a request to the server. On the server-side a command class derived from MapBaseCommand does processing for the tool.

Error Management

Error handling in the Web controls can be handled in many ways, and it is very specific to the application. Therefore, this section only explains one of the ways of handling errors. Since the response expected by the client side is an image, we can catch the exception using a detailed message, and send the response back with both an image and an error message. As a result, the MapControl will now contain the error message.

The Global.asax files provided with our sample applications demonstrate an example of handling application errors. If the application encounters an error while processing a request for a map image, the Application_Error method creates an image containing an error message, and returns that to the client.

For more information on Error Management, refer to the MSDN site on error handling with ASP.NET.

State Management

The MapInfo.WebControls.StateManager is an abstract class that includes methods SaveState and RestoreState. MapXtreme’s map tools call RestoreState and SaveState before and after the processing, respectively. Since state management is application specific, it is your responsibility to implement these methods in a concrete class in the application. This allows you to control what gets restored and saved and how things are restored and saved.

For more information on state management, see Chapter 6: Understanding State Management.
Creating a Custom Tool

To create a custom tool, you can modify or add behavior to a built-in tool or write your own custom commands and tools.

For example, you may wish to modify a Zoom-In tool to zoom-in and select a feature with one click. This tool requires a server command class that contains code to zoom-in and perform the selection. Since we provide the source code for all the server command classes, you can simply modify the ZoomIn command class to add the selection code.

If our source code does not provide a starting point for your customization, you must write your own commands and tool classes. The source code is located in \Samples\MapXtremeWebControlsSourceCode.

As you plan your customization, keep the following MapXtreme tool architecture in mind. A MapXtreme map tool consists of:

- A client-side tool class that inherits from MapInfo.WebControls.WebTool.
- Properties on the tool class that control the behavior of the tool, including:
  - JavaScript that describes the interaction of the tool with the MapControl (click, draw rectangle, etc.)
  - JavaScript that creates the url request for the tool.
- A server-side command class that is derived from MapInfo.WebControls.MapBaseCommand that carries out the desired tool behavior.

Properties for the tool are defined either in the class or on the web page. The custom tool is referenced on the web page by the tool class name.

The code used in the following procedure can be found in its entirety in the ..\MapInfo\MapXtreme\6.x.x\Samples\VisualStudio2005\Web\Features\CustomTools\CustomToolsCS.

To create a custom tool:

1. Drag the generic WebTool from the MapXtreme toolbox onto your web form. You can also use one of the existing map tools if you want to extend the existing behavior.
2. In the WebTool property page, set the properties for MapControlID, InActive/ActiveImageUrl and CursorImageUrl.
3. Set the appropriate ClientInteraction property by selecting from the drop-down list.
   The built-in interactions include mouse operations for clicking and dragging, drawing lines, polygons, rectangles and radii, which will cover the needs of most web application. See the Interaction.js in the MapXtremeWebResources folder in your project.
4. Set the appropriate ClientCommand property by selecting from the drop-down list.
   The built-in client commands for mapping, panning, zooming, etc., create the URL request that is sent to the server. For a description of these commands, see the Command.js in the MapXtremeWebResources folder in your project.
   If one of the built-in commands does not meet your needs, either modify the existing Command.js or write your own. The custom command takes the name of the interaction from step 3 as input. See CustomCommand.js in the CustomToolsCS or CustomToolsVB sample for an example of how to get multiple responses from the server with a single click.
5. Register the JavaScript manually in your .aspx page. Insert the following line within the web page body.

```html
<script language="javascript" src="CustomCommand.js" type="text/javascript"></script>
```

6. Create a new server command class that derives from MapInfo.WebControls.MapBaseCommand. Include code that carries out the behavior that the client command requested. Alternatively, you can extend an existing server command class.

7. In the server command class, assign the name of the server command in the constructor.

```csharp
namespace ToolsSample
{
    public class AddPinPointCommand : MapInfo.WebControls.MapBaseCommand
    {
        /// <summary>
        /// Constructor for this command, sets the name of the command
        /// </summary>
        /// <remarks>None</remarks>
        public AddPinPointCommand()
        {
            Name = "AddPinPointCommand";
        }
    }
}
```

8. In the server command class, override the Process() method by adding code that carries out the business logic for the command.

```csharp
public override void Process()
{
    // Your code here.....
}
```

9. In the Page_Load method of the webform1.aspx, add your server command to the collection of commands in the MapControlModel.

```csharp
MapInfo.WebControls.MapControlModel controlModel = MapControlModel.SetDefaultModelInSession();
controlModel.Commands.Add(new AddPinPointCommand());
```

10. Add the tool to the Visual Studio toolbox by creating a new assembly. See Using and Distributing Custom Web Controls on page 88.

11. Drag and drop the custom tool onto the web form and set the properties to the names of the ClientCommand and ClientInteraction, as set forth in your JavaScript from steps 3and 4.

12. Set the property for the server Command, as defined in step 6.

13. Register the tag prefix in the web form by specifying the assembly and namespace in which this control exists.

```csharp
<%@ Register TagPrefix="cc1" Namespace="MapInfo.WebControls" Assembly="MapInfo.WebControls, Version=4.0.0.476, Culture=neutral, PublicKeyToken=0a9556cc66c0af57" %>
<%@ Page language="c#" Inherits="ToolsSample.WebForm1" CodeFile="WebForm1.aspx.cs" %>
<%@ Register TagPrefix="cc2" Namespace="CustomizedWebTools" %>
```
Using and Distributing Custom Web Controls

Once you have created a custom web tool, you must include it in an assembly so that it is available in the Visual Studio toolbox or to distribute it to others.

Whether you modify the MapXtreme source code or create your own tool class from scratch, consider the following important factors regarding the MapXtreme web control assembly.

- The assembly name is MapInfo.WebControls.dll and is installed in the global assembly cache. This assembly has a specific version number which is used by our templates and sample applications.
- The controls in the assembly are installed in the toolbox in Visual Studio.
- The assembly has references to MapInfo.CoreEngine.dll and MapInfo.CoreTypes.dll.
- Resources such as images and scripts are installed in the C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x\MapXtremeWebResources folder.
- The following files are used by the Web controls:
  - *.GIF images that represent tool actions (for example, DistanceToolControlActive, DistanceToolControlInactive).
  - *.BMP images that represent tool icons (for example, label, selection arrow)
  - JavaScript defining tool interactions and behaviors (Interaction.js, Command.js, LayerControl2.js, LegendControl.js and Tool.js)
  - *.CUR (cursor) files that display images when the mouse is used.

Creating a Web Assembly

To create a custom web control assembly:

1. Copy the customized web control source files to another directory so the original is preserved.
2. Remove the original assembly from the global assembly cache and from the Visual Studio toolbox (Start > Run > Assembly).
3. Create a strongly named key file (.snk) using `sn – k MapInfo.WebControls.snk` and copy this key file to the main project folder (same level as the project files).
4. Change the AssemblyInfo.cs or AssemblyInfo.vb file to reflect the version number of your web assembly.
5. Open the project in Visual Studio, make any changes you need and build the project. The new assembly should be in the bin\Release directory so you can distribute the release version of the assembly.
6. Register your new assembly in the global assembly cache, and in the Visual Studio toolbox using the following syntax as a model. For more information see http://msdn2.microsoft.com/en-us/library/ex0ss12c(vs.80).aspx
   ```
gacutil /i MapInfo.WebControls.dll
   ```
7. Write your application using the new assembly. Drag and drop the new controls onto your form from the Visual Studio toolbox.
8. In the installer for your Web application, make sure the new assembly is installed in the global assembly cache. As long as the Web application points to this version of the assembly, it will use the new controls.
Consider the following scenarios:

- You can insert the Web controls project and resources right into your Web application solution. In this case, change the URLs for your resources (scripts and images) to begin with the project root. This prevents you from having to create virtual directories.
- You do not have to use the global assembly cache and a strongly named assembly. You can set the Copy property for the assembly to true, and have the assembly in the bin folder of your application.

**ASP.NET AJAX and MapXtreme Web Applications**

MapXtreme’s web controls and tools include JavaScript which provides an efficient request/response cycle for web applications. Each time a map tool is used, JavaScript interactions and commands carry out the operation without requiring a full-page postback to the client. Typically, only the map image is refreshed.

Microsoft’s ASP.NET AJAX technology takes this behavior further, by integrating scripting libraries with the ASP.NET 2.0 Framework. The principal controls are the UpdatePanel, a container for server controls that are frequently refreshed, and the ScriptManager, which manages the scripting activities for the web page.

MapXtreme provides a sample application that demonstrates how to use Microsoft's ASP.NET AJAX controls in a MapXtreme web mapping application. The sample is located in ..\MapInfo\MapXtreme\6.x.x\Samples\VisualStudio2005\Web\Features.

**Note** The AJAXDemo sample requires that the Microsoft ASP.NET 2.0 AJAX Extensions 1.0 be installed on your system. These are available on the MapXtreme product CD.

The following section provides the steps to add AJAX controls to your MapXtreme application.

Before proceeding, as an exercise in understanding AJAX, it’s a good idea to create a web application using the "ASP.NET AJAX-Enabled Web Site" template that is provided with the AJAX extensions. Examine its Web.config file; the httpHandlers section and the httpModules section contain settings that you may need to copy into your application.

You should also familiarize yourself with the AJAXDemo sample web application. The steps that follow refer to JavaScript code and Web.config settings taken from this sample.

**Adding ASP.NET AJAX Controls to a MapXtreme Web Application**

To add ASP.NET AJAX controls to an existing MapXtreme web application:

1. Open your web form in Visual Studio’s Designer mode.
2. From the "AJAX Extensions" section of the Visual Studio Toolbox, drag a ScriptManager control onto the form. (It does not matter where you place the ScriptManager, as it will not be visible at run-time.)
3. Drag an AJAX UpdatePanel control onto your form.
4. Move standard controls, such as Button controls, inside the UpdatePanel, to prevent the Button from causing a full-page update.

**Note** Do not move MapXtreme controls, such as the MapControl or the LayerControl, inside the UpdatePanel. For a detailed example, see the AJAXDemo sample application.

5. Open your application's Web.config file, and locate the httpHandlers section. Depending on the contents of your web application, the httpHandlers section might contain just one or two entries—one for MapController.ashx, and, if your application contains the LayerControl, one for LayerController.ashx.

```xml
<httpHandlers>
    <add verb="*" path="MapController.ashx" . . .
    <add verb="*" path="LayerController.ashx" . . .
</httpHandlers>
```

6. Open the Web.config file from the AJAXDemo sample application, and locate its httpHandlers section, which contains additional entries needed by ASP.NET AJAX:

```xml
<httpHandlers>
    <remove verb="*" path="*.asmx"/>
    <add verb="*" path="*.asmx" . . .
    <add verb="*" path="*_AppService.axd" . . .
    <add verb="GET,HEAD" path="ScriptResource.axd" . . .
    <add verb="*" path="MapController.ashx" . . .
    <add verb="*" path="LayerController.ashx" . . .
</httpHandlers>
```

If any of those first four httpHandlers entries are missing from your Web.config file, copy the missing entries from the AJAXDemo Web.config file, and paste them into your Web.config file. (It is not necessary to copy the LayerController.ashx entry; if you place the LayerControl on your page in Designer mode, the LayerController.ashx entry will be generated automatically.)

7. Locate the httpModules section of your Web.config file. The httpModules section probably already contains one entry, for MapInfo.Engine.WebSessionActivator. Copy the "ScriptModule" entry from the AJAXDemo Web.config file, so that your httpModules section resembles this:

```xml
<httpModules>
    <add type="MapInfo.Engine.WebSessionActivator. . .
</httpModules>
```

If the controls in your UpdatePanel affect the map in some way, you will need to add JavaScript to the page to force the map image to update. The AJAXDemo sample application contains sample JavaScript that demonstrates how you can update the map image whenever the designated UpdatePanels cause a page update.

8. Open the AJAXDemo application's MapForm.aspx page in Source mode. Copy the <script> block and paste it into your aspx page. **NOTE:** You must paste the <script> block after the ScriptManager tag, because the script makes use of objects provided by the ScriptManager.

9. In the <script> block, delete the DisplayEventInfo function and any calls to it. The DisplayEventInfo function is a debugging tool for the AJAXDemo application; it is not needed in other applications.

10. If you renamed your UpdatePanel, edit the <script> block to use the new UpdatePanel name. (The UpdatePanel name is passed as the second parameter to the TargetPanelWasUpdated function; the default name is "UpdatePanel1".)
11. If you use more than one UpdatePanel in your application, but you do not want all of the UpdatePanels to affect the map, then you should set each UpdatePanel’s UpdateMode property to Conditional. For details, see the ReadMe file provided with the AJAXDemo sample application.

For additional details about the Web.config settings required by ASP.NET AJAX extensions, please consult Microsoft’s ASP.NET AJAX documentation.

### HTML/XHTML Validation Issues

If you create a MapXtreme web application and run the resulting HTML through a validator service, you might see the following validation error, depending on what version DOCTYPE tag you use:

```
value of attribute "ID" invalid. "_" cannot start name
```

This validation error is in reference to a hidden field with the attribute

```html
id="__VIEWSTATE"
```

The id attribute that causes this validation error is not output by MapXtreme; it is an attribute that is output by ASP.NET.

To resolve this validation error, you may need to replace the DOCTYPE tag on your .aspx page. Specifically, if you update your DOCTYPE tag to an XHTML DOCTYPE tag, the resulting page will validate, even with the id attribute shown above. (ASP.NET will wrap the offending tag in a DIV tag, all of which will validate against the XHTML DOCTYPE.) As an example, you might use the same DOCTYPE tag that is generated for you when you create a new web site from Visual Studio’s ASP.NET template:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

### Migrating from pre-v6.5 Web Controls to 2005/2008 Web Controls

The process of migrating your pre-v6.5 Web controls\(^1\) to the MapXtreme Web controls in versions 6.6 and higher is not automatic. Every application and migration process is different and depends on how tightly coupled the application is with the functionality and design of the older Web controls.

The recommended migration process is to take a phased approach. In some cases code restructuring might have to be done. The following processes need to be considered while migrating your Web controls:

- Loading Data
- Replacing Controls

\(^1\) In MapXtreme releases prior to version 6.5, the web controls required a postback page that called Page_Load and Page_Unload every time a tool was used. These controls have been replaced by more efficient JavaScript-enabled partial-page update controls. We provide the postback controls for backward compatibility. Information on these controls is contained in the MapInfo.Web.UI.WebControls namespace. For the JavaScript controls API, see the MapInfo.WebControls namespace.
Specialized Topics for Web Controls

• State and Event Management

Loading Data

The only way to load data with MapXtreme Web controls is by using a preloaded workspace. The MapControl points to the MapAlias in the workspace and tries to display the map, not load the map. The preloaded workspace is specified in the Web.config file. See What Should the Initial Map Look Like? in Chapter 6 on page 99.

In the case where the MapControl is used to display multiple map images, set up the data so that all maps are in one location, and change the MapAlias of the MapControl to choose the maps.

Replacing Controls

Replacing controls can be done in various ways, either by deleting the older Web controls from the form and dragging and dropping the new Web controls on the form, or by creating new Web forms, dragging and dropping the new controls and then start adding functionality from the old forms to the new forms. Make sure to set the appropriate MapAlias for the MapControl and set the appropriate MapControlID for all dependent tools and the LayerControl.

State and Event Management

In most cases, the older method of state management is to restore state in Page_Load and save state in Page_Unload. This code must be moved to the new class which derives from the StateManager into the RestoreState and SaveState methods.

If your web application handles state manually, then you must implement the StateManager and put this in the ASP.NET session. It is a best practice with MapXtreme to manually handle state so that you are only restoring the information needed. See Chapter 6: Understanding State Management.

In the case where Server.Transfer or Response.Redirect is used in a web page, the session is no longer new. You must put the StateManager in the ASP.NET session if it previously does not exist.

If you have created event handlers for your pre-v6.5 Web controls, for example the zoom or navigation tools, you must create custom commands and move that processing into the Process method. See Creating a Custom Tool on page 86.

Specialized Topics for Web Controls

The following sections refer to special usage of MapXtreme Web controls:

• Using Web Controls in Frames
• Using MapControl in Table Cells
• Localizing Web Controls
Using Web Controls in Frames

The MapXtreme Web controls work in frames. For information on how to create frames, framesets, and assign pages to frames see the Visual Studio documentation.

When using Web controls in frames, remember that a frame points to a web page and a frameset contains one or more frames. Take the following scenario: A page with a MapControl and other pages with tools or a LayerControl that depend upon the MapControl. All of the pages form a frameset.

Given the above scenario, the following rules apply:

• The MapControlID must be manually entered into the properties for the dependant controls. If there is a MapControl on the same page as the dependant control with the same ID, the tools and LayerControl will pick that one.

• Since the pages are drawn in a particular order, the session is not new. To make sure that the StateManager is in place, you can implement the following (notice that before restore state, session is added (if it does not exist)):

  If the StateManager doesn't exist in the session put it in, else get it.
  If StateManager.GetStateManagerFromSession() Is Nothing Then
    StateManager.PutStateManagerInSession(New AppStateManager())
  End If

Using the MapControl in Table Cells

Due to HTML behavior, as soon as an element is dragged out of another element, it is resized to a basic size (mainly 0). If you put a MapControl in a table cell you may have a problem when an element does not have an absolute width and height in the HTML. When the element is dragged out of the cell, it will collapse and therefore the cell size becomes 0.

To solve this issue, set the MapControl height and width, explicitly. The following sample shows how to set the size in the HTML:

```html
<table style="Z-INDEX: 101; LEFT: 32px; POSITION: absolute; TOP: 64px"
borderColor="#ff00ff" border="1">
  <TR bordercolor="#ff3366">
    <TD bordercolor="#0066ff">
      <cc1:mapcontrol id="Mapcontrol2" runat="server" Width="300px"
Height="300px"></cc1:mapcontrol>
    </TD>
  </TR>
  <TR>
    <TD>
      <cc1:pantool id="Pantool2" runat="server"
MapControlID="MapControl2"></cc1:pantool>
      <cc1:zoomintool id="ZoomInTool1" runat="server"
MapControlID="Mapcontrol2"></cc1:zoomintool>
      <cc1:zoomouttool id="ZoomOutTool1" runat="server"
MapControlID="Mapcontrol2"></cc1:zoomouttool>
    </TD>
  </TR>
</table>
```
Web Control Localization

It is important to understand how the Web controls handle localized resources. This section describes the basics of localization using the Web controls, and describes some scenarios for localization.

When working with the MapXtreme Web controls it is important that the following general steps are followed:

- Move all strings into the resx resource file. This is the file that will get translated. Make sure that you do not hard code in your source code.
- Generate a satellite assembly for the Web controls using this resx file.
- Install the assembly in an appropriate place on your file system so depending upon the locale of the system, resources from this satellite assembly are used.

In the MapXtreme Web controls, the following has been designed:

- There is a strings.resx resource file containing the string resources. This file is an embedded resource, and is in the assembly. In the case where resources for a particular locale are not found, the resources in this file are used (English).
- For a localized version of MapXtreme, the satellite assembly MapInfo.WebControls.resources.dll is installed in the GAC.
- The source file, Resources.cs, contains an implementation of code to get strings from resources. The method GetString is used to get strings from resources.

Consider the following scenarios:

- You can directly use translated resx files if your application is going to be installed for one language, instead of using satellite assemblies. Using satellite assemblies is good practice, as it simplifies adding and removing languages without rebuilding your code.
- Satellite assemblies are created using resgen.exe and al.exe (assembly linker). The resx file is converted into a resources file using the resgen.exe. The al.exe is used to create the satellite assembly.
- You can either put the satellite assembly in the GAC or copy the assembly where the executables. To copy the assembly, create a folder with the same name as the culture name and copy the satellite assembly into the new folder. For example, for a Japanese satellite assembly, create a ja-JP folder.
Understanding State Management

This chapter describes concepts and best practices for state management—a subject of great importance when writing web applications.

In this chapter:

- Terminology ......................................................... 96
- What is State Management? .......................... 98
- InProc Development Model ............................. 100
- State Management For Pooled Objects .......... 103
- A Detailed Look at Manual State Management . 108
- A Closer Look at the MapXtreme Session ...... 114
Overview

State management is a key consideration in the design and implementation of a MapXtreme web application. How the changes a web site visitor makes when interacting with your web application is central to a successful user experience and to building a scalable, high performing application.

Before getting too far into this subject, the following section is a "must read" for what follows in this chapter. Understanding the difference between MapXtreme Session and HTTP Session, user state and application state, clean and dirty MapXtreme Session objects, and more, will better prepare you for planning and building your web application successfully right from the beginning.

Terminology

**MapXtreme Session** - The MapInfo.Engine.Session object that holds the Catalog, MapFactory, and CoordSysFactory. The user interacts with an instance of a MapXtreme Session object.

**HTTP Session** - The System.Web.HttpSessionState object where the user's changes are saved between requests. These changes are saved and restored for each request. The user's changes are known as the user state.

**Browser Session** - The period of time a unique user interacts with the web application. This is also referred to as the ASP.NET Session.

**InProc Development model** - A web application development model in which each user has his own instance of the MapXtreme Session object. Any changes the user makes during the browser session do not affect other users. The entire MapXtreme web application is stored in memory with the current application state and is associated with this individual user. This model is useful for small, light-use web applications where the number of users is known, such as a department's intranet. Contrast this with the more scalable Pooled Development model in which users share the MapXtreme Session and system resources. See InProc Development Model on page 100.

**Pooled Development model** - A model in which multiple MapXtreme Session instances are available from a COM+ pool associated with the web application and are activated to serve web requests. Each of the pooled MapXtreme Session objects is used to handle requests from multiple web users. This model is more complicated than the InProc model, since your application must manage the state of the MapXtreme Session for each user. However, this is a more efficient use of system resources. Use this model to build scalable applications. See State Management For Pooled Objects on page 103.

**State Management** - A general term in web application development that deals with saving and restoring information from a browser session.

**Background map** - The initial map(s) that is pre-loaded with the web application as defined in the Web.config file. This map contains reference layers, such as street networks and postal boundaries, as well as application-specific data such as store locations or cell towers. This base workspace will be pre-loaded in MapXtreme Session instances and will be available unless the application allows the user to change the base maps. See What Should the Initial Map Look Like? on page 99.
**Application State** - An ASP.NET-defined mechanism for storing state information in memory that applies to all users and sessions of a web application. See the [ASP.NET Application State Overview](#). See also [Handling Initial Requests on page 112](#).

**User state** - The saved state of the MapXtreme Session and application state between user requests. Any changes the user makes to the MapXtreme Session or application must be saved to the HTTP Session. Changes can be as simple as re-centering the map, or as complicated as a query to create a thematic map.

**Beginning state** - The condition of a pooled MapXtreme Session instance when the user accesses it. There are four possible beginning states:

- New user to site, the available MapXtreme Session instance from the pool is clean, with the background map in its initial state.
- New user to site, the available MapXtreme Session instance from the pool is dirty (map has changes from another user)
- Returning user, the available MapXtreme Session instance from the pool is clean.
- Returning user, the available MapXtreme Session instance from the pool is dirty (map has changes from this or another user)

**MapInfo.Engine.Session.State** - A MapXtreme-defined mechanism that determines whether the MapXtreme Session state will be saved automatically or manually. The automatic Session state means the entire MapXtreme Session is saved to the HTTP Session. Automatic Session state is set in the application's Web.config file in the key:

```xml
<add key="MapInfo.Engine.Session.State" value="HttpSessionState" />
```

Manual Session state means that the developer is responsible for saving state that changes from one user to another. This is the mechanism to use when building scalable, pooled applications.

Manual Session state is set in the application’s Web.config file in the key:

```xml
```

**StateManager** - A MapXtreme class with methods and properties to help with saving and restoring user state in a pooled web application. When MapInfo.Engine.Session.State is set to Manual in the Web.config file, the application must provide a StateManager class that implements SaveState and RestoreState methods. See [Implementing a StateManager on page 110](#).

**sessionState** - A standard ASP.NET Web.config element for configuring which storage mechanism is used for saving user state. Three types:

- InProc - user state is stored in memory for the lifetime of the ASP.NET Session.
- StateServer - user state is saved on the server so user changes can be retrieved at a later time.
- SQLServer - user state is saved to an SQL Server database for later access.
What is State Management?

Many types of web applications need to perform state management—the process of saving and restoring information about what state, or condition, each user's session is in. For example, when a retail web application provides a shopping cart, the application must remember the state of each user's shopping cart.

In a web mapping application, if your application allows the user to click the map to zoom in, your application needs to remember that one user might be zoomed in on Europe while another user is zoomed in on Australia. The code that remembers each user's map state is referred to as state management code.

What State Management Options are Available?

MapXtreme provides you with these options for how you manage state:

- You can use InProc state management, which is easy to code; however, it is not appropriate for all applications because it taxes server resources.
- You can use a Pooled architecture with automatic state management. This model is easy to code, but the resulting application may not be fast enough, depending on your needs.
- You can use a Pooled architecture with manual state management. This model requires more coding on your part. You must write code to save and restore the appropriate MapXtreme Session objects that comprise the user's state. However, this model produces the most scalable applications, so it is the best choice for applications that have large numbers of users.

Thus, there are trade-offs associated with each option. You must consider many factors when designing web applications using MapXtreme. Some factors are MapXtreme design decisions and others are Microsoft technology design patterns. This chapter will walk you through some of the decisions you must make and show you how to make the correct choices for your type of application.

It is strongly recommended that you think about state management before you write your web application. The type of state management that you choose will affect how you write your application. If you write a web application, and then later decide to adopt a different type of state management, you may need to rewrite significant parts of your application.

Questions to Ask Before Writing Your Application

If you want to create a simple MapXtreme web application, you can do so quickly and easily - just create a new web site application using the Visual Studio template provided with MapXtreme ("MapXtreme 6.7 Web Application").

However, for a more complicated web application, before you invest significant time and effort coding the business logic for your web application, there are several questions that you should think about, as the answers to these questions will help you decide which type of state management is right for you.
How Many Users Will Access Your Application?

The most important question you can answer is: How many people will access this site? The earlier you identify the number of users your web application will have, the better off you are. If you develop and test an application with a very small number of users (as a pilot project perhaps), then later on you might discover that your application does not perform well when there are many users. In that situation, you may find that you need to change your application architecture - a change which might require you to rewrite your application. It is far better to plan ahead and anticipate your user load in advance, so that you can use the appropriate state management architecture from the beginning.

Known Number of Users

Perhaps you are creating an intranet site where you have a finite, defined list of potential users. For example, you might be creating a web site to be used by a specific department in your organization. Perhaps you know the names of everyone who will access your site. Given such a well-defined and finite number of concurrent users, you can consider using the InProc, non-pooled development model. It is the easiest model to code, but not an appropriate choice for all applications, because it places substantial demands on web server resources. It creates one MapXtreme session instance for each concurrent user. However, if your pool of users is finite and well-defined, these server requirements might not be a problem.

Unknown Number of Users

The other development model is when you do not know the number of users who will want access to the site. Perhaps you are hoping or expecting that over time your web site will attract more and more visitors. In this case, the InProc, non-pooled model would be inappropriate, and you should instead plan on using a pooled development model where you capture each user’s state manually. In the pooled development model, you will create a finite number of MapXtreme sessions, and each user request will be serviced by re-using one of the objects from the pool.

The pooled architecture lets you develop your application for scalability. You may later need to add more servers to handle additional load; thus your application must understand how to save the users current state and when and how to apply it when the next request comes in to the server. This may occur on the same server or a different server in your farm. This scalability leads to many options and choices you can make about how your system will store state, access data, and respond to multiple requests. This is why it is important to plan out your application and create a strong architecture that supports distributed applications.

What Should the Initial Map Look Like?

When you develop an application, you must decide what information remains the same for all users. This includes map layers, labels, titles, and color-coding or other types of thematic shading. This is known as the background map.

To set up your background map, run the MapXtreme Workspace Manager (a desktop application installed with MapXtreme), and save your map as a workspace file (.mws file). See Workspace Manager in Chapter 23 on page 385.
Once you have created an .mws file, edit your web application's Web.config file to include a reference to your .mws file. The following Web.config excerpt shows the syntax:

```xml
<configuration>
    <appSettings>
        <add key="MapInfo.Engine.Session.Workspace" value="C:\MIDATA\EvalData\WorldDetail\World_Detail.mws" />
    </appSettings>
</configuration>
```

This tag instructs the MapXtreme Session to load this workspace whenever a new instance of the Session is created.

**How Will Users be Allowed to Modify the Map?**

Most web applications allow the user to click, drag, or perform other actions that modify the map in some way. In your application, you might consider the following.

Will you allow your users to:

- Zoom in or out or re-center the map?
- Select some features on the map (perhaps by clicking directly on the map)?
- Turn map elements on or off (such as clearing a checkbox to turn off street display)?
- Create and/or modify thematic shading (such as color-coding on the map to show data)?
- Place map annotations, such as symbol markers, on the map?

All of these operations can be supported in a mapping application. However, if you decide to implement a web application using a pooled model, you should be aware that pooled applications need to include code that carefully saves and restores all changes the user has made to the map.

For example, if your web application allows users to place annotations on the map, then you will need to write code to save each user's custom annotations, and then write code to restore that user's custom annotations with each subsequent request.

If your application uses the pooled model with manual state management, then your state management code will grow more complex as you add more and more features that allow the user to modify the map. As you consider features that allow the user to modify the map in various ways, remember to set aside time for the development and testing of your state management code as well.

**InProc Development Model**

If you use the InProc development model, you code your application much like you would code a desktop application. With the InProc model, there is one MapXtreme Session object for each user; this means you will have the overhead of each user having their own process space, and resources are not shared. (The MapXtreme Session is the holder of the Catalog, MapFactory, CoordSysFactory, and other MapXtreme objects.)

The InProc model is a simple model for development purposes. Because each user has their own MapXtreme Session, changes can be made within a Session without disrupting any other users' Session.
For example, if the user clicks to zoom in on the map, your application code can simply modify the map object's Zoom property, without worrying about whether the change in the zoom level will adversely affect other users. Thus it is easy to code an InProc web application.

You might find it helpful to think of a web application as being analogous to a restaurant. Web applications need to service simultaneous requests from multiple users, just as restaurants need waiters to take food orders from multiple tables.

The InProc model is analogous to a restaurant that hires one waiter per table. It would be expensive for a restaurant to employ one waiter per table. However, if a restaurant was able to afford hiring one waiter per table, then doing so would certainly provide each table with excellent service. Also, having one waiter per table would make each waiter's job simpler. Since each individual waiter would be servicing one table, the waiter would not need to waste any time or energy remembering which order went to which table.

Pros and Cons of the InProc Development Model

If you choose the InProc model, it will be easier for you to write your web application, because you will not need to provide any complex code for saving and restoring each user's state with each subsequent request.

However, the InProc model is not scalable, because it requires having a dedicated MapXtreme Session for each concurrent user. For thousands of users to use your site concurrently, the application would need to maintain thousands of MapXtreme sessions on the server, which taxes your server's resources. If you anticipate a large number of concurrent users, then the InProc model is not an appropriate choice for your application.

InProc Management: A Walk-Through

An example helps illustrate how MapXtreme Sessions are created and used when an application uses the InProc model. The following describes, in simplified terms, the sequence of events:

1. A user launches a browser and goes to your mapping application web site.
2. A new MapXtreme Session is created on your server. It will be used to service all requests from this user during this ASP.NET session.
3. The workspace file(s) specified in your Web.config file is loaded. As a result, your MapXtreme Session's MapFactory has one or more Map objects. For the sake of this example, let us assume that the workspace contains one map, which initially shows the entire world.
4. The web page's HTML is rendered and returned to the client browser. Part of the page is an HTML img tag, which requests an image of the map from the server. On the server, objects from this user's MapXtreme Session—a Map object, in particular—are used to render the map image, which is returned to the user's browser.
5. The user selects the Zoom In tool, and then draws a marquee box around Australia, to zoom in on Australia.
6. On the server, the application modifies the Map object's Zoom property so that it is zoomed into Australia. The application renders a new map image and streams the image to the client browser.

   The Map object is now in a different state than it was initially—its Center and Zoom properties have changed (to show Australia rather than showing the entire world).
7. The user clicks on the Select tool, and then clicks on the portion of the map that shows Australia.
8. On the server, the application performs a selection: to select the Australia region. A new map image is rendered, to display the new selection.

   There are now two ways the Map's status has changed from its original state: Its center and zoom changed (in step 6), and now there is a Selection.

9. Elsewhere, a second user launches a browser and goes to your mapping application web site. A new MapXtreme Session is created on your server for the second user. This session contains a map, which shows the entire world. This Map object is separate from the first user's Map.

Note that the second user sees a map of the world, rather than a map that is zoomed in on Australia. Only the first user's map is zoomed in on Australia. A second MapXtreme Session (with its own map) was created to service the requests from the second user. Thus the second user will not see the same map as the first user.

### Configuring an Application to Use the InProc Development Model

The Web.config file for your web application holds the settings to control the application model. For the InProc model the Web.config file settings are:

```xml
<!--Use this setting to turn Session pooling on/off (true/false)-->  
<add key="MapInfo.Engine.Session.Pooled" value="false" />

<!--Use this setting to save Session state automatically (HttpSessionState) or manually (Manual)-->  
<add key="MapInfo.Engine.Session.State" value="HttpSessionState" />

<sessionState mode="InProc" stateConnectionString="tcpip=127.0.0.1:42424"  
               sqlConnectionString="data source=127.0.0.1;userid=sa;password="  
               cookieless="false" timeout="20" />
```

These settings specify that the MapInfo.Engine.Session object is not pooled, and the state of the MapXtreme Session object is automatically stored in the user's HTTP Session (key value = HttpSessionState).

In the sessionState element, we have set the mode attribute to InProc to specify that everything is saved in process. InProc is the default mode if sessionState is not specified.

The lifetime of the MapXtreme Session instance depends on the Session state configuration. The MapXtreme Session instance is cached in memory for the lifetime of your ASP.NET session when you use server-based Session State and the ASP.NET sessionState element in your Web.config file is set to InProc.

If you use any other type of Session State configuration (for example, client-based or server-based with sessionState element set to StateServer or SQLServer), the MapXtreme Session instance is created and disposed for each ASP.NET request. However, this has performance implications and should be avoided as a development model. If you follow the InProc model, sessionState should be set to InProc.

Once you decide to follow this option of an InProc session, each user who accesses your web site will be given a copy of the MapXtreme Session object and all it contains. Obviously if the number of users grows, then so will the memory footprint.
Your application will also load the workspace defined in the Web.config each time a MapXtreme Session is created. For the InProc model, this means a new MapXtreme Session is created and the workspace is loaded when the user first visits the site.

**Using the MapXtreme Template with the InProc Development Model**

If you created a web application using the "MapXtreme 6.7 Web Application" template, be aware that it does not follow the InProc model.

To create an application with the template that uses InProc:

1. Modify the Web.config file as shown in Configuring an Application to Use the InProc Development Model on page 102.
2. Remove the application's Page_Load and Page_Unload methods from MapForm1.aspx.vb or MapForm1.aspx.cs or comment out the manual state-management code that was placed automatically by the MapXtreme Web Application template.

**State Management For Pooled Objects**

State management in a pooled application is a development model that is designed to make applications scalable. If you anticipate that your application will have a large and/or a continuously growing number of users, the pooled model is a better choice for you than the InProc model.

**What is Pooling?**

In the pooled model, your application creates a finite number of MapXtreme Session objects on the server. Each one of those object instances is then shared within the application and re-used; each MapXtreme Session handles requests from multiple users, as illustrated in the following diagram.

![Diagram of pooled application](image)

This diagram represents an application that has a COM+ pool of two MapXtreme Session objects, being accessed by three users (web clients). Whenever a client submits a request, the request will be serviced by one of the two MapXtreme Session objects.
State Management For Pooled Objects

In this model, no client has its own dedicated MapXtreme Session. Instead, MapXtreme Session objects are shared and re-used. The following sequence of events is possible:

1. Client1 might request a map zoomed in on Africa. This request might be handled using SessionObj1.
2. Client3 might request a map zoomed in on Australia. This request might also be handled using SessionObj1. In other words, each Session handles requests from multiple clients. No one client "owns" the MapXtreme Session; they are all shared.
3. Client1 might submit a request to pan the map to show Europe. This request might be handled using SessionObj2. In other words, there is no guarantee that every request submitted by a particular client will be serviced by the same MapXtreme Session, or the same Map object.

Pros and Cons of Pooling

Pooling helps reduce the number of concurrent MapXtreme Session instances, and thereby reduces the overhead of creating each object from scratch. When an object is needed, it is pulled from the pool. When the object is not needed anymore, it is placed back into the pool to await the next request. Pooling in MapXtreme means that multiple MapXtreme Session objects can already be created and loaded with a background map and available when a web request is received.

Pooling helps you optimize the server's resources in the following ways:

- Improves overall response time for web applications by running several ASP.NET requests concurrently.
- Conserves resources by reducing the number of concurrent MapXtreme Session instances.
- Maximizes CPU utilization by reducing the number of requests running at the same time (and reducing thread context switching). This is particularly useful when requests are CPU intensive (for example, Map image exports). The general recommendation is to have a pool size of 1-2 session instances per CPU.

However, a pooled application can make an application more complicated. MapXtreme Sessions (and their maps) are shared and re-used; therefore, you need to take steps to ensure that each Map is re-set to an appropriate state each time your application handles a client request.

For example, suppose a client requests a map of Germany. You cannot simply fetch a MapXtreme Session from the pool and then use its Map to render an image, because there is no guarantee that particular map is zoomed in on Germany. The Map that services the request must be set to an appropriate, known state. As the application developer, you must manage the state of the object to ensure that the object is in an appropriate state before you use it to service the current request.

Saving State for Pooled Applications

The pooled development model offers you two options for managing the state of your pooled objects. You can use automatic state management, and have MapXtreme take care of the current user state, or you can use manual state management and control the changes that must be saved and restored for each user.
Automatic State Management

This scenario indicates that the state of the MapXtreme Session object will be saved automatically, to the session mechanism defined in the Web.config (either StateServer or SQLServer). The application programmer does not have to programmatically save any of the current user state information, such as layers, thematics, open tables, or current zoom. The entire state of the MapXtreme Session object will be saved.

This option saves system resources compared to the InProc model, since you do not have a MapXtreme Session for each user and all users will share pooled MapXtreme Session objects. The drawback is that the entire MapXtreme Session will be serialized to the currently defined state mechanism. In general, the MapXtreme Session object retrieves all objects in its factories and serializes them to the state mechanism. This includes all maps and their included layers, all open table definitions, and any projections loaded. If the application is using automatic state management, the MapXtreme Session object does not know about individual user’s state and stores all available information. This includes layers that do not change from one user to another, which is the background map. This can be a time consuming process and can cause this type of application to perform more slowly than an InProc model. For information on serialization see Serialization in Chapter 9 on page 154.

Manual State Management

This configuration means the MapXtreme Session instance will not automatically save any instance data. It is up to the application programmer to include code in the application to save and restore the state for each user. The MapXtreme Session object will not store any of its state to the currently defined state mechanism.

This configuration provides for the greatest gain in performance and scalability. You are using pooled MapXtreme Session instances that do not save any state, and these pooled objects are being returned to the COM+ pool "dirty". By this we mean the pooled object has any modifications to the map done by the current user and your application is responsible for cleaning up the state and setting it correctly for the next user. You can choose to clean up at the beginning of your web application request or the end. Implementing a StateManager on page 110.

The combination of using pooled MapXtreme Sessions and managing user state manually makes this a complicated development model. You cannot tell which MapXtreme Session will be used for a particular request; however, at the same time you need to satisfy the user’s request with a MapXtreme Session that is specific to that user. For this reason, your application must be aware of the beginning state of the pooled MapXtreme Session.

There are four possible beginning states that your application must handle. Each situation involves a user condition and a MapXtreme Session state.

<table>
<thead>
<tr>
<th>User is...</th>
<th>MapXtreme Session in the Pool is...</th>
</tr>
</thead>
<tbody>
<tr>
<td>New to the site</td>
<td>Clean *</td>
</tr>
<tr>
<td>New to the site</td>
<td>Dirty †</td>
</tr>
</tbody>
</table>
Manual State Management: A Walk-Through

An example helps to illustrate how pooled MapXtreme Sessions are managed when an application uses the manual state management model. The following describes, in simplified terms, a possible sequence of events:

1. A user launches a browser and goes to your mapping application web site.
2. Since this application uses pooling, a MapXtreme Session instance is retrieved from the pool.
3. The RestoreState method is called from Page_Load.
   If the MapXtreme Session is "dirty," with some user's changes, the application sets the map back to the desired beginning state; see the RestoreDefaultState method in the AppStateManager class.
   For this example, let us assume that the beginning state shows a map of the entire world.
4. The web page's HTML is rendered and returned to the client browser. Part of the page is an HTML img tag, which requests an image of the map from the server. On the server, objects from this user's browser session—a Map object, in particular—are used to render the map image, which is returned to the user's browser.
5. In the Page_Unload method, the application saves the user's state, by calling the AppStateManager.SaveState method.
6. The user selects the Zoom In tool and draws a marquee box around Australia. Javascript that is built into the Zoom In tool submits a request to the server to zoom in on Australia.
7. On the MapXtreme server, the application once again must fetch a MapXtreme Session object from the pool. Note that the MapXtreme Session returned from the pool may be a different object than was used to service the previous request. This object could be clean or dirty.
8. On the server the application calls the AppStateManager.RestoreState method, to restore the user state that was saved in step 5. This restores the MapXtreme Session to a known state that is appropriate for this user.
9. The application modifies the Map object's Zoom property so that it is zoomed into Australia. The application renders a new map image and streams the image to the client browser. Again, the application saves the user's state at the end of the request.
10. The user clicks on the Select tool, and then clicks on the Australia region of the map, to select Australia.
11. The application again fetches a MapXtreme Session from the pool, and again calls RestoreState to set the pooled object to a known state.
12. On the server, the application carries out the operation to select Australia. A new map image is rendered, to display the new selection.

### Table

<table>
<thead>
<tr>
<th>User is...</th>
<th>MapXtreme Session in the Pool is...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returning to the site</td>
<td>Clean *</td>
</tr>
<tr>
<td>Returning to the site</td>
<td>Dirty †</td>
</tr>
</tbody>
</table>

* MapXtreme Session and its background map are in their initial state.
† MapXtreme Session and/or its background map contain another user's changes.
13. On the server, the application again calls AppStateManager.SaveState, to save the user state. Since this example allows the user to perform selections, the AppStateManager's SaveState and RestoreState methods will need to include code to save and restore Selections.

Any aspect of the MapXtreme session that you allow a user to change, such as layers, themes, queries, map views, must be handled in your code to save and restore each item.

Using pooling and managing user state manually is beneficial in that it improves efficiency and allows applications to be scalable. However, this model does require additional work, in terms of saving the user state at the end of each request, and then restoring that state at the beginning of that user's next request.

Configuring a Pooled Application to Use Manual State Management

For examples of a pool application using manual state management, see the sample web applications that ship with MapXtreme. The following are the main points for configuring such an application.

Configure pooled objects in the Web.Config file using the settings:

```xml
<!--Use this setting to turn Session pooling on/off (true/false)-->
<add key="MapInfo.Engine.Session.Pooled" value="true" />
<!--Use this setting to save Session state automatically (HttpSessionState) or manually (Manual)-->
<sessionState mode="StateServer" stateConnectionString="tcpip=127.0.0.1:42424" sqlConnectionString="data source=127.0.0.1;user id=sa;password=" cookieless="false" timeout="20" />
```

These two settings can be combined into the following options:

- For pooled objects and automatic state management, set Pooled to true, and set State to HttpSessionState.
- For pooled objects and manual state management, set Pooled to true, and set State to Manual.

Place the logic that determines the user's current state into the Page_Load method. See Implementing a StateManager on page 110.

User State

Your application must now consider the user's current state and the state of the MapXtreme Session object when it was retrieved from the pool. In this situation you will have to decide:

- Do you return the pooled object clean to the pool in your SaveState method?
- Do you let go of the pooled object and then restore any state in the RestoreState method?

The correct choice here is not always clear since many variables about the difference in state from one user to another may lead you to leave the session object dirty prior to releasing it back to the pool. Another application may save time by waiting for the Page_Load to clean up or to check if the Session needs cleaning.

The example in the section Manual State Management: A Walk-Through on page 106 followed the second option here: to clean up the pooled object in the RestoreState method.
A Detailed Look at Manual State Management

The next section provides more information on how to save and restore state.

A Detailed Look at Manual State Management

This section provides a detailed example of how a pooled application can perform manual state management. We will examine relevant sections of code from the Thematics sample web application, which is installed as part of MapXtreme. If you have not already done so, you might want to run the Thematics sample to familiarize yourself with it, before reading this section.

The Thematic sample is one of the projects included in the solutions here:

C:\Program Files\MapInfo\MapXtreme\6.x.x\Samples\VisualStudio2005\Web\Features

This discussion covers the following topics:

- Overview of the Thematics Sample
- Application Settings
- Implementing a StateManager
- Serializing MapXtreme Objects in the Proper Order
- Automatically Deserializing MapXtreme Objects
- Handling Initial Requests
- Handling Subsequent Requests

Overview of the Thematics Sample

This application displays a map of the world, loaded from the workspace World.mws (part of the sample data installed with MapXtreme).

Options on the web form allow the user to create various types of thematic shading on the map. These themes modify the appearance of regions (country boundaries) in the table world.tab. The attribute data for the themes comes from a Microsoft Access database, eworld.mdb (included within the Visual Studio project).
In this sample, the world.mws workspace file is pre-loaded when the MapXtreme Session is created; however, the columns from eworld are added later, when a client accesses the application.

**Note**  To get better performance, put attribute column information into a workspace file, so that all needed data will be pre-loaded at the MapInfo Session creation time.

**Application Settings**

First let's take a look at the parts of the Web.config file that contain relevant application settings. Near the top of the Web.config file you will find these settings:

```xml
<configuration>
  <appSettings>
    <!--Use this setting to turn Session pooling on/off (true/false)-->  
    <add key="MapInfo.Engine.Session.Pooled" value="true" />

    <!--Use this setting to save Session state automatically (HttpSessionState) or manually (Manual)-->  

    <!--Use this setting to preload a workspace on Session creation-->  
    <add key="MapInfo.Engine.Session.Workspace" value="c:\Program Files\MapInfo\MapXtreme\6.7\Samples\Data\World.mws" />
  </appSettings>
</configuration>
```

The settings are explained below:

- **MapInfo.Engine.Session.Pooled**: Because this key has a value of "true", MapXtreme Session objects are pooled by the COM+ service. When a new MapXtreme Session is retrieved upon a new client request, the Session could be "clean" - a new object just created - or it could be a "dirty" object which has already been used to handle a previous request. Because of the uncertain state of this object, this application will take steps within each request to restore the MapXtreme Session to a known state before using it.

- **MapInfo.Engine.Session.State**: Because this key has a value of "Manual", the application is responsible for explicitly saving and restoring Session state. The Manual setting aids performance, because the application’s StateManager class is written with intelligence about which objects it should save and restore. The StateManager selectively saves and restores only the minimum number of objects that are needed; that is why it is the most efficient state management option. (See Implementing a StateManager on page 110).

- **MapInfo.Engine.Session.Workspace**: This setting lets you specify a semi-colon separated list of.mws workspace files, which will be pre-loaded at the time the MapXtreme Session is created. To specify multiple workspaces, separate the filenames with semi-colons.

Farther down in the Web.config file, you will find this setting:

```xml
<sessionState mode="StateServer"
  stateConnectionString="tcpip=127.0.0.1:42424" sqlConnectionString="data source=127.0.0.1;user id=sa;password=" cookieless="false" timeout="20" />
```

This indicates how you wish to save state; in this case, to the StateServer where the information can be retrieved at a later time. You can also set this to SQLServer, where state is saved to a database.
Implementing a StateManager

The Thematic sample application uses manual state management, which means it requires an implementation of a StateManager class to save and restore the appropriate MapXtreme Session changes.

This class inherits from the abstract base class MapInfo.WebControls.StateManager and must implement the following methods:

```vbnet
Public Overrides Sub RestoreState()
Public Overrides Sub SaveState()
```

**Note**  If you use the "MapXtreme 6.7 Web Application" template to create a new web site in Visual Studio, the resulting project includes a simple implementation of the StateManager, called AppStateManager.

The exact contents of the AppStateManager class vary from application to application. The more options your application provides, allowing the user to customize the map, the more code you need to add to the AppStateManager, to save and restore those customizations. This is why the AppStateManager that is provided with the Thematics sample application contains more code than the AppStateManager that you get when you create a new project from the MapXtreme Web Application template.

For example, the thematic sample handles customizations to theme layers, theme tables, attribute tables, and a group layer. The web template, in contrast, has code to handle customizations to layers and selections.

The RestoreState and SaveState methods are used as follows:

1. Each time a client submits a request, the Page_Load method calls RestoreState. RestoreState is also called each time a map tool is used on the client. The RestoreState method ensures that the MapXtreme Session object—which was retrieved from the pool in an unknown, possibly "dirty" state—is restored to a known state, either to the user’s state, if it exists, or to the default state from Application state.

For example, if the user's map was zoomed in on France the last time the user requested a map, then the RestoreState method will ensure that the map retrieved from the pool is returned to a "zoomed in on France" state. This way, the user's session can continue from where it left off.

The following VB code example is from the Thematic sample application’s WebForm1.aspx.vb.

```vbnet
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As _
        System.EventArgs) Handles MyBase.Load
    ' The first time in
    If Session.IsNewSession Then

        '*******************************************************************************
        '// You need to follow below lines in your own application.
        '*******************************************************************************
        Dim stateManager As New AppStateManager

        ' tell the state manager which map alias you want to use.
        ' You could also add your own key/value pairs, the value should be
        ' serializable.
```

```vbnet
//*******************************************************************************
// You need to follow below lines in your own application.
//*******************************************************************************
Dim stateManager As New AppStateManager
```

```vbnet
't tell the state manager which map alias you want to use.
' You could also add your own key/value pairs, the value should be
'serializable.
```
chapter 6: understanding state management

' Put state manager into HttpSession, so we could get it later on from ' different class and requests.

MapInfo.WebControls.StateManager.PutStateManagerInSession_(stateManager)
Me.InitState()
End If
MapInfo.WebControls.StateManager.GetStateManagerFromSession.RestoreState()

PrepareData()
End Sub

2. After the Page_Load method, the application applies whatever business logic is appropriate, given the nature of the client request.
   For example, if the request was generated by the user clicking with the Zoom In tool, the business logic adjusts the map's center and zoom level.

3. At the end of the request processing cycle the Page_Unload method calls SaveState. SaveState is also called after a map tool is used on the client.
   The SaveState method saves the map's latest state, so that when and if the user submits yet another request, that request will be able to call RestoreState again. Each call to SaveState is performed in anticipation of a possible subsequent call to RestoreState, the next time a request is received.

Private Sub Page_UnLoad(ByVal sender As Object, ByVal e As _
    EventArgs) Handles MyBase.Unload

    MapInfo.WebControls.StateManager.GetStateManagerFromSession()._SaveState()
End Sub

serializing mapxtreme objects in the proper order

In the AppStateManager class provided with the Thematics sample, the SaveState method calls the ManualSerializer.SaveMapXtremeObjectIntoHttpSession method several times. The objects are saved (serialized) in the following order:


ManualSerializer.SaveMapXtremeObjectIntoHttpSession(MapInfo.Engine.Session.Current.Catalog(SampleConstants.ThemeTableAlias), "theme_table")

ManualSerializer.SaveMapXtremeObjectIntoHttpSession(map.Layers(SampleConstants.ThemeLayerAlias), "theme_layer")

ManualSerializer.SaveMapXtremeObjectIntoHttpSession(map.Layers(SampleConstants.GroupLayerAlias), "group_layer")

It is important that these SaveMapXtremeObjectIntoHttpSession calls occur in the correct order, because some objects are dependent on other objects.
A Detailed Look at Manual State Management

In this example, first we save tables (which are referenced through the Catalog). Next, we save layers (which are referenced through the map’s Layers collection). That is the appropriate order, because most types of layers are dependent on tables.

For example, each FeatureLayer is associated with a table. The table must be open before you can instantiate (or de-serialize) a FeatureLayer that uses that table. Therefore, this AppStateManager class saves the tables first, and then saves the layers.

Similarly, in the RestoreState method, the data objects are restored first, followed by calls to restore the layers:

```csharp
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("mdb_table")
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("theme_table")
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("theme_layer")
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("group_layer")
```

Automatically Deserializing MapXtreme Objects

In the SaveState method, we identify the objects to be saved using specific references, such as a specific layer in the map:

```csharp
ManualSerializer.SaveMapXtremeObjectIntoHttpSession(map.Layers(SampleConstants.ThemeLayerAlias), "theme_layer")
```

The RestoreState method, however, does not reference the Catalog or the map.Layers collection. In the RestoreState method, we call methods on the ManualSerializer class, but we do not do anything with the results from that call, and there is no reference to a map.Layers collection: They are deserialized automatically to the appropriate place in the Catalog or Layers collection, or wherever necessary.

```csharp
ManualSerializer.RestoreMapXtremeObjectFromHttpSession("theme_layer")
```

**Note** Use the ManualSerializer class to carry out saving and restoring objects. Do not store MapXtreme objects directly into the HttpSessionState using syntax such as:

```csharp
```

Handling Initial Requests

Application state is a server-side state management mechanism that is used to store information that is common, or global, to all user sessions. Application state is initialized when Session.IsNewSession is true. Application state is an important element when handling initial requests.

In the case of MapXtreme, Application state holds the initial state of a background map so that any user’s initial request will receive the map in this default state.

The MapXtreme Thematics sample demonstrates the steps to take to handle initial requests.

There are two different types of “first” requests that your application must handle: 1) the first request from the very first user to access the web application, and 2) first requests from all subsequent users.
The very first time any user uses the application, a clean MapXtreme Session is retrieved from the pool with a background map in its initial state. This initial state is stored in Application state and used like a template. This happens in the SaveDefaultState method.

The first request from all subsequent users makes a call to RestoreDefaultState, which retrieves the initial state of the map from Application state.

The following are the steps related to initial requests.

1. In the Page_Load method, check to see if the current session is new. If it is, instantiate your StateManager and add the MapAlias into the state manager's ParamsDictionary property. This is mandatory when using Manual state management.

2. Next in the Page_Load method, since this is a new session, we initialize application data by calling the InitState method. In the InitState method, we prepare the initial state of the map.

3. Next, in the Page_Load method, we make a call to RestoreState to check whether this is the first request for this application.

   If this is the first request for the application, we call SaveDefaultState method to set various properties on the map to their initial state and we call a ManualSerializer method, to store this initial map state in a byte array into HttpApplicationState.

   Conversely, if this a user's first request, but not the very first request for the application, we make a call to RestoreDefaultState to reset the properties to their initial state.

Handling Subsequent Requests

After the user's first request has been handled, he or she will likely submit additional requests, such as clicking the map to zoom in. This time when RestoreState is called (either in the Page_Load method or from the map tools), the initial request logic will be skipped. Instead the code that restores the user's state is executed.

At the end of the user's session, or after using a map tool, the SaveState method is called to save the state of the session.

Note This Page_Unload method will not get called if you use Server.Transfer or Response.Redirect with endResponse flag as true, because those methods will ignore this event handler. You could explicitly invoke MapInfo.WebControls.StateManager.GetStateManagerFromSession().SaveState() before processing Server.Transfer or Response.Redirect with endResponse flag as true to work around this issue.
A Closer Look at the MapXtreme Session

The MapXtreme Session is the starting point for all MapXtreme applications. It manages the initialization of resources needed for a MapXtreme application. The Session also provides access to other important objects such as Data.Catalog, MapFactory, CoordSysFactory, Selections, and others.

To access the MapXtreme Session, call the MapInfo.Engine.Session.Current() method. Each thread in your process has a MapXtreme Session object available. There can be only one MapXtreme Session per thread and this session cannot be shared by multiple threads.

Web applications can have one MapXtreme Session object per user, or pooled MapXtreme Sessions available to many users. The following section describes the COM+ pooling options in more detail.

For more information on web application performance, see Chapter 7: Web Application Performance.

Configuring Microsoft COM+ Object Pooling

The MapXtreme Session object is registered with the COM+ services on your system. This system handles configuration and activation of any registered pooled objects. The runtime installer has the logic to register and create default settings for the MapInfo Session object. By default the MapXtreme Session object is configured with two pooled objects and a 60 second time-out.

Configuring the Pool Size

The MapXtreme Session is configurable using the system configuration methods for the appropriate section. These settings are available in both the system dialogs in the Control Panel and the .NET configuration files of your application. Web applications must manage these settings when using pooled MapXtreme Sessions.

One important setting is the number of pooled objects created by the COM+ system to service your running application. This setting is accessed using the Control Panel > Administrative Tools > Component Services. Browse through the Component Services > Computers > My Computer > COM+ Applications > MapInfo.CoreEngine > Components. Here you will find the MapInfo.Engine.Session+PooledSession object. Right click the icon and select Properties.

The Property dialog allows you to set various properties including the number of pooled objects per application and the activation time-out for session creation. Setting the number of pooled objects correctly directly impacts your applications performance. To set the number of pooled objects correctly you need to understand your application and how it accesses data.

Background Map Affects Performance

Another aspect of the session that can directly impact your application's performance, is the startup workspace that you define in the application Web.config file. This workspace defines which maps, layers, and tables will be loaded when the session object is created. This happens in the COM+ pool thread and outside your web application space.
When the application starts, a request is made to the application pool for a MapXtreme Session object. The COM+ pool then creates the number of pooled objects you specified in the control panel Component Services dialog box. This MapXtreme Session object reads the Web.config file, and the specified workspace is loaded before your application gets an instance from the pool. Therefore, if loading the workspace takes more time than the time-out specified in the Component Services dialog box, you will receive a COM+ activation exception. So it is important to understand what you are loading in this workspace and how long it will take to completely load.

If you specified a Minimum Pool Size of 2, then the COM+ pool will create both copies of the MapXtreme Session instance on the first request for a pooled object. When the COM+ pool senses a time-out waiting for a pooled object, it will create more objects up to the maximum setting. If your workspace is very complex and takes a great deal of time to load, then the activation of subsequent objects may cause time-out errors when the application is active. The web application pool has settings to control the reactivation of pooled applications, including settings for recycling the pooled classes. So at some point your application and the pooled objects may be recycled, and this will cause another workspace loading.

You must decide which maps, tables, and layers are important to have as the background for your application, and load these via the workspace in the Web.config file. But be aware of the implications of a large workspace that takes time to load. To test your load time, simply load the workspace in Workspace Manager on the server machine to determine the actual load time and adjust your time-out setting accordingly.

Initial workspaces are very useful in loading maps, layers, and tables that do not change from user to user, but you have to be aware of the implications of loading complex workspaces.

**Further Reading**

The MapXtreme Session is highly configurable and is designed to work in conjunction with Microsoft COM+ and web technologies. To better understand how best to use this, you need to understand the Microsoft web delivery architecture. There are many places to get more information on the topics listed above. Here are some links to very good information on web architecture:

<table>
<thead>
<tr>
<th>IIS 6.0</th>
<th>Internet Information Services and Internet Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>IIS 7.0</td>
<td>IIS 7.0: Managing Web Applications in IIS 7.0</td>
</tr>
<tr>
<td>COM+ pooling</td>
<td>COM+ Object Pooling Concepts</td>
</tr>
<tr>
<td>IIS Session state</td>
<td>Selecting the Method for Maintaining and Storing ASP.NET Session State (IIS 6.0)</td>
</tr>
</tbody>
</table>
Web Application Performance

This chapter focuses on performance tuning for web applications. It builds upon the information presented in the previous chapter on state management and pooling.

In this chapter:

- Web Application Performance ......................... 118
- Performance Tuning ................................. 118
- IIS Server Settings ............................... 120
Web Application Performance

Performance is always the most difficult aspect to predict. Each application has specific nuances that change the overall performance footprint. This section explains how some of the server settings affect performance and how to apply these settings to different types of web-based applications.

Note that the sections below refer to testing scenarios using Internet Information Services (IIS) 6.0, which runs on Windows Server 2003-based systems. This offers the best performance options. This section does not describe performance issues for IIS 5.0 on XP systems.

How Many Users?

CPU and memory are the most important performance aspects of any server, and more of each will improve performance. But since there are usually restrictions on hardware setups, you need a good heuristic to determine the most important hardware needs.

Ideally, when designing a system, you should first examine the number of users and space requirements for your application, then look at what hardware you need to purchase. See Questions to Ask Before Writing Your Application on page 98 for information on the number of users and their affect on web application scalability.

Performance Tuning

The decision to use a pooled or non-pooled development model for your application will have a significant effect on performance. Think of a pooled object as a shared resource for you applications to use. This helps limit the memory requirements for your application.

The following sections are testing experiences for both pooled and non-pooled applications in terms of memory usage, CPU usage, data access and rendering operations.

Pooling and state management is covered in detail in Understanding State Management in Chapter 6 on page 95.

Memory

The memory footprint for a MapXtreme web application can vary quite a bit depending on the number of consecutive users and the type of development model used (pooled or non-pooled). As an example, suppose the MapXtreme Session object, including the background map, takes up 50 MB of memory, and suppose the application creates one or two changeable layers that take up 1 MB of memory. If you are using the non-pooled development model and you have 100 consecutive users, your application would require 5100 MB of available memory.

\[
\text{Number of Users} \times \text{Session Memory} = \text{Total Memory Required}
\]

\[
100 \times 51 \text{ MB} = 5,100 \text{ MB}
\]
On the other hand, suppose your application is a pooled application in which two pooled MapXtreme Session objects are shared. With manual state management, the changeable layers for each user would be saved in HttpSessionState. The total memory footprint for this scenario assuming 100 consecutive users would be 200 MB.

<table>
<thead>
<tr>
<th>Pool Memory</th>
<th>100 User Layers</th>
<th>Total Memory Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 MB</td>
<td>+ 100 MB</td>
<td>= 200 MB</td>
</tr>
</tbody>
</table>

You can see that in the first scenario, a MapXtreme Session for each user is not scalable. But the second scenario allows us to store only what is necessary for each user. This makes it much easier to manage system resources as the number of users increases. If the number of users never exceeds ten, then it may not be worth the effort to manage state information. But if you ever opened this application to a larger audience, then you would probably have to rewrite the application to handle state management.

In order to understand how to set the number of pooled objects in an application, you must understand the nature of your application and how it accesses data. Our suggestion of two pooled object per CPU is based on accessing local data from native TAB files.

**CPU Usage**

The recommended number of pooled objects can vary depending on your specific application. We tested three different scenarios for which we have made recommendations. The recommendations, which are outlined in this section, are meant to serve as a guide. You may have to make adjustments to get the best performance for your specific application.

One of the factors in determining pool size is CPU context switching. As your pool size grows, you have more processes competing for CPU cycles, which can increase the amount of context switching occurring in the CPU. The overhead required for this context switching can degrade performance.

In our first tested scenario, the web application is accessing data from local TAB files. This method of retrieving data is processor intensive, so as the number of pooled objects increases, so does the amount of context switching. For this scenario, we found that two pooled objects per CPU was optimal.

In our second tested scenario, we are accessing data from a remote server. This method of retrieving data is less processor intensive because the process spends more time in a waiting state. The recommendation in this case is to gradually increase the pool size while monitoring the CPU usage. Both the middle and service tier servers must be monitored as the settings are changed to determine optimal settings. In this scenario, you must also determine if your database server can handle the multiple consecutive requests without degrading its performance.

A third tested scenario involved accessing WMS servers in which all the data access and rendering is done on a separate server from the IIS system. In this scenario, the web application CPU usage is very low because the WMS server is doing most of the work. The web application process spends much of its time in a waiting state, plus in this example the WMS server can only handle two
consecutive requests at any one time. Increasing the pool size here will help increase performance. Although increasing the pool size will help, the overall performance of this web application will be limited by the WMS server.

As you can see that there is no easy, one size fits all solution for pooled object tuning. You will achieve the best results by knowing your application and your data access methods, and measuring CPU usage numbers.

**IIS Server Settings**

IIS has many settings that allow you to fine tune how your application works. The following MSDN articles provide a detailed discussion of the IIS architecture:

**IIS 6.0 Documentation (IIS 6.0)**

Specifically look at the section IIS 6.0 Deployment Guide and under the Appendix A you will find many of the configuration settings explained in detail.

A good place to look for performance tips is:

**Performance Tuning (IIS 6.0)**

There are many other online resources online for information about IIS 6.0. The following sections describe the IIS settings make the most significant differences for MapXtreme applications.

**Application Pools**

An Application pool is defined by Microsoft as “A grouping of one or more URLs served by a worker process”. Application Pools in IIS 6 allow you to:

- Isolate individual web applications so that one application does not adversely affect the other.
- Group URLs to run under the same ID token.
- Control performance and health of these groups.

You can create application pools using the IIS manager. Once created you can start assigning virtual directories to specific pools and configure them to the specific application. If we look at the general settings for pools we see the categories:

- **Recycling**: administer the recycling of worker processes. In worker process isolation mode, you can configure IIS to periodically restart worker processes in an application pool, allowing you to precisely manage faulty worker processes. This ensures that specified applications in those pools remain healthy, and that system resources can be recovered.
- **Performance**: configure the way that IIS shuts down idle worker processes, the number of requests waiting to be processed, CPU monitoring, and the number of processes in a Web garden.
- **Health**: configure worker processes’ health monitoring. You can set limits for communicating with the worker processes, determine a course of action if a number of processes rapidly fail, and set startup and shutdown times.
- **Identity**: The identity of an application pool is the name of the account under which an application pool’s worker process runs.
Recycling allows the administrator to configure criteria for recycling the worker processes for this pool. In general you can recycle the process after a set number of minutes or requests, at specific times, or based on memory usage. You can configure each process independently or have multiple conditions.

Our testing has shown that recycling is best determined based on the traffic for your site and your observations of the system. Recycling does not mean an interruption in service; current users will finish a normal cycle before the process is shut down and a new one is started. The Recycle setting is best set based on your observations of your system and usage. You may want to recycle based on slow traffic times or perhaps on memory for applications that are memory-intensive. Recycling is necessary for long-running processes, and the application pool makes this easy to accomplish.

**Performance** allows the administrator to set options to shut down non-active processes, limit the number of queued requests, monitor the CPU usage and take actions, and create many worker processes for a single pool. Each of these has its own effect on the running system.

Our testing has shown that when serving WMS requests, waiting for the WMS server to respond causes thread pool saturation based on default settings. When you are stressing your system, you may notice that response times are slow and the server CPU is not fully utilized. Our testing showed a 35 percent CPU rate when running our stress test. This means the CPU is mostly waiting for something to happen.

An increase in the number of pooled objects freed up more threads to be used to service these requests. This gave better performance numbers, but the perceived performance was worse. That is, if you opened a web browser and viewed the same page there would be a long delay. In this scenario you were queued up based on the one worker process and its available threads, and this produced delays. After increasing the number of worker processes, the pure performance declined a small amount but the perceived performance was much better. The individual web request did not have to wait in a long queue; it jumped to a shorter line. The response it faster, but there is a small trade-off in performance because of context switching in worker processes.

**Health** allows the administrator to have the system do self diagnostics. Settings allow checking of running processes to see if its still active. You can also check for a certain number of failures and recycle if this falls outside the defined bounds. You can also set time limits on how long it should take to start and shut down.

**Identity** allows the administrator to set which user ID token this application pool will run under. This is helpful if some running applications need different credentials than other (which is typically the case).

In general, you get the best performance gain by changing the settings in the IIS Performance section. Remember that these settings are very application-specific and a setting that improves performance for one application may actually hinder performance for another.
Desktop Applications, Controls, Dialogs, and Tools

This chapter provides information on planning a desktop application, as well as, an overview of the myriad desktop controls, dialogs, and tools available in MapXtreme.

In this chapter:

- Planning a Desktop Application ......................... 124
- MapInfo.Windows.Controls Namespace ................... 126
- Key Controls to Use in Desktop Applications ........... 127
- MapInfo.Windows.Dialogs Namespace .................... 132
- Customizing Controls and Dialog Boxes ................ 137
- Overview of the MapInfo.Tools Namespace .............. 137
- MapXtreme Desktop Tools API ............................ 139
- Customizing Tools ....................................... 144
- Tool Events ........................................... 145
- Editing a FeatureGeometry with the Select Tool ....... 146
Planning a Desktop Application

MapXtreme is a .NET-based object model that exposes mapping functionality to any .NET-based development work. There are many emerging technologies that can be used to develop applications using the .NET framework.

Desktop applications provide a rich tool interaction for the user. While web applications are getting more sophisticated, applications designed for the desktop still have the advantage. For example, desktop applications have a “snap to node” capability that can be useful when placing a Point feature exactly at the intersection of the street. Desktop applications also offer InfoTips as you hover the mouse pointer.

If you have an existing WinForms or other .NET-based application framework, you can simply start integrating the MapXtreme classes into your application. The MapXtreme object model is a fully compliant .NET object model, so you have the full power and functionality of the .NET framework to interact with our object model. Simply develop your application in the mode defined by your application framework and reference the MapXtreme objects.

Best Practices for Desktop Applications

The MapXtreme Map Application template is designed for rapid prototyping only, since it only references the MapXtreme assemblies. Use this as a starting point for a quick demonstration or proof of concept.

WinForms and the .NET framework provide basic support for single document (SDI) and multiple document interface (MDI) based applications. WinForms does not provide an application framework comparable to Microsoft Foundation Classes (MFC). It is missing document view, data exchange, and other MFC-based user interface concepts.

However, Microsoft does provide its Composite UI Application Block (CAB), as a way of encouraging their recommended patterns and practices. Note that the Composite UI Application Block is available for .NET 2.0 Framework and above. More information is available at the Microsoft Developer Network website at [http://msdn2.microsoft.com/en-us/library/aa480450.aspx](http://msdn2.microsoft.com/en-us/library/aa480450.aspx).

A very important aspect of the .NET development model is the separation of business logic from presentation. MapXtreme uses this paradigm in its controls to allow the user to open a dialog with a copy of the object to be modified. In this way, the dialog is not directly editing your runtime objects and you can create a new object in another process or machine and pass back the serialized version for use by your application.

MapXtreme and COM

.NET does not mean the end of COM. Many development shops put a great deal of effort into the COM object model and have a large investment that can’t simply be discarded. So what is the best method of developing applications that have COM objects?

It is more likely that new .NET code will have to interoperate with existing COM code (rather than the reverse situation), so we will focus on this scenario. A .NET client accesses a COM server through a runtime-callable wrapper (RCW). The RCW wraps the COM object and mediates between it and the
.NET common language runtime (CLR) environment, making the COM object appear to .NET clients as if it were a native .NET object and making the .NET client appear to the COM object just as if it were a standard COM client.

You can create references to these COM object directly from Visual Studio for instance, and simply interact with them just as you would with a .NET class. This allows you to build your .NET application while talking to legacy COM objects and passing the results to MapXtreme .NET objects. Think of the .NET application you are developing as an orchestration piece of your application framework. In this way you can use the latest technologies available in the .NET framework and retain your existing investment in legacy libraries. Because all data from the COM objects has been translated to the .NET environment, passing these to other managed classes is allowed and causes no domain issues.

Sample Applications and Project Templates

MapXtreme integrates into Visual Studio and comes with various sample applications and project templates. The sample applications show different aspects of how to use our object model within the .NET framework and Windows Forms applications. The templates are basic starting points for application development.

The project templates that are part of Visual Studio allow you to create a starter project and demonstrate a generalized application framework. From Visual Studio, create a new project by selecting the Map Application template. This creates a basic single document interface (SDI) application. See the sample desktop applications for a basic multiple document interface (MDI) application and how best to integrate MapXtreme into this model.

The .NET Windows Forms development platform is basic in its approach and functionality. We recommend using third-party tools that supplies a more complete UI development platform.

The Desktop sample applications that ship with MapXtreme are designed to show you specific tasks with minimal overhead. Use these as learning tools instead of a starting point for production application as these were not designed as application development examples.

The remainder of this chapter describes the wide variety of controls, dialogs and tools available for building desktop applications.
The MapInfo.Windows.Controls namespace contains classes that implement controls in desktop applications. The use of Windows controls is quite simple as you drop the desired controls onto your form and use the various properties to configure them to your specifications. You will find that the controls in this namespace are similar to controls found in the MapInfo.WebControls namespace. However, the controls in each namespace cannot be substituted for each other. The controls in this namespace are specifically to be used in desktop applications. For information about controls for web-based applications, see Chapter 5: Web Applications, Controls, and Tools.

**Note** Some controls do not display at design time. When dragged and dropped onto a form they merely appear as a rectangle. These controls display properly at run time.

The desktop controls included with MapXtreme can be categorized into standard controls and map-specific controls. Standard controls include buttons, view tools, label tools, all kinds of boxes, etc. These are quite similar to (and in many cases, inherited from) classes that exist within the System.Windows.Forms namespace.

The map-specific controls are specific to Pitney Bowes MapInfo's mapping implementations. These include controls that set or modify aspects of object styles, labels, layers, coordinate systems, themes, and other related functions.

The most complex of the controls is the Layer Control, which is available as either a modal Dialog Box, or as a control that you can display directly on your form, as it is shown in the Workspace Manager. The Layer Control has an API that allows extensive customization; see the “LayerControlDemo” sample application for examples.

**Visual Studio Toolbox and Designer showing the list of available MapXtreme controls.**
Key Controls to Use in Desktop Applications

While all the controls included in MapXtreme are useful, several stand out from the rest as being central to most mapping applications. The following sections discuss these. Each of these controls assume the presence of the appropriate references included in your project. (These are automatically included if you have created your project from the MapXtreme mapping template in Visual Studio).

Note If you are not creating your project from one of our templates, please make sure to add references to the appropriate components (for example, MapInfo.CoreEngine, MapInfo.Windows, MapInfo.Windows.Dialogs, and MapInfo.Windows.Framework). See also MapXtreme Merge Modules in Chapter 2 on page 50.

MapControl

The MapControl is necessary for every application that displays a map. To add a MapControl to your form, drag it onto your form from the toolbox in Visual Studio. Once the MapControl is on your form there are several operations you can perform on it to make your map more useful to the user.

Modifications to the MapControl can be performed via the Properties Window when the MapControl is selected on your form. Right-click on the MapControl in design-mode to display a context menu with the following choices: Load Map, Clear Map, Layer Control, and Create Thematic. Load Map opens a standard file chooser from which the developer can choose a map to preload into the MapControl. Clear Map clears any map already in the control, either when adding the MapControl to the form or somewhere in the middle of the design process. Layer Control launches a Layer Control dialog box that allows you to customize the map (see Layer Control on page 130). Create Thematic launches the CreateThemeWizard (see CreateThemeWizard on page 133) allowing you to create a theme on the map.

A Form showing a MapControl, a Layer Control, and a MapToolBar
Key Controls to Use in Desktop Applications

The MapControl also has several properties relating to the loaded map that can be preset so that the map shown at run time is further customized for the specific application. Properties that can be modified include zoom level, coordinate system, map center, and settings for three mouse buttons during run time. The mouse buttons can be set to Zoom, Pan, Draw geometries, Select, or any of several other map tools. See Overview of the MapInfo.Tools Namespace on page 137.

You can also set a design time tool to further manipulate the map while working in Visual Studio. Design time functionality includes Zoom In, Zoom Out, Select, Pan, Center, and the default arrow. Clear Map from the context menu removes all layers from the MapControl’s map and closes the corresponding tables (if they are not in use in another MapControl).

The MapToolBar

The MapToolBar combines several of the map tool controls (e.g., zoom in, open table) into a single control.

If you create a map application using the templates that comes with MapXtreme, the application will automatically contain a MapToolBar. Alternatively, you can add a MapToolBar to your form by dragging it from the Visual Studio toolbox. By default, the MapToolBar contains the following buttons: OpenTable, LayerControl, Select, ZoomIn, ZoomOut, Pan, and Center.

You can also add buttons to the MapToolBar using the following steps:

1. Open the ToolBarButton Collection Editor by clicking on the ellipsis (...) next to the Buttons property in the Properties window.

   ![ToolBarButton Collection Editor](image)

   The ToolBarButton Collection Editor appears.
2. Click the **Add** button.
   This creates a new MapToolBarButton below the last one in the list.

   **Note** If you would like to add a customized tool you need to click on the arrow next to the Add button and choose ToolBarButton. You need to write a customized handler for the new button.

3. Name the new button.

4. Select the **ButtonType** from the right-hand pull-down menu.
   For example, if you are adding a Select tool button, choose Select from the list.
5. Click **OK**.
   The MapToolBarButton is now created and added to the button bar. The newly added tool works in the default manner.

**Adding Custom Buttons to a ToolBar**

This procedure assumes you have created a custom tool and added it to the MapControl.Tools collection. Adding custom buttons involves assigning the button's ToolId to the name of the custom tool and adding a custom bitmap to the MapToolBar's ImageList.

For information on how to create a custom desktop tool, see **Customizing Tools on page 144**.

For information on custom bitmap symbols that ship with MapXtreme see **Custom Symbols on page 508**.

To add a custom button to a toolbar:

1. Add a MapToolBar to your form.
2. In the Visual Studio property window, highlight the Buttons property and press the ... button to invoke the ToolBarButton Collection Editor dialog.
3. Click the **Add** button to add a new MapToolBarButton.
4. Set the button's ButtonType property to CustomTool (last item the drop-down list). Note that the button will now appear blank, as there is no image associated with the custom button yet.
5. Set the button's ToolId property to the name of the custom tool you added to the MapControl's Tools collection. If the ToolId value does not match a tool in the collection, then a runtime exception will be thrown when the user clicks the button.
6. Click **OK** to close the ToolBarButton Collection Editor dialog.
7. In the form's constructor, after the InitializeComponent call, add code to add a custom bitmap to the MapToolBar's ImageList. How the custom bitmap is associated with the application is up to the programmer. One option is to add an ImageList to the form at design-time, populate it with the custom images, then write code to transfer its images to the toolbar's ImageList at runtime. That code would look like this:

```csharp
// Add custom tool button's bitmap to toolbar's image list
foreach (Image image in this.imageList1.Images) {
    this.mapToolBar1.ImageList.Images.Add(image);
}

// Associate the bitmap with the custom tool's button (last image in the list)
this.mapToolBarButtonBlueSelect.ImageIndex = this.mapToolBar1.ImageList.Images.Count-1;
```

**Layer Control**

The Layer Control dialog box shows all the layers that make up the current map and the status of the layers’ attributes. These attributes are: visible, editable, selectable, and auto label. The icons above each check box column represent the attributes types. ToolTips display over the attribute icons when you move your cursor over them to help familiarize yourself with each icon. It is easy to change a layer’s, or multiple layers’, attributes using the check boxes.
The Layer Control also has options available to change the Display and Label settings; modify any thematic maps you have created, and reorder, add, or remove layers.

You can rearrange layers in the LayerControl by dragging and dropping them in the Layer Control Layers list.

**Note** Dropping a layer onto a Label Layer adds a new set of labels to the Label Layer. This can happen by accident—for example, if you were attempting to move a layer to the spot just above the Label Layer.

**Tip:** If you want the dropped layer to be placed outside of the Label Layer, hold down the Shift key before you drop the layer. As you press and release the Shift key, the cursor changes to indicate whether the dropped layer will go above or inside the target layer onto which you are dropping.

The Layer Control puts all the functionality of the Layer Control dialog box onto a form. A single line of code is necessary in the Form_Load() method to link the Layer Control to the MapControl’s map.

```csharp
layerControl1.Map = mapControl1.Map;
```

To better understand the complexities and capabilities of the Layer Control, run the MapXtreme utility called Workspace Manager. This tool includes a working example of Layer Control. See [Chapter 23: Workspace Manager](#). Workspace Manager is available from the Program Menu under MapInfo\MapXtreme\6.x.x.

### Customizing Context Menus

You can create customized context menu items for your Layer Control that appear when the user right-clicks a node in the layer tree. Use the ContextMenuTargetObject property to return the object that the user right-clicked.

A code sample has been provided that shows how to define a LayerControlEnhancer class that allows a user to add custom items to the Layer Control's context menu. This example can be found in the LayerControl sample application located in the ..\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features\LayerControl directory.

### Keyboard Shortcut Programmability

You can create keyboard shortcuts to access everything on the LayerControl toolbar. The PerformDown, PerformUp, and PerformRemove methods already provide programmatic access to the Down, Up, and Remove buttons.

The AddMenuMnemonic property provides programmatic access to the key associated with displaying the Add menu. The syntax for the AddMenuMnemonic property is:

```csharp
public System.Windows.Forms.Keys AddMenuMnemonic {get; set;}
```
The MapInfo.Windows.Dialogs namespace contains classes that create dialog boxes with specific functions. Contrary to controls, dialog boxes are not visible during design-time and can only be created and configured in code. You can create your own customized dialog boxes using classes from the MapInfo.Windows.Dialogs namespace and then utilize them by calling the System.Windows.Forms.Form.ShowDialog() method.

The stock dialogs are built from the MapXtreme object model's public API. There are no hidden internal fields or anything private. You can use these dialogs or customize them for your needs, or write your own from scratch. You may wish to provide fewer controls on a stock dialog so that your users are restricted from changing some behavior.

To use a particular dialog box in your application, add the MapInfo.Windows.Dialogs namespace to your code as follows:

```csharp
using MapInfo.Windows.Dialogs

private void DoLayerControl()
{
    LayerControlDlg layerControl = new LayerControlDlg();
    layerControl.Map = mapControl1.Map;
    layerControl.ShowDialog(this);
}
```

The code above displays the LayerControl dialog box when the DoLayerControl() method is called.

**Stock Dialogs**

MapXtreme ships with many stock dialogs and controls to handle tasks of manipulating the MapXtreme objects. All the dialogs and controls used in Workspace Manager are available for use in your application. The dialogs are simply containers for the low level controls. These controls are designed to work specifically with many of the MapXtreme objects.

The controls that ship with MapXtreme are used by the stock dialogs to create the basic UI components. The controls are placed in dialogs to create specific UI components. You can, in the same way, use the base controls to design your own dialogs. Simply create a dialog class and start dragging the controls onto the surface to create your dialog.

The stock dialogs also can be customized through visual inheritance. You can derive a class from our dialogs and customize some of the behaviors by overriding methods and properties. Other dialogs such as the Layer Control are customizable, so you can change the default behavior to hide controls you don't want users to access, change the look of the icons, or remove tabs.

Basically you have control over all the UI components that ship with MapXtreme. This allows you to create your own custom interface. Using any of the methods described allows you to create a customized desktop application that exposes only what you need rather than everything MapXtreme defines.
CreateThemeWizard

The CreateThemeWizard class is a convenience class that you can add to your application. This class is used to guide the user through the process of creating a new theme by launching a wizard (a set of dialog boxes). There are three basic steps to creating a theme: 1) Select the theme type; 2) Select the table and columns to use for the theme; and 3) Modify the theme attributes (style, number of ranges, etc.). The wizard ties all these settings together into a sequence of dialog boxes to make theme creation as simple as possible for the end users of MapXtreme applications.

Using the CreateThemeWizard

The CreateThemeWizard displays several different dialog boxes, depending on the choices made by the user. The first dialog box allows you to select a theme type.

Create Theme: Step 1 of 3 dialog box

If the user chooses a single variable theme type (Ranged, Dot Density, Individual Value, or Graduated Symbol) the second dialog box displayed allows them to choose a table and single column.

Create Theme: Step 2 of 3 dialog box (single column)

If the user chooses a multiple variable theme type (Pie or Bar), the second dialog box displayed allows them to choose multiple columns of data from which to create the theme.
Create Theme: Step 2 of 3 dialog box (multiple columns)

The following dialog box is specific to the type of theme chosen in Step 1. The figure shows a Step 3 dialog box specific to a ranged theme.

Create Theme: Step 3 of 3 dialog box (Ranged Theme)

The dialog box shows the default theme distribution method: EqualRangeSize. The corresponding name of the distribution method in the user interface of the wizard is Equal Ranges.
When a user makes a change in the Settings tab, such as changing the number of ranges, the Recalculate Theme button is activated. The user must click the **Recalculate Theme**, **Apply**, **OK** button, or on a new tab to perform the recalculation. The recalculation does not occur automatically. The figure below shows a Step 3 dialog box with a change to the number of ranges. The **Recalculate Theme** button is now available.

![Create Theme: Step 3 of 3 dialog box (Ranged Theme) After Settings Change](image)

Programmatic checks look for changes in the values of the Settings tab and if the theme bins need to be recomputed after:

- Tabbing to the Styles or Legend tab
- Clicking the Apply button
- Clicking the OK (accept) button

The CreateThemeWizard is accessible from many different places at design time and at runtime, depending on the controls in your application. At design-time the CreateThemeWizard can be accessed from the menu at the bottom of the MapControl properties window and also from the Layer Control by right-clicking on a layer and choosing **Add Theme**.

**Developing with the CreateThemeWizard**

To add a CreateThemeWizard programmatically, follow the example code.

```csharp
using MapInfo.Windows.Dialogs;
```
CreateThemeWizard themeWizard = new CreateThemeWizard(mapcontrol1.Map, this);

**Methods**

Once the wizard is created, call the CreateTheme() method from the CreateThematicWizard class to create the theme. An optional string parameter can be added to set the theme’s alias. The alias can be used to access the theme from the Layers collection in the case of an ObjectTheme (Bar, Pie, GraduatedSymbol) or the Modifiers collection in the case of a FeatureStyleModifier (Ranged, RangedLabel, IndividualValue, IndividualValueLabel, Dot Density). You would access the theme to modify or delete it.

ITheme theme = themeWizard.CreateTheme("theme1");

If you already know which table is to be themed or which label source to use for a label theme, you can use the methods, CreateFeatureTheme() for tables or CreateLabelTheme() for labels.

**Properties**

There are a few properties that can be useful for obtaining information about the theme creation. WizardResult contains an enumeration WizardStepResult that allows you to check how the user exited the wizard. These choices are WizardStepResult.Done and WizardStepResult.Cancel. These are useful to appropriately update the controls or menus in your application.

The following example shows the use of this.

```csharp
if (createThemeWizard.WizardResult == WizardStepResult.Done
{
    // Update the menus
    mnuRemoveTheme.Enabled = true;
    mnuModifyTheme.Enabled = true;
}
```

SelectedLabelSource, SelectedLayer, and SelectedThemeType are properties on the CreateThemeWizard class that can be used to find out which layer or label source the theme has been applied to. This is useful when you want to provide the ability to modify a theme and need to access that object.

To modify a theme, determine the type of theme and then launch the appropriate type of modification dialog. Dialog box classes for modifying themes are ModifyBarThemeDlg, ModifyDotDensityThemeDlg, ModifyGradSymbolThemeDlg, ModifyIndValueThemeDlg, ModifyPieThemeDlg, and ModifyRangedThemeDlg.

MapXtreme ships with a ThemeDialogs sample application in which you can analyze the implementation and customize it for your needs. See ..\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features\ThemeDialogs under your installation of MapXtreme.
Customizing Controls and Dialog Boxes

MapXtreme ships with a wide variety of dialog box classes in the MapInfo.Windows.Dialogs namespace. Each of the dialog boxes represented there have possible customizations specific to each one. To modify a specific dialog box, assign values to the different properties that are specific to each dialog box. Since there are so many, it is not practical to list all the possible customizations here. Please refer to the online reference for details about each dialog box.

An example of customizing a dialog box is as follows: The LineStyleDlg class creates a Line Style Dialog Box. This dialog box can be created in either a sparse style mode (where nothing is selected when opened), or in a mode where some choices are visible but not enabled (grayed out).

The customization provided by setting property values is obviously limited to the specific dialog object being modified. To create a new dialog box based upon a design of an established one is done by adding some controls to a new form at design time. For example, to create a modified LineStyle Dialog Box, you can drop a LineStyle Control on a new form and then add other controls to that form, making, in effect, a modified LineStyle Dialog Box.

The MapInfo Knowledgebase provides examples of customizing desktop controls. For example, to create a custom LayerControl, see the topic under MapXtreme 2004/2005 > Selections.

Overview of the MapInfo.Tools Namespace

MapTool Object Model

The MapTools are the objects you use to interact with the map. These tools implement the basic behaviors you expect of a map. There are view-based tools, zooming and panning, tools for creating geometries on a specific layers, and tools for generating selections based on an area of the map. Each tool is customizable using events to capture pre- and post-process events.

Properties of the methods associated with the tools are on the tool and not the layer. For example, the selectability of a layer is specified on the tool and not the layer. In this way, you can use one tool to select an object on one layer and use another tool to select from a second layer. This gives much more freedom in the tools but also enables you to emulate the properties on layers.

This design also allows you to create tools that insert into specific layers. For example, a city planner may want the application to insert manholes on the water layer only, but to insert trees on the vegetation layer. You could create a custom tool that when selected, inserts only the specified symbol or geometry on the appropriate layer. This also works for the stock tools, so that drawing a polygon will always be on the specified layer. Again the stock Layer Control manipulates the tools collection to make it seem as if the insertion is on a single layer.

The MapInfo.Tools namespace contains all the classes that allow you to create basic and customizable tools for your MapXtreme application.

You can customize new instances of these stock tools with event code, or write your own tool classes that derive from stock tools but override specific methods. See Customizing Tools on page 144.
Overview of the MapInfo.Tools Namespace


For example, the code below sets the left mouse button to the ZoomIn tool:

```csharp
mapControl1.Tools.LeftButtonTool = "ZoomIn"
```

MapXtreme provides ready-to-use tools in its Windows Forms templates and in the Visual Studio designer toolbox. Simply drag-and-drop a tool onto your form and set properties based on your needs.

For an example of using the desktop tools programmatically, see the sample application under ..\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features\DesktopTools under your installation of MapXtreme.
MapXtreme Desktop Tools API

The diagram below shows a UML representation of the MapTool class.

Tools are ultimately derived from the MapTool class. Select and Add tools are derived from the SelectMapTool and AddMapTool classes, respectively.

The Tools included in this namespace can be divided into four categories: View tools, Select tools, Add tools and Custom tools.

Within the Add, Custom and Select tool categories are tools for drawing or selecting these geometric objects: Ellipse, Arc, Circle, Rectangle, Polygon, Point, Line, and Polyline.

All of the tools have properties associated with them which implement three interfaces: IMapToolProperties, ISelectMapToolProperties, and IAddMapToolProperties.
Overview of the MapInfo.Tools Namespace

View Tools

The View tools (ZoomMapTool, CenterMapTool, and PanMapTool) change the view of the map accordingly to the tool being used. These tools are part of the default tool collection. If you begin building your desktop applications with a MapXtreme template, these tools will appear on the default toolbar.

The LabelMapTool allows you to label features on the map. This tool is not included in the default tool collection. To add it click on the Collections ... button under the Buttons property of the MapToolBar.

Select Tools

The Select family of tools (SelectMapTool, SelectPointMapTool, SelectPolygonMapTool, SelectRadiusMapTool, SelectRectMapTool, and SelectRegionMapTool) select points and regions that fall within the geometric object shaped by the use of these tools. For example, the SelectRectMapTool selects objects that fall within the rectangle. SelectMapTool is the abstract base class that the other select tools derive from.

Note  MapXtreme does not support the intersection of an object with a polyline, hence there is no select polyline tool.

The SelectPointMapTool can be used to rotate objects.

A Select Node mode allows individual points to be moved, added, or removed from objects. See Editing a FeatureGeometry with the Select Tool on page 146 for an example.

Dynamic Selection shows what objects can be in the selection as the mouse is being moved when using the SelectRectMapTool, SelectRadiusMapTool, and the SelectPolygonMapTool.
The selection map tools have properties to set various options, including which layers are selectable. These default tool properties can be set on the MapTools collection which can be found in MapInfo.Windows.Controls.MapControl.Tools. Optionally, each tool can set its own overriding properties and specify to use the default value on the MapTools collection or its overriding value.

To set the default list of selectable layers, create an IMapLayerFilter which specifies which layers are selectable. Then set the default property on the MapTools for the SelectableLayerFilter.

For example, to set the default list of selectable layers to work on all vector layers:

```csharp
// select from all vector layers
IMapLayerFilter selectableLayerFilter = MapLayerFilterFactory.FilterAnd(MapLayerFilterFactory.FilterVisibleLayers(true), MapLayerFilterFactory.FilterByLayerType(LayerType.Normal));

mapControl1.Tools.SelectMapToolProperties>SelectableLayerFilter = selectableLayerFilter;
```

For more information on Select tool properties see MapInfo.Tools.MapTool.ISelectMapToolProperties in the MapXtreme Developer Reference Help.

### Add Tools

The Add family of tools (AddArcMapTool, AddCircleMapTool, AddEllipseMapTool, AddLineMapTool, AddPointMapTool, AddPolygonMapTool, AddPolylineMapTool, and AddRectangleMapTool) are all based on the AddMapTool class. The Add tools allows you to draw a particular geometric object on your map. In order to add an object to a map, you must have an active insertion layer in your map where the object resides after it is created. The size and shape of the particular object depends on the tool used and can be constrained through the use of the modifier keys (Shift and Ctrl) while the tool is being used.

For more information on Add tool properties see MapInfo.Tools.MapTool.IAddMapToolProperties in the MapXtreme Developer Reference Help.

### Custom Tools

The Custom tools (CustomArcMapTool, CustomCircleMapTool, CustomEllipseMapTool, CustomLineMapTool, CustomPointMapTool, CustomPolygonMapTool, CustomPolylineMapTool, and CustomRectangleMapTool) are very basic tools that only fire events. Use these classes to design custom behaviors for specific tools. For example, you can use the CustomEllipseMapTool to draw green Ellipses with a red outline every time the tools is used. Use the various tool events (see Tool Events on page 145) to specify particular behaviors to respond to specific events generated by the tools.

Custom tools are not included in the default tools collection. If you wish to assign a custom tool to a mouse button, you may provide any string you wish to identify the tool.

For more on customizing tools, see Customizing Tools on page 144.
Overview of the MapInfo.Tools Namespace

Shape Tools

MapXtreme provides a group of tools for drawing geometric features on your map. Most of these tools are used by “click-and-drag.” As you click and then drag to another location on your map, a rubber-band image is displayed showing the current size and shape of the object being drawn. Each of these tools uses the Esc key to cancel the current operation (if appropriate).

The following are types of shape tools MapXtreme supports.

Line Tools

The tools included in this group are AddLineMapTool and CustomLineMapTool. These tools draw a Line. The tool is activated by clicking and dragging from the beginning point to the end point. Releasing the mouse button creates the Line. If the Shift key is pressed while dragging, the angle of the Line is constrained to multiples of 45°. If the Ctrl key is pressed while dragging, the line doubles in length and height. Pressing Esc before releasing the mouse cancels the operation.

Polyline Tools

The tools included in this group are AddPolylineMapTool and CustomPolylineMapTool. These tools draw a Line with multiple segments. The tool is activated by clicking on a point and then clicking on a subsequent point. When you are done clicking on points, click again on the last point and the Polyline is drawn. If the Shift key is pressed while dragging, the angle of the Line is constrained to multiples of 45°. The Ctrl key has no effect on this tool. Pressing Esc before completing the line, cancels the operation.

Ellipse and Arc Tools

The tools included in this group are: AddEllipseMapTool, CustomEllipseMapTool and CustomArcMapTool. The Ellipse tools draw an Ellipse. The CustomArcMapTool draws an arc only; it does not insert the arc into a layer.

The Ellipse tool is activated by clicking and dragging from a point on one side of the perimeter to a point on the other. Releasing the mouse button creates the Ellipse. If the Shift key is pressed while dragging, the Ellipse is constrained to a circle where the radius is constant. If the Ctrl key is pressed while dragging, the Ellipse is drawn with the starting point in the center. Pressing the Esc key before releasing the mouse cancels the operation. Note that the axis of the Ellipse is always aligned to the coordinate system.

MapXtreme provides a lightweight CustomArcMapTool that draws a reference arc on the map, but which does not provide or insert a LegacyArc FeatureGeometry into the layer. To create an arc tool with such behavior, you must create your own implementation. For example, if you want a CustomArcMapTool to have the same operations as an AddMapTool, create a new arc tool class that derives from CustomArcMapTool and implements the IAddMapToolProperties interface.

To view C# and VB sample code that illustrates this, go to MapXtreme 2004/2005 > Tools in the Pitney Bowes MapInfo Knowledgebase. The Knowledgebase is accessible from the MapXtreme Learning Resources browser on your Program menu or from the MapInfo website under Support & Training.
Circle Tools

The tools included in this group are AddCircleMapTool and CustomCircleMapTool. The Circle tools draw a Circle. This is equivalent to using the Ellipse tool while holding down the Shift key. The tool is activated by clicking and dragging from the center point of the Circle center to a point on the perimeter. As you drag, you will see the size of the Circle expand around that center point. Releasing the mouse button creates the Circle. If the Ctrl key is pressed while dragging, the Circle is drawn from perimeter point to perimeter point. Pressing the Esc key before releasing the mouse cancels the operation.

Rectangle Tools

The tools included in this group are AddRectangleMapTool and CustomRectangleMapTool. These tools draw a Rectangle. The tool is activated by clicking and dragging from one corner to the opposite corner. Releasing the mouse button creates the Rectangle. Note that the drawn object is always axis-aligned. If the Shift key is pressed while dragging, the Rectangle is constrained to a square. If the Ctrl key is pressed while dragging, the Rectangle is drawn with the starting point in the center. Pressing the Esc key before releasing the mouse cancels the operation.

Using InfoTips

An InfoTip is a small text box that can appear when the mouse hovers over a map feature. A map is associated with an instance of MapInfo.Tools.MapTools, which contains all the tools (instances of MapInfo.Tools.MapTool) that can be used on the map.

The MapInfo.Tools.MapTools class has three properties that control the appearance of InfoTips. They are:

- InfoTipsEnabled—Gets/Sets whether InfoTips will appear when the mouse is idle.
- InfoTipTimerDelay—Gets/Sets the amount of time (in milliseconds) that must elapse before the InfoTip appears. The default is 500.
- InfoTipDisplayDelay—Gets/Sets the amount of time (in milliseconds) that the InfoTip will be displayed. The default is 0, which means that the InfoTip will be displayed until the mouse is moved.

Note that as long as the mouse is moving, InfoTips will not appear. An InfoTip will only appear when the mouse is over a feature with a label and has been idle for at least the interval specified by InfoTipTimerDelay. The InfoTip will disappear when the mouse is moved or when the mouse has been idle for longer than the time specified by the InfoTipDisplayDelay provided that time is greater than zero.

You can prevent InfoTips from being displayed at all with:

```csharp
mapControl1.Tools.InfoTipsEnabled = false;
```

You can set the delay before an InfoTip is displayed to one second with:

```csharp
mapControl1.Tools.InfoTipTimerDelay = 1000;
```

You can set the length of time an InfoTip is displayed even though the mouse remains idle to 2.5 seconds with:

```csharp
mapControl1.Tools.InfoTipDisplayDelay = 2500;
```
See the MapInfo.Tools.MapTools class in the MapXtreme Developer Reference for more information on using InfoTips in the API.

Customizing Tools

Tools for desktop applications can be customized in two ways: by creating a subclass of an existing tool, or by using one of the tools in the Custom Tools group. Any of the Tool classes provided in MapXtreme can be customized by creating a subclass derived from the desired particular Tool class. An example of this is deriving a new tool class from the AddLineTool class. The tool could be changed to a particular behavior (e.g., always beep when line is drawn), a particular appearance (e.g., always draw the line in red), or to a particular functionality (e.g., always constrain line drawing to 90 degree increments).

Another example would be to create two different tools based upon the same tool. The AddPointTool could be used to subclass two other tools that each use different symbols for indicating two kinds of points on a map. Assign one tool to the left mouse button and assign the other to the right mouse button.

The following code example illustrates a customization of the AddPolygonTool.

**VB example:**

```vbnet
Dim insertionlayerfilter As IMapLayerFilter
Dim style As MapInfo.Styles.CompositeStyle
Dim addmaptoolproperties As MapInfo.Tools.AddMapToolProperties
Dim maptool As MapInfo.Tools.MapTool

insertionlayerfilter = _
    MapLayerFilterFactory.FilterByLayerType(LayerType.Normal)
style = New MapInfo.Styles.CompositeStyle
addmaptoolproperties = New _
        MapLayerFilterFactory.FilterVisibleLayers(True), _
        "CustomPolygonAddMapToolProperties", Nothing), style)

maptool = New MapInfo.Tools.AddPointMapTool(MapControll.Viewer, _
    MapControll.Handle.ToInt32(), MapControll.Tools, New _
    MapInfo.Tools.MouseToolProperties(Cursors.Default, _
        Cursors.Default Cursors.Default), _
    MapControll.Tools.MapToolProperties, addmaptoolproperties)
```

The Pitney Bowes MapInfo KnowledgeBase has additional code examples for customizing desktop tools. For an example of how to create a desktop InfoTool, go to MapXtreme 2004/2005 > Tools.

Additionally, MapXtreme ships with a desktop tools sample application in which you can analyze the implementation and customize it for your needs. See ..\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features\DesktopTools\cs under your installation of MapXtreme.
### Tool Events

The MapInfo.Tools namespace supports tool events with information and the ability to cancel the tool's operation. A Select Tool's event lists the items being selected or deselected. Events can be fired at every stage of use of a particular tool. In your code you can trap certain moments in the lifecycle of the usage of a tool. Adding code to particular events gives you the maximum flexibility on customizing the use of each tool in your mapping application.

The events in the namespace are as follows:

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureAddingEventArgs</td>
<td>This event is fired when an add tool is about to draw an object.</td>
</tr>
<tr>
<td>FeatureAddedEventArgs</td>
<td>This event is fired when an add tool has added an object to a table and map.</td>
</tr>
<tr>
<td>FeatureSelectingEventArgs</td>
<td>This event is fired when a selection tool is about to change the selection.</td>
</tr>
<tr>
<td>FeatureSelectedEventArgs</td>
<td>This event is fired when a selection tool changes the selection.</td>
</tr>
<tr>
<td>FeatureChangingEventArgs</td>
<td>This event is fired when a selection tool is about to change a feature. Use this event to check to see if the change that is going to happen is valid.</td>
</tr>
<tr>
<td>FeatureChangedEventArgs</td>
<td>This event is fired after a selection tool changed a feature.</td>
</tr>
<tr>
<td>NodeChangedEventArgs</td>
<td>This event is fired when a selection tool alters the nodes in a selection.</td>
</tr>
<tr>
<td>NodeChangingEventArgs</td>
<td>Fired when a selection tool is about to change a node of a selected feature. Use this event to check to see if the change that is going to happen is valid.</td>
</tr>
<tr>
<td>ToolActivatedEventArgs</td>
<td>This event is fired when a mouse tool is activated.</td>
</tr>
<tr>
<td>ToolActivatingEventArgs</td>
<td>This event is fired when mouse tool is about to be activated.</td>
</tr>
<tr>
<td>ToolEndingEventArgs</td>
<td>This event is fired when a mouse tool is about to end. This is a good place to have a a new action begin.</td>
</tr>
<tr>
<td>ToolUsedEventArgs</td>
<td>This event is fired when a mouse tool is in use. Use this event to set flags for the beginning, middle, and end of a the sequence of mouse clicks.</td>
</tr>
</tbody>
</table>
Editing a FeatureGeometry with the Select Tool

The following section shows you how to edit features by reshaping and adding nodes to a FeatureGeometry. It illustrates the use of a MapControl, LayerControl and the Select tool. See also MapControl on page 127 and Layer Control on page 130.

Reshaping a Feature

1. Load a map into the MapControl and zoom in on the Feature you wish to modify.

2. Select the object's layer in the LayerControl.

3. On the Visibility tab of the LayerControl, select the Show Nodes check box.

4. On the Options tab, select the Selectable and Editable check boxes.

5. Select the root node of the map in the layer control tree view.
6. On the Editing tab, select Allow Node Editing from the Editing mode combo box.

7. Click OK to accept your changes. The map display will change and display the object's nodes.

8. Using the Select Item tool, select the polygon you want to modify, then click and drag on a node to change its position. Use the Shift key to select a range of nodes and the Ctrl key to toggle the selected state of a node.

Adding Nodes

To add nodes to a Feature Geometry:
1. Open the LayerControl again and select the root node of the map.
2. On the Editing tab, select Allow node adding from the Editing Mode combo box.

3. Click OK to accept your change.
4. On the map, select the polygon you want to modify.
5. Using the Select Item tool, hold down the Ctrl key and click on the edge of the polygon where you want to add the node. The new node will appear.

Reshaping and Adding Nodes Programmatically

This section describes how to reshape and add nodes programmatically. Code is provided in C# and VB.

1. Add a button named "btnEditNodeTool" to the main form in the Visual Studio designer.
2. Double-click the button to open the button's handler in the code page. Add the appropriate code sample:

   VB example:
   
   Private Sub btnEditNodeTool_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
   MapControl1.Tools.LeftButtonTool = "Select"
   End Sub

   C# example:
   
   private void btnEditNodeTool_Click(object sender, System.EventArgs e)
   {  
   mapControl1.Tools.LeftButtonTool = "Select";
3. Run the application.
4. Select the object's layer in the LayerControl.
5. On the Visibility tab of the LayerControl, select the Show Nodes check box.

6. On the Options tab, select the Selectable and Editable check boxes.

7. Click **OK** to accept the change.
8. Now click the new button you added to the form. The cursor will change to the Select arrow.
9. Select an object to modify, then click and drag on a node (as in step 8 of the previous example).
Working with Core MapXtreme Classes

The MapInfo.Engine namespace contains the interfaces and classes that relate directly to the core functionality that drives all applications based on MapXtreme. This includes the core ISession interface which is the starting point for all MapXtreme applications. Classes in this namespace include Session and Selections, and SearchPath. Other types in the namespace are supporting classes, delegates, structures, and enumerations for Collections, Resources, and CustomProperties.

In this chapter:

- ISession Interface ........................................... 152
- Serialization and Persistence .............................. 154
- Selection Class .............................................. 156
- Selection Code Examples ................................ 157
- Event Arguments ......................................... 159
- Exceptions .................................................. 159
The ISession interface is the starting point for all MapXtreme-based applications. It manages the initialization of resources needed for a MapXtreme application.

An instance of ISession holds components of the MapXtreme object model such as the DataAccess engine, MapFactory, CoordSysFactory so that the desktop or web application can do work. The following diagram illustrates the classes that implement ISession interface.

For an ASP.NET application each client request has its own ISession instance. This instance resides in the calling context and is available throughout the lifetime of the client's request.

For a single-threaded desktop application there is only one instance. On a multi-threaded desktop application there is one instance per thread.

The MapInfo.Engine.Session class provides access to the ISession object. To get the current ISession instance, access the MapInfo.Engine.Session.Current property.

Session Management

Session management is a key topic to understand when you are designing your application. While a desktop application's session management is straightforward (each user has his own ISession instance), a web application needs to factor in, perhaps, an unknown number of users who will use your application. It will have to known how to handle each user's state so that the correct information and visual display is returned to the correct user.

MapXtreme provides templates for building web applications that help you manage state properly. This topic is discussed in greater detail in Chapter 6: Understanding State Management. Key decisions about state management, pooling, performance and data access are presented to help you make informed decisions during the design phase of your project, before you start coding.
Using Session.Dispose Method

The MapInfo.Engine.Session class has two overloaded Dispose methods. Your choice will depend on the type of application you are building.

Session.Dispose()

Session.Dispose() disposes the ISession instance that is accessible through the Session.Current property. This method is used only for multi-threaded desktop applications.

Do not use this for web applications or single-threaded desktop applications. For web applications, ISession is managed by WebSessionActivator.

For single-threaded desktop application, Dispose is called automatically when the application is shutdown or when the AppDomain using MapXtreme is unloaded.

Session.Dispose(HttpSessionState)

Use Session.Dispose(HttpSessionState) for web applications that use the default session state settings in which the ISession is stored in memory. Do not call this method for any other configuration since the ISession instance is not stored in memory in any other configuration.

The state settings are represented in the Web.config file of your application project by the following keys:

    <add key="MapInfo.Engine.Session.State" value="HttpSessionState" />
    <sessionState mode="InProc" />

The first setting is an application-specific setting that controls the mechanism for saving and restoring the state of the MapInfo.Engine.ISession instance. This instance is accessible through the MapInfo.Engine.Session.Current property. The HttpSessionState setting indicates that the session is saved and restored through the ASP.NET session state. This state is exposed through the current HttpContext and is of type HttpSessionState.

The second setting is an ASP.NET setting that controls how the HttpSessionState is saved and restored. InProc indicates that the state of the ASP.NET session is to be placed in memory and is unique to each ASP.NET ISession instance. This is the default setting.

When you use these settings, there is an ISession instance for each ASP.NET session and it is stored in the HttpSessionState throughout the lifetime of the ASP.NET session. In order for the ISession instance to be properly disposed of when the session times-out or ends, you must add the following statement to the Session_End method in your Global.asax source code file.

C# example:

    MapInfo.Engine.Session.Dispose(this.Session);

VB example:

    MapInfo.Engine.Session.Dispose(Me.Session)

Making this call will ensure that the ISession instance is properly disposed of and that memory is reclaimed.
Serialization and Persistence

A Map is bound to a ISession object. You cannot take a Map object from one ISession object and use it in another. You have two options if you want to clone the entire ISession object; either by using Serialization or by using Persistence.

Serialization

Serialization is the process of converting an object into a stream of data in order to preserve it on the server. This process is an essential part of maintaining objects in MapXtreme web applications. If the objects are not maintained the server would need to recreate the object (such as a map) for each web request. When an object is requested, it is deserialized (or recreated from the stream of data) and then modified. This algorithm does not make a copy of the object (as other serialization algorithms do) such that the object being deserialized is created only once.

The serialization is performed by formatters that are embedded in the Microsoft.NET Framework. Two different formatters are included in the Framework, one for binary objects, BinaryFormatter, and one for SOAP objects, SOAPFormatter (SOAP is a lightweight protocol intended for exchanging structured information in a distributed environment, such as the web.). SOAPFormatter is relatively faster than BinaryFormatter. SOAPFormatter is used for certain basic types of data (Int, Byte, Decimal, etc.) and BinaryFormatter is called for complex of objects. See the Microsoft MSDN documentation for more information about the SOAPFormatter and the BinaryFormatter.

To pass an object to one of these formatters, use the GetObjectData() method. To deserialize the object (restore it from the stream) use the SetObjectData() method.

Any object that supports the ISerializable interface will automatically be restored, or deserialized. The ASP.NET framework automatically deserializes the context.Session[] array after HttpApplication.BeginRequest. MapInfo.Engine.Session is set up in HttpApplication.BeginRequest handler so objects are deserialized into the MapInfo.Engine.Session.

Serialize/Deserialize the Session Object

The following is an example of how to serialize/deserialize the Session object.

```csharp
// Create a MemoryStream to serialize into
MemoryStream stream = new MemoryStream();
// Serialize the MapXtreme Session object
BinaryFormatter formatter = new BinaryFormatter();
formatter.Serialize(stream, Session.Current);
stream.Position = 0;
// Make changes to the Session object to make sure the
// deserialization works correctly
...
// Recreate the MapXtreme Session object from the stream
// Note: this will replace the current MapXtreme Session object with the
// contents of the stream
formatter = new BinaryFormatter();
formatter.Deserialize(stream);
```

The stream parameter passed to the formatter.Serialize method can be anything derived from System.IO.Stream.
This serialization functionality was designed to be used with MapXtreme’s state management feature. If you persist this information to disk, then try to reload it in the future with a different version of the product, it’s not guaranteed to work.

For more on serialization and state management see Implementing a StateManager in Chapter 6 on page 110.

Persistence

Persistence in MapXtreme is the is the process of storing application objects to MapInfo Workspace (.MWS) files, which exist in XML format. Persisting and de-persisting ISession objects is done using the WorkSpacePersistence and WorkSpaceLoader classes. These two classes will write out and read in .MWS files. The application can write out the workspace (a copy of the Session), and apply the contents of the workspace file to a new Session object, creating a clone of the Session.

VB example:

```vbnet
Public Shared Sub MapInfo_Persistence_WorkSpacePersistenceSave()
    ' Create a named connection point to "D:\data\version2"
    Dim info As NamedConnectionInfo = New NamedConnectionInfo("file", _,
        ConnectionMethod.FilePath, "D:\data\version2")
        info)

    ' Create a map
        "MyMapAlias", New Size(400, 400))
    Dim table As Table = Session.Current.Catalog.OpenTable("MyDataFolder", _
        "myTableAlias", "Seamless\Lines - NYALBA\SeamCapDist.TAB")
    map.Layers.Add(New FeatureLayer(table))

    ' Save the Session to a workspace file
    Dim w As WorkSpacePersistence = New WorkSpacePersistence
    w.Save("c:\workspace\mySeamless.mws")
End Sub
```

Note that adding a workspace (using the WorkSpaceLoader class) is an cumulative process. To ensure the MapXtreme Session only contains the contents of the new workspace file, call Reload() method on the ISession object first. This method clears the Session state and reloads pre-defined workspaces in the application's Web.config file.

For more information on persistence and the XML schema, see Appendix D: Understanding the MapInfo Workspace.
Selection Class

A Selection is a collection of IResultSetFeatureCollection objects that holds lists of features. These features are a subset of rows in a table. They could be property boundaries, street networks, cell tower locations, or natural features such as rivers. They are typically drawn with special highlighting when they display in a Map. There can only be one IResultSetFeatureCollection for a given table in a Selection.

There can be more than one Selection in a ISession. The Selections collection contains all of the selections in the application. There is always at least one selection, known as the DefaultSelection.

Each Selection must have a name and unique alias. By default, map selection tools modify the Selection when used. Each tool can be set to use any particular Selection.

A selection in MapXtreme is not a copy; it is a reference to an IResultSetFeatureCollection for a given table in a Selection. If you attempt to modify a Selection after you have closed the table that you are working with, the reference to the IResultSetFeatureCollection will be invalid, causing an exception.

Features are selected using tools or through search methods. For a discussion of the different selection tools that you can use when building a Windows Forms application, see Chapter 8: Desktop Applications, Controls, Dialogs, and Tools. For selection Web controls and tools, see Chapter 5: Web Applications, Controls, and Tools.

Features can also be selected through search methods from the MapInfo.Data.Catalog class which returns IResultSetFeatureCollection collections. A Selection object can be passed into a search, which can be used to populate or change a Selection.

Features can also be selected via the ExecuteFeatureCollection method from the Data.MICommand class. In this case, you would execute SQL commands against the MapInfo Data Provider.

For more information on features, tables, the Catalog, and the MICommand, see Chapter 10: Working with Data.

Using Selection Properties

The properties of the Selection class are used to set the (required) name and alias, set a selection to be Visible or Editable, or to get the selection’s Style. To determine whether a Selection displays its highlighting to indicate it is selected or is available for editing, use the Visible or Editable property, respectively.

If the Editable property is set to true, the table that you are working with must also be editable.

The Style property indicates the Selection’s style and returns a reference to the Selection’s composite style. When you change a Style property, the Selection object is notified of the change. It will take affect the next time the Selection object is drawn. Styles are discussed in Chapter 16: Stylizing Your Maps.
Selection Highlighting and Exporting

Selections are typically drawn on the map with special highlighting, to distinguish these features from surrounding non-selected features. The highlighting is controlled by the `Mapping.FeatureViewer.DrawSelections` property. When this property is set to true, they are drawn with selection highlighting, provided they are drawn on a visible layer.

Similarly, the `MapExport.ExportSelection` property, can be used to control whether the selections are drawn into the exported image.

SelectionChangedEvent

A delegate method is attached to the `SelectionChangedEvent` in order to receive notification that this selection has changed. For example, if a record is added, the `SelectionChangedEvent` is fired.

ISerializable Interface on Selection and Selections Classes

The `I Serializable` interface is implemented on the Selection and Selections classes. The following code example demonstrates how to serialize or deserialize a Selections object:

```csharp
// Create a MemoryStream to serialize into
MemoryStream stream = new MemoryStream();

// Serialize the Selections object
BinaryFormatter formatter = new BinaryFormatter();
formatter.Serialize(stream, Session.Current.Selections);
stream.Position = 0;

// Make changes to the Session's Selections object to make sure the
// deserialization works correctly.
...

// Recreate the Selections object from the stream.
// Note: this replaces the current Session's Selections object with
// the contents of the stream
formatter = new BinaryFormatter();
formatter.Deserialize(stream);
```

Selection Code Examples

The following are code examples of common selection operations. Find this and other examples in the MapInfo Knowledgebase under `MapXtreme 2004 > Selections`. You can access the Knowledgebase from the MapXtreme Learning Resources browser on your Program menu or from the MapInfo website under Support & Training. Additional code examples are included in many topics of the MapXtreme `Developer Reference`. 
Selecting Features Within Another Feature

A common search technique using MapXtreme is to find features within another feature. You may do this to find all the customers within a postal code boundary or all the highways that are under construction in a sector. Follow the example below. The parameter \( f \) is a MapInfo.Data.Feature.

VB example:

```vbnet
Dim si As MapInfo.Data.SearchInfo = _
MapInfo.Data.SearchInfoFactory.SearchWithinFeature(f, _
    MapInfo.Data.ContainsType.Centroid)
Dim irfc As MapInfo.Data.IResultSetFeatureCollection = _

irfc.Close()
```

Checking a Table for Selections

Follow the code example below to learn how to get a count of selections in a table.

VB example:

```vbnet
Public Shared Sub MapInfo_Engine_Selection2()
    Dim session As ISession = MapInfo.Engine.Session.Current
    Dim tableUsa As Table = session.Catalog("usa")

    ' Get fc for selection on usa.
    Dim fc As IResultSetFeatureCollection = _
        session.Selections.DefaultSelection(tableUsa)
    Dim nCount As Integer = 0
    If Not fc Is Nothing Then
        nCount = fc.Count
    End If
End Sub
```

You can also perform selection operations using MapInfo SQL queries and with the ADO.NET data provider. See Chapter 10: Working with Data.

Returning All Columns From a Table

The following sample shows how to return all columns from a selection:

VB example:

```vbnet
Dim Connection As MIConnection = New MIConnection
    Connection.Open()
Dim lyr As FeatureLayer = MapControl1.Map.Layers("usa")
Dim ti As MapInfo.Data.Table = _
    MapInfo.Engine.Session.Current.Catalog.GetTable("usa")
Dim si As MapInfo.Data.SearchInfo = _
    MapInfo.Data.SearchInfoFactory.SearchAll()
si.QueryDefinition.SetColumns("*")
```
Dim irfc As MapInfo.Data.IResultSetFeatureCollection = _
Dim l As MapInfo.Data.Feature
For Each l In irfc
   Dim column As MapInfo.Data.Column
   For Each column In l.Columns
      MessageBox.Show(column.ToString())
   Next
Next

Changing the Map View Following a Selection

The following example shows how to change the zoom to display all the features in a selection.

VB example:

Me.MapControl1.Map.Bounds = _

Event Arguments

The MapInfo.Engine namespace contains various event argument classes that provide data for events. Refer to the online help for more information. Some of the event argument classes include:

- CollectionCancelableEventArgs – Provides data for a collection event that can be cancelled.
- CollectionEventArgs – Provides data for a collection event.
- NodeSelectionChangedEventArgs – Fires these event arguments when the node selection changes.
- SelectionChangedEventArgs – Other objects can attach delegates to this event to get notified when a selection changes.

Exceptions

The Engine namespace contains various exception classes. Refer to the online help for more information. Some of the exception classes include:

- ResourceNotFoundException – Throws this type of exception when the requested object is not found in the Resource table.
- ResourceTypeMismatchException – The exception that is thrown when the object read from Resources was not of the expected type.
- TimeoutException – The exception is thrown on Current timeout while waiting for a pooled ISession to become available.
The MapInfo.Data namespace contains the classes and interfaces that provide multiple ways of accessing data from a MapXtreme application.

In this chapter:

- Overview of MapInfo.Data Namespace .......................... 162
- Catalog and Tables .................................................. 163
- Supported Table Types ............................................ 167
- Working with Catalog and Tables ............................. 169
- Table Metadata (TableInfo) ...................................... 172
- MapInfo ADO.NET Data Provider ............................. 187
- Features and Feature Collections ............................. 191
- Searching for Features ........................................... 193
- Analyzing Data ..................................................... 198
- Improving Data Access Performance ....................... 200
The MapInfo.Data namespace contains the classes and interfaces that provide multiple ways of accessing data from a MapXtreme application. Within this namespace is the MapInfo ADO.NET data provider with a MapInfo SQL language for standard querying of databases and tables. The Feature object model is another way to access data that uses objects instead of SQL. The Catalog is the starting point for data access, containing methods for managing tables (open, close, create) and searching for data in a variety of ways.

This chapter is organized to follow the MapXtreme Data Access Model diagram above, and includes these topics:

- **Catalog and Tables**
- **Table Metadata (TableInfo)**
- **MapInfo ADO.NET Data Provider**
- **Features and Feature Collections**

Data access is central to any MapXtreme application, and covers a wide variety of topics. Within the topics listed above are other important topics of information that should not be overlooked.

Following this chapter are two additional chapters related to data access: **Chapter 11: Creating Expressions**, and **Chapter 12: Accessing Data from a DBMS**.
Catalog and Tables

Catalog is the manager of the MapXtreme data access model. Tables are a fundamental unit of MapXtreme. Tables hold the data that you want to display and analyze in your application. The Catalog, as manager, holds a list of tables that are currently open in the session. Tables are also opened, created and closed from the Catalog.

Nearly all of MapXtreme’s data access operations involve the Catalog and tables.

Tables

The Table class is the basic unit of all data access. Table, Column, and all TAB file metadata information is accessible from a MapInfo Table. Tables may be mappable (contain a column of type FeatureGeometry) or non-mappable. Tables also may be open and accessed without displaying a map.

Table Aliases

When tables are opened, they can be assigned a name (or alias) which is used while the table is open for referencing the table. For example, the table may be referred to by its alias in SQL statements. A table that is opened from a TAB file is assigned a default alias if no alias is specified. The default alias is based upon the name of the TAB file. This property is optional and may be set to null. However, it is good practice to assign an alias.

Columns

A Column object identifies the properties of a column in a table, feature, or feature collection and specifies the column’s name (alias), data type, width (for string and decimal columns), and other properties of the column.

Note  The MapInfo.Data.MIDataReader.GetDateTime method works with both DateTime and Date data types. In both cases, a System.DateTime value is returned. However, the MapInfo.Data.Column.DataType property will reflect the actual data type, either Date or DateTime.
Supported data types include:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int</td>
<td>Provides a 32-bit signed integer. This maps to the .NET Framework datatype Int32.</td>
</tr>
<tr>
<td>SmallInt</td>
<td>Provides a 16-bit signed integer. This maps to the .NET Framework datatype Int16.</td>
</tr>
<tr>
<td>Double</td>
<td>A floating point number within the range of -1.79E +308 through 1.79E +308. This maps to Double.</td>
</tr>
<tr>
<td>dBaseDecimal</td>
<td>Provides a floating point number which is treated internally the same as a Double. The dBaseDecimal has a fixed precision and scale when persisted in a table. This is a legacy data type derived, as its name suggests, from the dBase file formats. This maps to Double.</td>
</tr>
<tr>
<td>Boolean</td>
<td>Provides a boolean value. This maps to Boolean.</td>
</tr>
<tr>
<td>String</td>
<td>Provides a variable-length, null terminated UNICODE string value. This maps to String.</td>
</tr>
<tr>
<td>Date</td>
<td>Provides a date value. The Date type is implemented as a structure in the MapInfo.Data namespace.</td>
</tr>
<tr>
<td>DateTime</td>
<td>Provides a combined date and time value. The DateTime type is mapped to System.DateTime.</td>
</tr>
<tr>
<td>Time</td>
<td>Provides a time value. Supports the Time type in MapInfo Professional tables (TAB files) version 9.0 and later. The Time type is implemented as a structure in the MapInfo.Data namespace.</td>
</tr>
<tr>
<td>FeatureGeometry</td>
<td>Provides a FeatureGeometry.</td>
</tr>
<tr>
<td>Binary</td>
<td>Provides an array of binary data. This maps to an Array of Byte values.</td>
</tr>
<tr>
<td>Key</td>
<td>Provides a key from a table. This is the data type of the Key pseudo column on a Table.</td>
</tr>
<tr>
<td>CoordSys</td>
<td>Provides a coordinate system. This type exists only for the purposes of binding a coordinate system object to an MICommand for functions which require the specification of a coordinate system.</td>
</tr>
<tr>
<td>Style</td>
<td>Provides an instance of a Style class. See MapInfo.Styles. Style. This is the data type of the Style object stored in the style column on a Table.</td>
</tr>
<tr>
<td>Raster</td>
<td>Provides a RasterInfo from table's raster column. This is the data type of the RasterInfo object stored in raster column on a Table.</td>
</tr>
</tbody>
</table>
Chapter 10: Working with Data

The following sections provide you with MI_Key, MI_Geometry, and MI_Style column information.

**MI_Key**

All tables have a pseudo column named MI_Key which returns instances of Key. The MI_Key pseudo column is similar in concept to the rowid pseudo column in MapInfo Professional and MapBasic. Unlike rowid, this column is not a numeric column. A Key instance may be converted to or from a string literal.

**MI_Geometry**

A Geometry column object in a table, feature, or feature collection contains FeatureGeometry objects and specifies properties such as the coordinate system of the column and the entire bounds of all the geometry objects it contains.

Geometry columns for most table types are given the name “Obj”. To be compatible with previous versions of MapX and MapInfo Professional, the alias “Obj” is resolved to the first GeometryColumn in the table. Additionally, the alias “MI_Geometry” may also be used for any table to refer to the same column that “Obj” refers to.

**MI_Style**

Tables with a Geometry column also have a column with the name “MI_Style”, or if not found, from the first column with type MIDbType.Style. This column is used to hold the style information for Geometry objects such as line width for polygons and symbol size for points. This column cannot be updated independently. The Style and Geometry columns must be updated at the same time.

The MI_Style column is created automatically when you are opening a table in MapInfo native format (.TAB). For all other table types, you must specifically create the column. If you use MapInfo.Data.ColumnFactory.CreateStyleColumn it will create a column with the name (alias) of “MI_Style” and a data type of MIDbType.Style.

When using MSQL to insert rows into a table, be sure to include the MI_Style column in the insert statement. See the code example below:

```csharp
Table tab = MapInfo.Engine.Session.Current.Catalog.GetTable("MapViewe");
    TableInfo ti = TableInfoFactory.CreateTemp("Test", ((MapInfo.Data.GeometryColumn)tab.TableInfo.Columns["Obj"]).CoordSys);
    Table tabTemp = MapInfo.Engine.Session.Current.Catalog.CreateTable(ti);

    MIConnection conn = new MIConnection();
    conn.Open();
    MICommand comm = conn.CreateCommand();
    comm.CommandText = "Insert Into " + tabTemp.Alias +
```
Catalog

The Catalog is essentially the manager of the MapXtreme data access model. The Catalog holds a list of tables that are currently open in the MapXtreme Session. Tables are also opened, created and closed from the Catalog. The Catalog can be thought of as a single database holding all the tables opened in it, regardless of their actual data source.

Each MapXtreme Session manages a single Catalog.

Catalog initially contains no tables. When a table is opened, an alias (or name) is assigned to the table or provided by the caller. The alias is used to identify the table in queries and other operations.

Tables can be mappable (contain a spatial component) or be non-mappable and contain only data columns. The MapXtreme Catalog can open both types and use either in queries and joins.

Catalog provides facilities for creating new table definitions and enumerating through tables which are currently opened. Catalog also contains search methods that can be used to access data in open tables.

The Catalog has an SQL engine that allows you to select, insert, update, and delete tables and data within tables. The SQL engine allows you to join any tables defined in the catalog (for example, Native to SQLServer, or SQLServer to Oracle). The Catalog handles the integration from various sources so you don’t have to. This is a powerful tool when organizing data from various sources.

The MapXtreme Catalog is exposed through the MapInfo ADO.NET Data Provider. Access to tables and result sets is controlled through this interface. See MapInfo ADO.NET Data Provider on page 187.

Code Sample

The following example illustrates how to access the Catalog off the MapXtreme Session object, open some tables and enumerate through all the tables in the Catalog followed by only the editable tables in the Catalog.

VB example:

Public Shared Sub MapInfo_Data_Catalog()
    ' Catalog is accessible off the Session object
    Dim catalog As Catalog = Session.Current.Catalog

    ' Open a bunch of tables
    Dim table As Table = catalog.OpenTable("States.tab")
    table.SessionInfo.ReadOnly = True ' Make states ReadOnly
    table = catalog.OpenTable("world.tab")

    MessageBox.Show(comm.CommandText);
    int numChanged = comm.ExecuteNonQuery();

    mapControl1.Map.Layers.Add(new FeatureLayer(tabTemp));
    mapControl1.Map.SetView(mapControl1.Map.Layers["Test"] as FeatureLayer);

    ")
}
table = catalog.OpenTable("worldcap.tab", "World Capitals")

' Enumerate the catalog directly - includes All tables
Dim t As Table
For Each t In catalog
    Console.Out.WriteLine("Table : {0}", t.Alias)
Next
Console.Out.WriteLine()

' Now enumerate through only tables that are editable (not ReadOnly)
Dim tEnum As ITableEnumerator = _
catalog.EnumerateTables(TableFilterFactory.FilterEditableTables())
While tEnum.MoveNext()
    Console.Out.WriteLine("Table: {0}", tEnum.Current.Alias)
End While

Session.Current.Catalog.CloseAll()
End Sub

---

**Supported Table Types**

One of the strengths of MapXtreme is its ability to access data "where it lives." This means we strive to handle a wide variety of data formats. Here are the supported table types in MapXtreme:

<table>
<thead>
<tr>
<th><strong>MapInfo .TAB format</strong></th>
<th><strong>MapInfo native table format.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This file-based table may have an associated .MAP file containing FeatureGeometry and Style information. Non-spatial data is stored in a .DAT file. See <a href="#">Creating a New Table on page 173</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>dBase</strong></th>
<th>Data stored in a dBase file.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The table may have an associated .MAP file containing FeatureGeometry and Style information. Non-spatial data is stored in a .DBF file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>ASCII</strong></th>
<th>Data stored in a delimited ASCII text file. The maximum string length is 255 characters (including up to two quotation marks). ASCII tables are Insert only.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The table may have an associated .MAP file containing FeatureGeometry and Style information. Non-spatial data is stored in a .ASC file.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>MS Access</strong></th>
<th><strong>Microsoft Access database table.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This file-based table located inside of a Microsoft Access .MDB database may have an associated .MAP file containing FeatureGeometry and Style information. Non-spatial data is stored in an Access file.</td>
</tr>
</tbody>
</table>
### Supported Table Types

<table>
<thead>
<tr>
<th>Table Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shapefile</strong></td>
<td>An ESRI Shapefile table.&lt;br&gt;These tables are read-only and support two-dimensional geometries. Non-spatial attribute data is stored in .DBF file format. FeatureGeometry values stored in ESRI .SHP file format. MapXtreme does not have access to the spatial index.</td>
</tr>
<tr>
<td><strong>MemTable</strong></td>
<td>An in-memory storage of non-spatial attribute data.&lt;br&gt;FeatureGeometry data and indices are stored on disk. These are temporary tables—all data is lost when the table is closed. MemTables are serializable. The data can be persisted in workspaces (data only; keys may be changed when relogging). This table type is available as a storage format to be used when caching. There is no .TAB file equivalent to a MemTable. See <a href="#">Create a Temporary MemTable on page 175</a>.</td>
</tr>
<tr>
<td><strong>RDBMS Server</strong></td>
<td>A spatial table stored in a remote database management system (such as SQL Server or Oracle).&lt;br&gt;The table is defined by a native SQL SELECT statement. MapXtreme performs query parsing and modification. Caching is enabled by default. Supported protocols (toolkits) include: OCI (Oracle Spatial) and ODBC (SpatialWare and XY). See <a href="#">Accessing Data from a DBMS in Chapter 12 on page 209</a>.&lt;br&gt;See <a href="#">Chapter 2: Getting Started</a> for supported RDBMS.</td>
</tr>
<tr>
<td><strong>ADO.NET</strong></td>
<td>A table of non-spatial data that is based upon an ADO.NET DataTable or IDbCommand.&lt;br&gt;This table type supports many different data providers with provider-specific implementations. ADO.NET is the choice when there is no MapXtreme supported data provider. ADO.NET is designed to support both Connected (IDBCommand) and Disconnected (DataTable) ADO.NET models. IDBTables are read-only. Cache may be applied forcibly (implicit keys). DataTables are editable and run-time serialization is supported. See <a href="#">Using an ADO.NET Data Provider on page 178</a>.</td>
</tr>
<tr>
<td><strong>Raster</strong></td>
<td>A table containing a raster image.&lt;br&gt;This typically provides a base map for other spatial table types. Tables have only a single record and a fixed column schema (RasterInfo, MI_Geometry, MI_Style). These tables may be joined with vector tables using spatial predicates (for example, “within”). See <a href="#">Working with Rasters and Grids in Chapter 18 on page 307</a>.</td>
</tr>
</tbody>
</table>
### Grid
A table containing a grid image.

This table type provides a base map for other spatial table types. Tables have only a single record and a fixed column schema (GridInfo, MI_Geometry, MI_Style). These tables may be joined with vector tables using spatial predicates (for example, “within”). GridRead class provides access to grid cell values. See *Working with Rasters and Grids in Chapter 18 on page 307*.

### WMS
A table containing an image from a Web Map Service (WMS).

This table type provides a base map for other spatial table types. Tables have only a single record and a fixed column schema (GridInfo, MI_Geometry, MI_Style). These tables may be joined with vector tables using spatial predicates (for example, “within”). WMS tables are accessed like dynamic raster through a MapInfo.Wms.WmsClient. See *Web Map Service in Chapter 21 on page 353*.

### Seamless
A table that combines two or more base tables with contiguous geography. It displays as a single map layer.

Seamless tables are specifically tuned for spatial queries, such as drawing a map, which uses seamless tables for optimally querying appropriate component tables. Component tables that make up a seamless table may be vector or raster. They must all have the same schema. They are read-only. The underlying component tables cannot be modified directly. Sorting and aggregating operations examine every record of every component tables (could be have a significant performance impact when working with vector tables.)

### View
A view based on a MapInfo SQL Select statement (not a native SQL supported by Server tables). See *View Tables on page 183*.

### ResultSet
A table containing the results of a search. ResultSet is used exclusively for IResultSetFeatureCollections. See *Result Sets on page 183*.

---

### Working with Catalog and Tables

This section covers some basic table operations, including:

- Locating Open Tables
- Closing a Table
- Packing a Table
- Listening to Table and Catalog Events

See also the MapInfo.Data.Table class in the MapXtreme Developer Reference.
Locating Open Tables

To locate open tables, you must enumerate the catalog. This is done by using the methods in the following sections.

**Catalog.GetTable**

The Catalog.GetTable method returns the Table object referenced by the TableAlias parameter. This must be a table which has already been opened. If no such table is found (or the table has subsequently been closed), then the method returns null.

**Catalog.Item (Indexer)**

Catalog.Item property can be used as an indexer for locating a Table by its Alias. This is functionally equivalent to using the Catalog.GetTable method but generates code that is easier to read. The Alias must specify a table which has already been opened.

**TableEnumerators**

Table enumerators may be obtained through the various overloaded EnumerateTables methods. A table enumerator may be created with a filter. The filter determines which tables are actually included in the enumeration while the enumerator simply provides the mechanics of enumeration. You can create your own table filters to use in the TableEnumerator. You can also create your own table enumerator by implementing the ITableEnumerator interface.

Closing a Table

Three methods are available to close tables. Catalog.CloseAll closes all open tables while Catalog.CloseTable closes a single, open table. The Table class also has a Close method.

Packing a Table

The MapInfo.Data.Table.Pack method removes records from the table that were previously marked for deletion. When the table is packed, the table's TablePacked event is raised. The arguments for the event indicate whether or not the table's keys changed as a result of the pack (which would be caused by removing deleted records). Keys only change if the PackType includes RemoveDeletedRecords and if there actually were deleted records in the middle of the table. If the only deleted records in the table are at the end of the table, then no keys are changed. The event does not indicate that keys were changed.

**Note** Since ResultSet tables hold collections of keys, these are vulnerable to pack operations on the table from which they were derived. The ResultSet is no longer valid if the keys have changed.

PackType Enumeration provides you with the following options.

- PackGeometry – Indicates that the geometry objects are packed. Packing the objects attempts to remove as much unused space as possible. A fully packed RTree (the spatial index used to spatially access the geometry objects) may reduce performance by causing many more unnecessary reads. To balance disk space and processing speed, packing the geometry objects
may continue to leave some unused space in the RTree. Also note: a packed RTree results in a slight performance penalty for insert and update operations as there is a higher likelihood that the RTree needs to be expanded.

- RebuildGeometry – Rebuilding the geometry objects removes unused space that has resulted from a series of insert, update, and/or delete operations. Unlike packing the geometry objects, this option intentionally leaves unused space in the RTree index to improve the performance of future insert and update operations.

- PackIndex – Non-spatial indices are maintained as B*trees. These structures do not always have filled internal or leaf nodes. This is intentional by default to allow room for the index to accommodate insert and update operations without requiring a significant restructuring of the index. The unused space may be exacerbated by the occurrence of insert, update, or delete operations. Packing an index fully packs every internal and leaf node (except possibly the “last” node). This option reduces the disk space used by the index as much as possible and also improves the read-performance of the index. There is a performance penalty for insert and update operations on a fully packed index.

- RebuildIndex – Rebuilding an index does not fully pack the internal and leaf nodes like the PackIndex option. Instead, rebuilding an index recreates the index with the amount of unused space that is intentionally put into the index to balance disk space, read performance, and modify performance. After several modification operations, an index may contain a considerable amount of unused space. This option regains that unused space.

- RemoveDeletedRecords – Some data sources, including MapInfo Native and dBase data sources, do not physically remove records when they are deleted. To physically remove the deleted records, the table must be packed with this option specified. The record number is typically used as the record key for these data source types. Removing deleted records from a table may cause keys to become invalid since they may change as a result of the pack.

- CompactDb – If the table’s data source is Microsoft Access (TableType of Access), then the MDB file containing the table’s data may also be compressed using the Pack method and specifying this option.

- All – This is a convenience option that is equivalent to PackGeometry | PackIndex | RemoveDeletedRecords.

### Listening to Table and Catalog Events

Table exposes several events which applications may subscribe to. They are:

- RowInsertedEvent – Occurs when a new row is added to the table.
- RowUpdatedEvent – Occurs when an existing row in the table is updated.
- RowDeletedEvent – Occurs when a row in the table is deleted.
- TablePackedEvent – Occurs when the table is packed.
- TableCloseRequestEvent – Occurs when the table has been asked to close.
- TableIsClosingEvent – Occurs when the table is closing.
- TableClosedEvent – Occurs when the table is closed.

Catalog also exposes the following events.

- TableOpenedEvent – Occurs when a table is opened.
- TableCreatedEvent – Occurs when a new table is created.
- TableIsClosingEvent – Occurs when the table is closing.
Table Metadata (TableInfo)

The TableInfo class in the MapInfo.Data namespace is an abstract base class that contains information, or metadata, about an existing table, including:

- Columns – number, names, data types, etc.
- Table alias, and description and pathname of the data source.
- Client metadata (the information between the begin_metadata/end_metadata tags in the TAB file).

TableInfo is used to open tables and create new tables. It is also used for retrieving information about the open table.

Classes that derive from TableInfo include provider-specific metadata. There is a TableInfo implementation for every MapXtreme supported table type. See Data Sources on page 176.

TableInfo instances may be constructed manually, or from a .TAB file definition (without opening the table), as shown below.

```
TableInfo.CreateFromFile(…)
```

TableInfo contains properties for enabling Table Services, including caching and making a table mappable via a spatial schema. See Working with the Cache in Chapter 12 on page 219 and Making Tables Mappable on page 185.

Examining TAB File Metadata

TAB file metadata is accessible and editable. The TableInfo class can be obtained from the Table to get information about the table structure. The following code demonstrates how to get the metadata for an open table. The code also demonstrates how the geometry column can be used to determine the coordinate system and bounds of the table.

VB example:

```
Public Shared Sub MapInfo_Data_TableInfo2()
    ' Get the metadata for an open table
    Dim ti As TableInfo = Session.Current.Catalog("states").TableInfo

    ' Print out some information to the console
    Console.Out.WriteLine("Table Alias={0}, Datasource={1}, Description={2}, Type={3}", 
        ti.Alias, ti.DataSourceName, ti.Description, ti.TableType)
```

Creating a New Table

The following sections illustrate how to create a permanent native table, a temporary native table, and a temporary MemTable.

Create a New Permanent Native Table

The MapInfo.Data.Table/TableInfo property for a MapInfo native table returns an instance of TableInfoNative. A native table is a MapInfo .TAB file. This class may be used to access properties that are specific to native table types. New instances of this class may be created and used to construct new tables. See also Data Sources on page 176.

Note the use of the ColumnFactory class. This is provided to help you know which arguments are necessary for different data types. For example, a geometry column requires a coordinate system.

VB example:

```vbnet
Public Shared Sub MapInfo_Data_TableInfoNative()
    Dim ti As TableInfoNative = New TableInfoNative("NewTable")
    ti.TablePath = "c:\data\Capitals.TAB"
    ti.Columns.Add(ColumnFactory.CreateIndexedStringColumn("Capital", _
                                                      25))
    ti.Columns.Add(ColumnFactory.CreateDoubleColumn("Pop_Grw_Rt"))

    ' Make the table mappable
    ti.Columns.Add(ColumnFactory.CreateStyleColumn())
                                                                     62, 7, 0")
    ti.Columns.Add(ColumnFactory.CreateFeatureGeometryColumn(Robinson))

    ' Note we do not need to (nor should we) add a column of type Key.
    ' Every table automatically contains a column named "MI_Key".
    Dim table As Table = Session.Current.Catalog.CreateTable(ti)
```

' Print out some information about each column
Dim col As Column
For Each col In ti.Columns
    Console.Out.WriteLine("Column {0} Type={1} Width={2}", _
                        col.Alias, col.DataType, col.Width)
    ' If the column is a geometry column, print csys and bounds.
    If col.DataType = MIDbType.FeatureGeometry Then
        Dim geocol As GeometryColumn = col
        Dim csys As MapInfo.Geometry.CoordSys = geocol.CoordSys
        Console.Out.WriteLine("CSys : {0}", csys.MapBasicString)
        Dim dr As MapInfo.Geometry.DRect = geocol.Bounds
        Console.Out.WriteLine("Bounds=({0},{1}),({2},{3})", dr.x1, _
                             dr.y1, dr.x2, dr.y2)
    End If
Next
End Sub
Create a Temporary Native Table

VB example:

Public Shared Sub MapInfo_Data_TableInfo3(ByVal conn As MIConnection)
    Dim ti As TableInfoNative = New TableInfoNative("NewTable")
    ti.Temporary = True
    Dim col As Column

    col = New Column
    col.Alias = "FString30"
    col.DataType = MIDbType.String
    col.Indexed = True
    col.Width = 30
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FInt32"
    col.DataType = MIDbType.Int
    col.Indexed = True
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FInt16"
    col.DataType = MIDbType.SmallInt
    col.Indexed = True
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FDouble"
    col.DataType = MIDbType.Double
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FDateTime"
    col.DataType = MIDbType.Date
    ti.Columns.Add(col)

    col = New Column
    col.Alias = "FBoolean"
    col.DataType = MIDbType.Boolean
    ti.Columns.Add(col)

    ' Note we do not need to (nor should we) add a column of type Key.
    ' Every table automatically contains a column named "MI_Key".
    Dim miTable As Table = conn.Catalog.CreateTable(ti)
End Sub
Create a Temporary MemTable

The MapInfo.Data.Table.TableInfo property for a memory table returns an instance of TableInfoMemTable. This class may be used to access properties that are specific to memory table types. New instances of this class may be created and used to construct new tables.

Data in forms such as XML or GML from a Web service can brought into the Catalog and used in this fashion. It can be converted to a MultiPolygon, LineString, Point, or other Geometry via the MapXtreme API. MapXtreme then turns the Geometry into a FeatureCollection, and, in turn, saves it to a memTable or native TAB format.

This approach is appropriate also if you wish to make data available for use in MapXtreme, but not necessarily for map display.

VB example:

```vbnet
Public Shared Sub MapInfo_Data_TableInfoMemTable()
    Dim ti As TableInfoMemTable = New TableInfoMemTable("NewTable")
    ' Note: The TablePath property does not apply - it can be set but it _
    ' is meaningless.
    ti.Columns.Add(ColumnFactory.CreateIndexedStringColumn("Capital", _
        25))
    ti.Columns.Add(ColumnFactory.CreateDoubleColumn("Pop_Grw_Rt"))
    ' Make the table mappable
    ti.Columns.Add(ColumnFactory.CreateStyleColumn())
    Dim Robinson As CoordSys = _
            7, 0")
    ti.Columns.Add(ColumnFactory.CreateFeatureGeometryColumn(Robinson))
    ' Note we do not need to (nor should we) add a column of type Key.
    ' Every table automatically contains a column named "MI_Key".
    Dim table As Table = Session.Current.Catalog.CreateTable(ti)
End Sub
```

Adding Expression Columns to a Table

Use the MapInfo.Data.Table.AddColumns method to add expression columns to a table. The form of AddColumns that takes a Columns object creates temporary columns based on expressions comprised of functions, operators, literal values, and other columns on the table. All instances of Column in the columns argument must have an expression string specified.

Note

TableAddColumns is not supported for the following table types: Server, View, Seamless, AdoNet, ResultSet, or Drilldown. MapXtreme checks for the table and throws an exception if it encounters one of these table types.

VB example:

```vbnet
Public Shared Sub MapInfo_Data_TableAddColumns(ByVal miTable As Table)
    Dim NewCols As Columns = New Columns
        MI_Area(Obj, 'sq mi', 'Spherical')")))
End Sub
```
MI_Area(Obj, 'sq mi', 'Spherical'))
miTable.AddColumns(NewCols)
End Sub

The expression string “Pop_1990 / MI_Area(Obj, 'sq mi', 'Spherical')” represents derived information that will be placed in the temporary column. It says 'For each record divide population by area in square miles to yield the population density.' The SQL function MI_Area() in the expression will derive the area from the geometry of the record.

Using the AddColumns method may offer performance improvements in desktop applications where the join can be performed once, rather than on each subsequent access (as in the case of a view).

For more information and code examples, see the MapInfo.Data.Table.AddColumns class in the Developer Reference Help system.

For more information on creating expressions, see Chapter 11: Creating Expressions.

Data Sources

The following table lists the data sources supported by MapXtreme. Each type of data source is accessed by a specific data provider, a TableInfo class that is derived from MapInfo.Data.TableInfo. For a short summary of each data source see Supported Table Types on page 167.

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native (MapInfo.TAB)</td>
<td>TableInfoNative</td>
</tr>
<tr>
<td>dBase</td>
<td>TableInfoDbase</td>
</tr>
<tr>
<td>MS Access</td>
<td>TableInfoMSAccess</td>
</tr>
<tr>
<td>ASCII</td>
<td>TableInfoAscii</td>
</tr>
<tr>
<td>RSBMS Server</td>
<td>TableInfoServer</td>
</tr>
<tr>
<td>ESRI Shapefile</td>
<td>TableInfoShapefile</td>
</tr>
<tr>
<td>Seamless</td>
<td>TableInfoSeamless</td>
</tr>
<tr>
<td>Raster</td>
<td>TableInfoRaster</td>
</tr>
<tr>
<td>Grid</td>
<td>TableInfoGrid</td>
</tr>
<tr>
<td>WMS</td>
<td>TableInfoWMS</td>
</tr>
<tr>
<td>ADONET</td>
<td>TableInfoAdoNet</td>
</tr>
<tr>
<td>MemTable</td>
<td>TableInfoMemTable</td>
</tr>
<tr>
<td>View</td>
<td>TableInfoView</td>
</tr>
<tr>
<td>ResultSet</td>
<td>TableInfoResultSet</td>
</tr>
</tbody>
</table>
Choosing the Correct Data Source

Choosing the correct data source can make a difference in your application’s performance. In some cases you will not have a choice, such as native MapInfo files (.TAB), but in other cases there may be multiple choices. In most cases, you will be using a supported data provider for the data source. In cases where the data is not accessible through one of these, you may be able to use the MapInfo ADO.NET data provider. This is the same data provider interface that the Catalog uses to retrieve data.

Each data source has certain performance characteristics. Native tables offer the best access and map drawing times. Data is stored locally on the system and optimized for your current operation. Other file-based table types perform well, depending on current hardware and file size.

Methods for Accessing Data

MapXtreme provides several ways to bring data into the Catalog:

- Direct access to data sources
- Access via an ADO.NET data provider (TableInfoAdoNet)
- XML/GML from third-party web services

The best method to access data is to open it directly using one of the TableInfo classes that are specific to where your data resides.

Use the second method (TableInfoAdoNet) to access data that is not internally supported but has an ADO.NET provider.

A third method allows developers to integrate data to the Catalog who may interact with HTTP services that return XML or GML.

Direct Access to Data Sources

MapXtreme provides native support for accessing data stored in file-based table formats and RDBMS servers, such as SQL Server and Oracle. In the case of file-based access, provide the path and filename in the appropriate TableInfo instance (TableInfoNative, TableInfoDataBase, TableInfoMSAssess, TableInfoAscii).

For direct access to data stored in RDBMS serves, use the TableInfoServer class to define the connection string and an SQL statement to execute on the remote table. Internally, MapXtreme uses ODBC or OCI to access the remote database.

TableInfoServer will open a connection to the server, query the table's metadata, and create the appropriate table definition with any spatial characteristics that are defined on the remote server. This tends to be the best performing method with remote data. Internally, MapXtreme can access only the data necessary to perform the current operation. During a map draw, MapXtreme will construct a query that returns only the geometry column, and not the data columns. This minimizes the network traffic. If caching is on, then this is only an issue for the first access, since all subsequent requests will come from the cache. See Accessing Data from a DBMS in Chapter 12 on page 209.
Access via an ADO.NET Data Provider

The second data access method is to use an ADO.NET data provider. This requires the definition of ADO.NET classes for data retrieval. Only non-mappable tables may be supplied as an AdoNet table. Non-mappable tables are those that do not contain geometry information about the data. Tables retrieved from an ADO.NET provider, however, can be made mappable by applying a SpatialSchema to the table definition. In this method, the MapXtreme DataAccess engine calls the ADO.NET data provider whenever data is requested by a user. This tends to be a slower method of accessing data. However, when used in conjunction with caching, it performs well. See Using an ADO.NET Data Provider on page 178.

Data from Third-Party Web Services

MapXtreme can integrate Web service XML or GML output into the Catalog for use in a MapXtreme desktop or web application. Data can be brought into the Catalog and converted to a MultiPolygon, LineString, Point, or other Geometry via the MapXtreme API. MapXtreme then turns the Geometry into a FeatureCollection, and, in turn, saves it to a memTable or native TAB format.

This approach is appropriate also if you wish to make data available for use in MapXtreme, but not necessarily for map display.

Using an ADO.NET Data Provider

Data that cannot be directly accessed with a specific TableInfo data source can use TableInfoAdoNet. The ADO.NET table can be in one of two forms: DataTable (a collection of rows from a single table kept in-memory and allows read-write access); or IDbCommand (an SQL statement executed at the data source that yields read only, dynamic data).

Accessing Data in a DataTable

When using a DataTable, the Catalog is essentially holding on to a reference to the DataTable you supply to the call to Catalog.OpenTable (using the TableInfoAdoNet class). DataTables are editable using the MapInfo ADO.NET Data Provider by issuing Insert, Update, and/or Delete commands. Your application may continue to access the DataTable directly as well. Note, however, that the structure of the table should not be changed while the Catalog has a reference to it. Also note that changes to the data outside of the MapInfo Data Provider (e.g., without using the MICommand to issue Insert, Update, or Delete commands) will not result in the raising of the insert, update, or delete table events.

The DataTable contains almost enough information for the Catalog to define the table. For string columns, however, the Catalog needs to assign a length to this field. The length would be used when constructing temporary indices, temporary tables for aggregation, etc. For these types of operations, it is important to get the string length correct. The DataColumn has a MaxLength property that should be set to indicate the maximum length string the column could hold. If not set, this value defaults to -1 in which case the value of 254 is used. Before checking the MaxLength property, the catalog looks to see if the DataColumn has a property defined in its ExtendedProperties collection with the name “StringWidth”. If found, the value for this property is used as the column's width.

This example illustrates how to create a MapInfo Table whose data is stored in a DataTable.

VB example:
Public Shared Sub MapInfo_Data_TableInfoAdoNet(ByVal connection As _
MIConnection)
    ' Create a new DataTable.
    Dim dt As DataTable = New DataTable("CityData")
    Dim dc As DataColumn
    dc = dt.Columns.Add("City", Type.GetType("string"))
    dc.MaxLength = 30
    dc = dt.Columns.Add("Country", Type.GetType("string"))
    dc.MaxLength = 30
    dc = dt.Columns.Add("Continent", Type.GetType("string"))
    dc.MaxLength = 30
    dc = dt.Columns.Add("Population", Type.GetType("string"))

    ' Populate the DataTable...
    dt.Rows.Add(New Object() {"Jim", "0001", -73.77, 42.67, 314.15})
    dt.Rows.Add(New Object() {"Tom", "0002", -73.87, 42.82, 628.3})

    ' Now open a MapInfo Table which accesses this DataTable
    Dim ti As TableInfoAdoNet = New TableInfoAdoNet("Cities")
    ti.ReadOnly = False
    ti.DataTable = dt
    Dim table As Table = connection.Catalog.OpenTable(ti)
End Sub

**Saving and Restoring ADO.NET objects**

Saving and Restoring ADO.NET tables can be accomplished in MapXtreme by using the steps outlined below. Explicit serialization/deserialization of ADO.NET tables is not supported due to limitations on restoring the underlying System.Data.DataTable. An ADO.NET table is a table in the Catalog which was created using a MapInfo.Data.TableInfoAdoNet object.

The proper method of serializing/deserializing the ADO.NET based MapInfo table is in the SaveState method. Serialize all tables that reference the ADO.NET table (i.e., ViewTables or joins) and then close the ADO.NET table. In the RestoreState method, re-create the ADO.NET MapInfo table with the same name and then deserialize any dependent MapInfo tables. Order is important because you have to create the ADO.NET table prior to restoring any other tables.

In the context of a MapXtreme Web Application which implements manual state management, follow these steps to save and restore an ADO.NET table between client requests.

**Note** The steps outlined below refer specifically to ADO.NET tables created from a DataTable.

1. In the MapInfo.WebControls.StateManager.SaveState method:
   a. Place the ADO.NET DataTable or DataSet into the HTTPSession instance.
   b. Save any MapInfo.Data.TableInfoView or MapInfo.Data.TableInfoResultSet tables that are dependent on the ADO.NET table using the ManualSerializer.SaveMapXtremeObjectIntoHttpSession.
   c. Close the ADO.NET table in the Catalog.

2. In the MapInfo.WebControls.StateManager.RestoreState method:
   a. Create a new TableInfoAdoNet object based on the DataTable retrieved from the HTTPSession instance.
   b. Open a new ADO.NET table in the catalog based on the TableInfoAdoNet object with the same name as the original.
c. Restore any MapInfo.Data.TableInfoView or MapInfo.Data.TableInfoResultSet tables based on the ADO.NET table using ManualSerializer.RestoreMapXtremeObjectFromHttpSession.

It is important to remember that the order you save and restore is critical to the proper creation of all the tables and their dependencies. For more information on serialization, see Understanding State Management in Chapter 6 on page 95.

Accessing Data Using an IDbCommand

The second form of ADO.NET table is based on the connected object types in ADO.NET: Connection, Command, and DataReader. MapInfo Tables constructed in this fashion are read-only. These types of tables are created by passing to the Catalog an IDbCommand object that is already configured to return all of the data that is to comprise the table. When the table is initially created (by calling Catalog.OpenTable), ExecuteReader is called on the IDbCommand. The resulting data reader is used to determine the columns and their data types. All subsequent cursor requests (other than cursors which retrieve a specific record - called a key fetch) also call ExecuteReader to fetch the data to satisfy the cursor. Notice that this may be very inefficient. If at all possible, use one of the other table types to access your data.

Since the Command-based form of the ADO.NET table is designed to use the generic interfaces without requiring any specific knowledge of any particular implementation of these interfaces, the table also does not assume that the IDbCommand.CommandText is any form of standard SQL. In fact, it may not be SQL at all. This table type does not access, parse, or modify the CommandText. This means that this table type has no mechanism for knowing which column(s) in the results formulate a unique, non-null key value. For this type of table, it is required to tell the table which column(s) constitute the key. This is accomplished by specifying the KeyType as Explicit and setting the KeyColumns property.

There are many operations inside the MapInfo Data Provider which require the retrieval of a specific record by key (also referred to as a key fetch). Select statements with a where clause of the form MI_Key = '5' is a simple example in which we need to retrieve the record whose MI_Key column can be represented by the string literal '5'. Key retrievals are very common in mapping selections, labeling, and scrolling in a MIScrollableReader (in which case the reader may be scrolling through a list of key values). MapInfo tables are dependent upon the ability to efficiently fetch records by key value. Just as the Command-based form of the ADO.NET table does not read, parse, or modify the CommandText of the IDbCommand object that defines the table (the “Sequential” IDbCommand), it has no ability to modify the IDbCommand object to fetch a specific record. Thus, a second IDbCommand object must be supplied for this purpose. The “FetchByKey” IDbCommand object must meet the following requirements:

- When ExecuteReader is called on this command object, it must produce a data reader that has the same columns as the sequential command object and in the same order.
- The FetchByKeyCommand must contain a Parameters collection and must contain one parameter for each member of the key. For example, if the TableInfo.KeyColumns specifies a key as consisting of the “city” and “state” columns, then the FetchByKeyCommand must contain two parameter objects. The first parameter object is assigned a value representing the first column specified in the TableInfo.KeyColumns collection (e.g., a value for “city”), the second parameter object is assigned a value representing the second column specified in the TableInfo.KeyColumns collection (e.g., a value for “state”), and so on. When ExecuteReader is called on the FetchByKeyCommand, the reader must return the record which represents the specified key.
This example illustrates how to create a MapInfo Table that accesses data through the ADO.NET connected command objects.

**VB example:**

```vbnet
Public Shared Sub MapInfo_Data_TableInfoAdoNet2(ByVal connection As MIConnection)
    Dim ti As TableInfoAdoNet = New TableInfoAdoNet("EuropeanCities")
    Dim _conn As OleDbConnection = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Data\EuropeCities.mdb")
    Dim selectQuery As String = "SELECT City, Country, Continent, Pop_1994 FROM EuropeCities"
    Dim _OleDbCommand As OleDbCommand = New OleDbCommand(selectQuery)
    _OleDbCommand.Connection = _conn

    selectQuery = selectQuery + " where City = @City AND Country = @Country"
    Dim _OleDbKeyCommand As OleDbCommand = New OleDbCommand(selectQuery)
    _OleDbKeyCommand.Parameters.Add("@City", OleDbType.Char)
    _OleDbKeyCommand.Parameters.Add("@Country", OleDbType.Char)
    _OleDbKeyCommand.Connection = _conn

    ' The MapInfo Table will Open/Close the connection as necessary.
    ' If this is expensive the application could open the connection
    ' before opening the table and closing the connection after the
    ' table is closed.
    ti.SequentialCommand = _OleDbCommand
    ti.FetchByKeyCommand = _OleDbKeyCommand

    ' Tell the table which column(s) constitute a key - for this table
    ' it is a compound key consisting of values from the City and County
    ' columns.
    Dim sc As StringCollection = New StringCollection
    sc.Add("City")
    sc.Add("Country")
    ti.KeyColumns = sc
    ti.KeyType = KeyType.Explicit

    ' Ask the Catalog to open the table.
    Dim tbl As Table = connection.Catalog.OpenTable(ti)

    ' Now the MICommand object may be used to select data from the table
    ' (by the name EuropeanCities since that is the alias we assigned to
    ' it). The data in this table may be joined with any other table and
    ' it may be used as source data in a call to AddColumns to populate
    ' temporary columns with data from this table.
    End Sub
```
Data Binding

Data binding is the process of bringing data from a data source into MapXtreme. Data binding of external data (ADO.NET and other legacy sources) to MapInfo.Data.Table is accomplished by opening an ADO.NET DataTable as a Table using TableInfoAdoNet. The table can then be joined with another table, joined to itself or use Table.AddColumns to bind columns to a second table.

To join a table to itself, following this example:

```plaintext
Select ... From T as A, T as B Where A.X = B.Y
```

If an application has data stored in a DataTable or data that is accessible through an ADO.NET data provider, that data can be presented to the Catalog and treated as a MapInfo table. This would be primarily useful if the data were not accessible through one of the other table types.

For example, if the data is stored in a dBase file, Microsoft Access table, or is accessible through ODBC or Oracle's OCI interface, it is recommended that those TableInfo types be used to access the data. Data which cannot be accessed through one of these types of tables, but that can be loaded into a DataTable or is accessible through some ADO.NET Data Provider that implements the Command, Parameter and DataReader object types can still be accessed by the Catalog.

An application may need to make data available as a MapInfo native table so that queries can be executed to join the data with other MapInfo table data. It may also need to be made available to the Catalog so it can be used as the source data in a call to the Table.AddColumns.

**Code Example: How to Join Data from an Oracle Table**

```csharp
public Shared Sub MapInfo_Data_TableAddColumns5(ByVal map As Map)
    Dim Connection As MapInfo.Data.MIConnection = New _
        MapInfo.Data.MIConnexion
    Connection.Open()

    'Add the USA table to the map
    map.Load(New _
        MapInfo.Mapping.MapTableLoader("C:\\maps\\usa.TAB"))

    Dim lyr As MapInfo.Mapping.FeatureLayer = map.Layers("usa")

    ' Open the table from Oracle

    Dim ti As TableInfoServer = New TableInfoServer("StateCapXY", _
        "SRVR=tempest;UID=tn;PWD=t", "Select * from usa_caps", _
    Dim StateCapXY As Table = Connection.Catalog.OpenTable(ti)

    ' Add the Oracle columns to the USA table
    Dim states As Table = Connection.Catalog.GetTable("usa")
    states.AddColumn(Nothing, MapInfo.Data.BindType.Static, _

    'Create a ranged theme on the USA layer using a field from the Oracle table
    Dim thm As MapInfo.Mapping.Thematics.RangedTheme = New _
MapInfo.Mapping.Thematics.RangedTheme(lyr, _
"pop_1990", "popusa", 4, _
lyr.Modifiers.Insert(0, thm)

'Create a legend to appear on the map
Dim legend As MapInfo.Mapping.Legends.Legend = _
    map.Legends.CreateLegend(New Size(5, 5))
legend.Border = True
Dim frame As MapInfo.Mapping.Legends.ThemeLegendFrame = _
legend.Frames.Append(frame)
frame.Title = "pop"
map.Adornments.Append(legend)
End Sub

For a C# version of this sample or to view the above sample in electronic form, visit the MapXtreme Learning Resources browser located on your Program menu. From the main page of the browser, look under the Learning Resources > Documentation link. Click on the first topic, “Code Snippets How Do I,” to open the help file of code examples. This sample is located under Data Access.

**View Tables**

A view is a way to relate information from one or more tables based on a named select statement. The Catalog allows you to create views based on any table definition. View tables have the following characteristics:

- The data is not copied.
- Access to views always accesses its base tables.
- View is an MapInfo SQL Select Statement with a name (Alias).
- Queries may be joins (forms composite keys).
- Membership in the View is live.
- Exception: Views that aggregate cache the data. Data changed events trigger recomputation.
- Can be serialized and persisted in workspaces.

For more information and code examples, see the MapInfo.Data.TableInfoView class in the Developer Reference Help system.

**Result Sets**

ResultSets are similar to view tables in that both are defined using a MapInfo SQL select statement and have an associated name (Alias). ResultSets, however, have a fixed membership of records based on the evaluation of the where clause (if any) at the time the result set is created. Any access to the data in a ResultSet always reflects the data in the source table. However changes to the source data will not cause the ResultSet to add/remove a record based on the original where clause. ResultSets manage a set of keys internally.

In general ResultSets are lightweight and temporary. Some of the characteristics of result sets are:

- The data is not copied.
Table Metadata (TableInfo)

- Access to result sets always accesses its base tables.
- A ResultSet is a sorted list of keys, a collection of column definitions, and a name.
- Membership in the ResultSet is fixed.
- Exception: ResultSets that aggregate, cache the data. Data changed events trigger recomputation.
- Can be serialized, but not persisted in workspaces.
- ResultSets are vulnerable to Delete and Pack operations.

For more information, see the MapInfo.Data.TableInfoResultSet class in the Developer Reference Help system.

Source Rows

Source rows represent a match between the table records involved in Table.AddColumns. When adding temporary columns to a table, multiple records from the data source may be aggregated together to compute a value for each record in the destination table (also referred to as the bind table). The MapInfo.Data.SourceRows class is a collection of SourceRows that identify the records from the data source that were aggregated together.

SourceRows only exist if the BindType property is DynamicCopy, which indicates that changes to the source data are propagated to the temporary column automatically.

Note Table.AddColumns is not supported for the following table types: Server, View, Seamless, AdoNet, ResultSet, or Drilldown.

See also Adding Expression Columns to a Table on page 175.

The GeoDictionary

The GeoDictionary maintains information about which map entities can be matched to which information. The GeoDictionaries class is a collection of GeoDictionary objects. The MapInfo.Data.GeoDictionary namespace provides support for data autobinding by being a programmatic representation of the GeoDictionary file. The GeoDictionary file contains information about tables (TAB files only). The GeoDictionary is used to automatically determine the table to which application data should be bound. The GeoDictionary is persisted in a file (typically GeoDict.DCT) and is maintained using the GeoDictionaryManager utility application (see Chapter 24: Using the GeoDictionary Manager).

AutoMatching Using the GeoDictionary

The MatchResolver.AutoMatch method in the Data.GeoDictionary namespace initiates the AutoMatching process. It does not call AddColumns, i.e., does not do the binding. A subsequent call to BindColumn is required to perform the autobinding, or a direct call to AutoMatchAndBind.

Auto matching can encounter ambiguous situations. These situations include:

- multiple source columns are detected in the user data
- multiple tables/layers are detected that match the source column
- multiple geosets/workspaces are available for the matched table/layer.
It is the MatchResolver object with which the GeoDictionary communicates during the match process to solve the ambiguity. It provides the matching algorithm. The basic class selects the first or the one with the highest matching percentage. This class is not sealed and client applications may derive their own class from this and override its behavior.

Making Tables Mappable

Tables can either be mappable (contain a GeometryColumn) or non-mappable (no spatial attribute data). Mappable tables are added to a MapXtreme application as a layer in a map. Non-mappable tables, such as customer data, can be made mappable when a GeometryColumn is created for it. MapXtreme provides spatial schemas to accomplish this.

Spatial schemas are services that can be applied to a table to enhance its spatial capabilities. There are two type of spatial schemas: XY and PointRef. Non-mappable tables that have attribute columns that represent X and Y values (such as longitude and latitude) use SpatialSchemaXY and tables that have an attribute column which can be used to reference a record in a mappable table uses SpatialSchemaPointRef.

SpatialSchemaXY

SpatialSchemaXY uses the X and Y values of each record in the table to construct point objects and store them in a temporary column known as MI_Geometry. This spatial schema may be applied to tables of any data source except Seamless, View, and ResultSet.

By having a GeometryColumn, the table can now be displayed as a layer in a Map and used for spatial analysis.

SpatialSchemaXY has the following characteristics:

• The Geometry column is editable.
• Editing the Geometry automatically changes the X and Y values.
• You can define styles for each point in the table.
• You can store the spatial information as a TAB file and open like any other table.

This spatial schema can be used for traditional server XY data without a MapCatalog. (Using a MapCatalog may offer better performance on RDBMS's, since more work is done on the server. See The MapInfo MapCatalog in Chapter 12 on page 221.)

MI_Geometry is a temporary column unless you write out the TAB file explicitly using the TableInfo.WriteToTab method. The schema is automatically regenerated when the table is opened.

VB example:

Public Shared Sub MapInfo_Data_SpatialSchemaXY()
Dim ti As TableInfo = 
  TableInfo.CreateFromFile("c:\data\customers.TAB")
  ' a non-mappable table
Dim xy As SpatialSchemaXY = New SpatialSchemaXY
xy.XColumn = "Xcoord"
xy.YColumn = "Ycoord"
xy.NullPoint = "0.0, 0.0"
  ' Any customer at 0,0 means we don't know their location.
xy.StyleType = StyleType.None
xy.CoordSys = _
SpatialSchemaPointRef

This spatial schema uses a value in the table’s data to create a Point geometry object by matching the value against an equivalent value in a mappable table.

For example, if your table of customers contains addresses with postal codes, the customer records can be tied to the spatial points in a postal code reference table.

SpatialSchemaPointRef is actually a join between two tables, one containing data and the other containing a join column and an object column. The join column contains the same values as the data column in the non-mappable table, such as postal codes. The result of applying SpatialSchemaPointRef is a table that contains a spatial geometry column for records that were previously non-spatial. This geometry column has the following characteristics:

• The data table may match more than one record in the geometry table. When this happens the similar rows are aggregated into a MultiPoint geometry.
• The geometry is the centroid of the geometry from the other table.

SpatialSchemaPointRef has these characteristics:

• The temporary Geometry column is read-only.
• Any edits to a value in the reference table changes the Geometry value in the data table.
• SpatialSchemaPointRef can be applied to any data source except Seamless, View, and ResultSet.
• You can define styles for each point in the table.
• You can store table information as a TAB file and open like any other table.

For more information and code examples, see the MapInfo.Data.SpatialSchemaPointRef class in the Developer Reference Help system.

VB example:

Public Shared Sub MapInfo_Data_SpatialSchemaPointRef(ByVal _
    ByVal _map As _Map)

    ' a non-mappable table
    Dim ti As TableInfo = _
        TableInfo.CreateFromFile("c:\data\customers.TAB")
    Dim pr As SpatialSchemaPointRef = New SpatialSchemaPointRef
        pr.CoordSys = map.GetDisplayCoordSys()
        pr.StyleType = StyleType.None
        pr.RefTable = "us_zips"

    ' the column in RefTable which will match the MatchColumn in my data
    pr.RefColumn = "zipcode"

    ' a column in the Customer table
pr.MatchColumn = "zip"
pr.RefTableLocation = "c:\data\us_zips.tab"

' Now set the spatial schema information before opening the table.
ti.SpatialSchema = pr
Dim table As Table = Session.Current.Catalog.OpenTable(ti)
End Sub

MapInfo ADO.NET Data Provider

MapXtreme provides mechanisms for issuing SQL commands which return record sets from tables using ADO.NET. The MapInfo ADO.NET Data Provider is one mechanism for accessing data in .NET applications in this fashion. For an alternative that uses the Feature class and SearchInfo methods on the Catalog, see Features and Feature Collections on page 191.

The following sections present the key interfaces and classes for accessing data via the MapInfo ADO.NET Data Provider.

- **MIConnection**
- **MICommand**
- **MIDataReader**
- **MapInfo SQL**

**MIConnection**

An MIConnection represents a connection to the Catalog. The connection provides a starting point for issuing SQL commands and obtaining results. Whereas most data provider connections allow the user to immediately begin issuing queries or other commands against existing tables (or schema objects), the MapInfo ADO.NET Data Provider initially has no tables available. Tables need to be opened or created before they can be accessed. When opened, a name (alias) can be associated with the table which is used when resolving identifiers in the query engine.

Connections are not pooled in the MapInfo Data Provider and there is no connection string required to create a new connection.

VB example:

```vbnet
Public Shared Sub MapInfo_Data_MIConnection()
    Dim connection As MIConnection = New MIConnection
    Dim command As MICommand = connection.CreateCommand()
    command.CommandText = "Select * From States Where Pop > 1000000"

    connection.Open()
    Dim reader As MIDataReader = command.ExecuteReader()
    Dim i As Integer, n As Integer = reader.FieldCount
    For i = 0 To n - 1 Step i + 1
        Console.Out.Write("{0}	", reader.GetName(i))
    Next
    Console.Out.WriteLine()
    While reader.Read()
        For i = 0 To n - 1 Step i + 1
            Dim o As Object = reader.GetValue(i)
            If o Is DBNull.Value Then
                Console.Write("null	")
            Else
                Console.Write("{0}	", o.ToString())
            End If
        Next
        Console.Out.WriteLine()
    End While
    reader.Close()
    command.Dispose()
    connection.Close()
End Sub
```

**MICommand**

MICommand provides the necessary interface for executing SQL commands against the MapInfo Data Provider. MICommand creates MIDataReader and MIScrollableReader instances for obtaining data via the ExecuteReader and ExecuteScrollableReader methods, respectively.

**Supported Commands**

The commands that are understood by the MICommand are:

**Select**

SELECT < select_list >
FROM { < table_source > } [ ,...n ]
[ WHERE < search_condition > ]
[ GROUP BY expression [ ,...n ] ]
[ ORDER BY {expression | column_position [ ASC | DESC ]} [ ,...n ]]
\[ \text{< select_list > ::=} \\
\text{\{} \text{* } \\
\text{\{} \text{ table_name | table_alias } \}.* \\
\text{\{} \text{ expression } \} \text{ [ [ AS ] column_alias ]} \\
\text{\} [ ,...n ]} \]

\[ \text{< table_source > ::=} \\
\text{ table_name [ [ AS ] table_alias ]} \]

**Insert**

\[ \text{INSERT [INTO] \{} \text{ table_name } \text{ [ [ column_list ] ]} \\
\text{ [ VALUES {expression | NULL}[, ...n] | query_specification} \]

**Update**

\[ \text{UPDATE \{} \text{ table_name } \text{ } \\
\text{ SET { column_name } = { expression | NULL } [, ...n] } \text{ WHERE < search_condition > } \]

**Delete**

\[ \text{DELETE [FROM] \{} \text{ table_name } \text{ WHERE < search_condition > } \]

\[ \text{< search_condition > ::=} \]
\[ \text{\{} \text{ NOT } \} \text{ < predicate > } \text{ | ( < search_condition > ) } \]
\[ \text{\{} \text{ AND | OR } \} \text{ NOT } \} \text{ < predicate > } \]
\[ \text{ ( < search_condition > ) } \text{ [ ,...n ]} \]

\[ \text{< predicate > ::=} \]
\[ \text{\{} \text{ expression [ = | < > | != | > | >= | < | <= } \text{ expression } \}
\text{ string_expression [ NOT ] LIKE string_expression [ ESCAPE \text{ escape_character } ]} \]
\[ \text{ expression [ NOT ] BETWEEN expression AND expression} \]
\[ \text{ expression IS [ NOT ] NULL} \]

**expression**

Is a column name, pseudo column, column alias, constant, function, or any combination of column names, column aliases, constants, and functions connected by an operator(s). Column names and pseudo columns may be prefixed with a table name or a table alias followed by the dot (".") character.

**group_by_expression**

Is a reference to a column in the select list - either an exact copy of the select list expression, the alias, a 1-based number indicating the position of the column, or \text{coln} where \text{n} is a number representing a column.

**order_by_expression**

Is a reference to a column in the select list - either an exact copy of the select list expression, the alias, a 1-based number indicating the position of the column, or \text{coln} where \text{n} is a number representing a column.
For more information on expressions, where they are used and how to create them, see Features and Feature Collections on page 191.

**ExecuteFeatureCollection**

The ExecuteFeatureCollection method in the MICommand class is the bridge between the MapInfo ADO.NET Data Provider and the Feature object model. This method executes command text (SQL statements) against the data source connection, and builds an IResultSetFeatureCollection. The Feature model is discussed in Features and Feature Collections on page 191.

**MIDataReader**

The MIDataReader provides forward-only, read-only access to the data returned from executing a SQL Select statement. To create a MIDataReader, you must call the ExecuteReader method of the MICommand object, rather than directly using a constructor.

The MapInfo Data Provider allows multiple MIDataReader instances to be in use on the same connection. However, if the Table being accessed resides on a Microsoft SQL Server database, only one MIDataReader may be open at a time.

IsClosed and RecordsAffected are the only properties that you can call after the MIDataReader is closed. Though the RecordsAffected property may be accessed at any time while the MIDataReader exists, always call Close before returning the value of RecordsAffected to ensure an accurate return value.

You must explicitly call the Close method when you are through using the MIDataReader.

When accessing the DataReader through the IEnumerator or IFeatureEnumerator interface, Close() is automatically called when MoveNext() returns false. Only one enumerator can be used on a DataReader.

**Note** For optimal performance, MIDataReader tries to avoid creating unnecessary objects or making unnecessary copies of data. As a result, multiple calls to methods such as GetValue may return a reference to the same object. Use caution if you are modifying the underlying value of the objects returned by methods such as GetValue.

The MIDataReader provides a means of reading a forward-only stream of rows from the MapInfo data provider. This cursor type is the best performing for accessing a selection of rows since there is little setup or overhead.

**Scrollable Data Readers**

MIScrollableReader derives from MIDataReader and offers forward and reverse reading. Some of the options available with MIScrollableReader include:

- ReadPrevious
- Rewind
- Unwind
- ReadTop
- ReadBottom
• AtTop / AtBottom

Note An MIScrollableReader is more expensive to create than MIDataReader. This is the most expensive cursor since there is setup and extra resources necessary to keep track of record order to allow scrolling. Use this cursor only if you need to scroll through the record set.

MapInfo SQL

The MapInfo SQL Language allows you to add powerful analytical processing to your MapXtreme application. MapXtreme exposes SQL processing to users via the MapInfo ADO.NET Data Provider for accessing data (specifically the MICommand object). Expressions are also used for labeling, thematics, legends, AddColumns, Feature searching, and Selection processing.

MapInfo SQL is standardized based on SQL-3 Specification. For example, you will find that:

• String constants are enclosed in single quotation marks
• Identifiers may be enclosed in double quotation marks
• Select has no relationship to the Selection

A complete reference including code examples for the MapInfo SQL language is provided in the MapInfo SQL Reference, which you can view directly from Visual Studio’s Help system.

Features and Feature Collections

The Feature class object model in MapXtreme offers a non-SQL-based approach to access and manipulate data. This section covers the Feature class and IFeatureCollection interface. A key task in working with features is the ability to search for them using a query definition object.

Feature

Features are described by their geometry, style, data source, key and attributes. Typically a feature is a row in a table. A feature’s geometry is a FeatureGeometry object. FeatureGeometries can cover a given area (MultiPolygon), a location (Points, MultiPoints); and distance (MultiCurves, LegacyArcs). Additional Geometry classes that derive from FeatureGeometry and are used for map features are FeatureGeometryCollection and LegacyText. (Rectangle, rounded rectangle and ellipse objects also derive from FeatureGeometry, but are used primarily for cosmetic display purposes.)
One of the main uses of computerized maps is to gather information about the features. In MapXtreme features are returned in FeatureCollections by any of several methods, either created from scratch using a schema, selected using selection tools or methods or by searching the Catalog for those that meet a specific set of criteria.

You can force a Load using the Load method. Changes made to the Feature are not reflected in the underlying table (if there is one) until the Feature is saved back to the table. This is done using the Update method, or UpdateFeature or InsertFeature. You can throw away any edits done to the Feature object before it is saved using the DiscardEdits method.

A Feature has a schema that describes the attributes of the Feature. The Columns property describes the schema.

## Retrieving Features from a Table

A Table is a type of Feature collection. As such, the Features within the table may be enumerated directly. For example:

**VB example:**

```vbnet
Dim ftr As Feature
For Each ftr In table
...
```

The default feature enumerator for a table uses an MIDataReader internally with the following command:

```sql
command.CommandText = "Select MI_Key, * From \\
" + table.Alias + "\\";
```

To retrieve a subset of the features in a table, use one of the Catalog.Search methods or use one of the MICommand.ExecuteFeatureCollection methods.

## Modifying Features in a Table

To modify features in a table, use one of the following methods.

- Feature.Update
- Table.UpdateFeature
- Table.InsertFeature

## Feature Collections

Feature collections are a group of Feature objects. All Features in a collection share the same Schema (columns). The Feature collection has a schema which is the schema of all of its member feature instances. Some Feature collections own their Features while other Feature collections maintain references to Features.
Searching for Features

One of the most common tasks in Pitney Bowes MapInfo’s mapping applications is to search for features that meet certain criteria. Once you have the features you are interested in, you can carry out further analysis, such as thematic mapping. In MapXtreme, searching for features can be done in a number of ways: using tools, using Catalog search methods, or using SQL and the MapInfo ADO.NET Data Provider.

The following code sample shows two ways to search for the same thing, in this case, cities in New York.

```
// Using SQL
command.CommandText = "Select Obj From States Where state = 'NY';
FeatureGeometry nyGeom = command.ExecuteScalar() as FeatureGeometry;
command.CommandText = "SELECT * FROM Cities WHERE Obj within @newyork";
command.Parameters.Add("@newyork", nyGeom);
MIDataReader reader = command.ExecuteReader();

// or... to get a FeatureCollection
IFeatureCollection fc = command.ExecuteFeatureCollection();
```

```
// Using Features
Feature fNY = catalog.SearchForFeature("States",
    SearchInfoFactory.SearchWhere("state='NY'"));
SearchInfo si = SearchInfoFactory.SearchWithinFeature(fNY,
    ContainsFilter.ContainsType.Centroid);
IDynamicFeatureCollection dfc =
    catalog.Search("Cities", si) as IDynamicFeatureCollection;
Console.Out.WriteLine(
    "There are {0} cities whose centroid is within NewYork."
    dfc.Count);
```

SQL searches are more fully discussed in MapInfo ADO.NET Data Provider on page 187. The following sections focus on searches using the Catalog and SearchInfo.

Catalog Search Methods

The Catalog has a number of search methods as members. The overloaded Search method can be used to search on one or more tables. They include different arguments to make each search unique. For example, the basic Search (Table, SearchInfo) searches the given table and returns a FeatureCollection. The Search (ITableEnumerator, SearchInfo) method searches on multiple tables and returns a MultiResultSetFeatureCollection.

The SearchForFeature method returns the first Feature from the results. The SearchReader method returns an MIDataReader cursor with the results.

Code Sample: SearchForFeature

The following example shows how to use Catalog.SearchForFeature and Catalog.SearchWithinGeometry. It finds all the cities in the uscty_1k table that are within Florida. It assumes that tables “usa” and “uscty_1k” are open and that there is one map.
VB example:

```vbnet
Public Shared Sub MapInfo_Data_SearchInfo(ByVal catalog As Catalog)
    Dim fFlorida As Feature = 
        catalog.SearchForFeature("usa", MapInfo.Data._
            SearchInfoFactory.SearchWhere_("State='FL'"))
    Dim si As SearchInfo =
        MapInfo.Data.SearchInfoFactory.SearchWithinGeomeTry(fFlorida._
            Geometry, ContainsType.Centroid)
    Dim fc As IResultSetFeatureCollection = 

    ' Set the map view to show search results
    ' Set the view of the first map.

    ' Add results to selection.
End Sub
```

**SearchInfo and SearchInfoFactory**

The MapInfo.Data.SearchInfo class defines the query used in a search and handles any necessary post processing of the search results.

The SearchInfoFactory creates SearchInfo objects. SearchInfoFactory contains a number of search methods that allow you to search using spatial references to your search location or by using geometries that are drawn on the screen.

The following table describes the SearchInfoFactory search methods.

<table>
<thead>
<tr>
<th>SearchInfoFactory Methods</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>SearchAll</td>
<td>Returns all the rows.</td>
</tr>
<tr>
<td>SearchNearest</td>
<td>Returns the rows with table geometries that are closest to the given search point.</td>
</tr>
<tr>
<td>SearchWhere</td>
<td>Returns the rows specified by the given where Clause.</td>
</tr>
<tr>
<td>SearchWithinDistance</td>
<td>Returns the rows where the table geometry is contained within a distance of the search point, rectangle or geometry. This method uses the Geometry.Distance method to determine if an object is in or out the search area. Previously SearchWithinDistance had buffered the distance and searched within the buffer, leading to less accurate results.</td>
</tr>
<tr>
<td>SearchWithinFeature</td>
<td>Returns the rows where the table geometry is contained within the search features's geometry.</td>
</tr>
</tbody>
</table>
MapXtreme ships with a Search sample application that you can run and learn more about each search type. The illustration below shows the Search menu with the SearchInfoFactory methods that use a spatial reference. The Map Search menu has methods for searching based on a drawn screen geometry object (circle or rectangle). The Query Definition menu highlights the use of various filters that act on an SQL statement. Find the sample in the ..\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features\Search folder.

<table>
<thead>
<tr>
<th>SearchInfoFactory Methods</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>SearchWithinGeometry</td>
<td>Returns the rows where the table geometry is contained within the search geometry.</td>
</tr>
<tr>
<td>SearchWithinRect</td>
<td>Returns the rows where the table geometry intersects the given rectangle.</td>
</tr>
<tr>
<td>SearchIntersectsFeature</td>
<td>Returns the rows where the table geometry intersects with the search features's geometry.</td>
</tr>
<tr>
<td>SearchIntersectsGeometry</td>
<td>Returns the rows where the table geometry intersects with the search geometry.</td>
</tr>
<tr>
<td>SearchWithinScreenRadius</td>
<td>Creates a SearchInfo that returns the rows where the table geometry intersects a screen circle.</td>
</tr>
<tr>
<td>SearchWithinScreenRect</td>
<td>Returns the rows where the table geometry intersects the given screen rectangle.</td>
</tr>
</tbody>
</table>
Code Samples

This section includes several code samples that pertain to SearchInfoFactory methods.

**SearchNearest**

Using the SearchNearest method, the code simulates the Select Point tool behavior to select the topmost items under a mouse click and add them to the selection.

VB example:

```vbnet
Public Shared Sub MapInfo_Mapping_SearchInfoFactory(ByVal mapControl1 As MapControl)
    ' Get a point from mouse click. Hard coded value use in sample.
    Dim pt As System.Drawing.Point = New System.Drawing.Point(100, 100)
    ' Assumes there is a MapControl with a map in it.
    Dim map As Map = mapControl1.Map
    Dim session As ISession = MapInfo.Engine.Session.Current
    Dim si As SearchInfo = MapInfo.Mapping.SearchInfoFactory.SearchNearest(map, pt, 3) ' 3 pixel tolerance radius
    si.QueryDefinition.Columns = Nothing
    ' fetch all columns instead of just default
    ' Customize to stop at topmost layer where something is found
    CType(si.SearchResultProcessor, ClosestSearchResultProcessor).Options = ClosestSearchOptions.StopAtFirstMatch
    ' Puts results of search directly into default selection
    ' Searches all tables in map in order from top to bottom.
        Layers.GetTableEnumerator(), si, _
        session.Selections.DefaultSelection, _
        ResultSetCombineMode.Replace)
End Sub
```

**SearchIntersectsFeature**

VB example:

```vbnet
Public Shared Sub MapInfo_Data_SearchInfoFactorySearchIntersectsGeomeTry(ByVal map As Map)
    Dim ti As Table = MapInfo.Engine.Session.Current.Catalog.GetTable("usa")
    Dim lParks As MapInfo.Mapping.FeatureLayer = CType(map.Layers("USA"), MapInfo.Mapping.FeatureLayer)
    Dim si As SearchInfo = MapInfo.Data.SearchInfoFactory.SearchIntersectsGeometry(g, MapInfo.Data.IntersectType.Geometry)
    Dim fc As IResultSetFeatureCollection = Session.Current.Catalog.Search("usa", si)
    map.SetView(fc.Envelope)
End Sub
```
**SearchWithinScreenRadius**

This sample illustrates how to search for features using a screen drawn circle.

```csharp
// find nearest city to center of map
private void menuItemSearchNearest_Click(object sender, System.EventArgs e)
{
    try
    {
        Cursor.Current = Cursors.WaitCursor;
        // to compare to SearchWithinScreenRadius, we are
        // calculating the search distance the same way it does
        System.Drawing.Rectangle rect=mapControl1.Bounds;
        System.Drawing.Point pt =
            new System.Drawing.Point(rect.Left, rect.Top);
        pt.X += rect.Width/2;
        pt.Y += rect.Height/2;

        DPoint dpt1 = new DPoint();
        // convert center point to map coords (could use map.Center)
        _map.DisplayTransform.FromDisplay(pt, out dpt1);
        Distance d =
            MapInfo.Mapping.SearchInfoFactory.ScreenToMapDistance
            (_map, 3);

        SearchInfo si =
            MapInfo.Data.SearchInfoFactory.SearchNearest
            (dpt1, _map.GetDisplayCoordSys(), d);
        IResultSetFeatureCollection fc =
            _catalog.Search("uscty_1k", si);

        MapInfo.Geometry.Point p =
            new MapInfo.Geometry.Point
            (_map.GetDisplayCoordSys(), dpt1);
        FeatureGeometry buffer =
            p.Buffer(d.Value, d.Unit, 20, DistanceType.Spherical);
        ShowSearchGeometry(buffer);
        SelectFeatureCollection(fc);
    }
    finally
    {
        Cursor.Current = Cursors.Default;
    }
}
```

**Code Example: How to Find a Feature Containing a Point**

You may also wish to perform the opposite type of search, in which you are looking for a feature that encompasses a point. For example, for delivery planning, you may want to know in which region a customer is located.

**VB example:**

```vbnet
' find nearest city to center of map
Private Sub menuItemSearchNearest_Click(sender As Object, e As EventArgs)
    Try
        Cursor.Current = Cursors.WaitCursor
        ' to compare to SearchWithinScreenRadius, we are
        ' calculating the search distance the same way it does
        Dim rect As System.Drawing.Rectangle = mapControl1.Bounds
        pt.X += rect.Width / 2
        pt.Y += rect.Height / 2

        Dim dpt1 As DPoint = New DPoint()
        ' convert center point to map coords (could use map.Center)
        _map.DisplayTransform.FromDisplay(pt, dpt1)
        Dim d As Distance = MapInfo.Mapping.SearchInfoFactory.ScreenToMapDistance(_map, 3)

        Dim si As SearchInfo = MapInfo.Data.SearchInfoFactory.SearchNearest(dpt1, _map.GetDisplayCoordSys(), d)
        Dim fc As IResultSetFeatureCollection = _catalog.Search("uscty_1k", si)

        Dim buffer As FeatureGeometry = p.Buffer(d.Value, d.Unit, 20, DistanceType.Spherical)
        ShowSearchGeometry(buffer)
        SelectFeatureCollection(fc)
    Finally
        Cursor.Current = Cursors.Default
    End Try
End Sub
```
Analyzing Data

Once your data is available in the Catalog, you will want to analyze it to meet your business objectives. The Catalog has an SQL processor that allows you parse and aggregate your data. Here you have two options:

- OGC object-based query interface
- ADO.NET SQL-based interface

The diagram below shows the relationship between the two.

Group 1 shows the OGC query interface. Use these objects to construct a query. The interface allows you to create queries to filter columns and rows, as well as add spatial and non-spatial conditions. The queries interact through the Search methods off those query objects to return data readers and result sets. Use these objects if you are more comfortable with object-oriented programming and less so with SQL syntax. See SearchInfo and SearchInfoFactory on page 194.

The ADO.NET interfaces, shown in group 2, use the defined ADO.NET model to allow access via the MapInfo SQL language. The ADO.NET interfaces use SQL syntax to interact with the Catalog. In this instance you need to generate the SQL statement and assign it to the MICommand object. These objects use the Execute command to return a data reader or result set. MapInfo ADO.NET Data Provider on page 187.
MapXtreme Data Model

Both the OGC query-based and ADO.NET command-based approaches use the Catalog (group 3) to organize the data sources as a response to the object or SQL query. The object-based query API will generate SQL and pass this to the Catalog for processing. In some instances you may be able to generate more efficient SQL by hand, but the objects are well defined and the interfaces restrict how you interact so the SQL tends to be optimal. If you are comfortable with the SQL language using the ADO.NET method may be more comfortable. But if you are inexperienced with SQL then the OGC object based query will work just as well.

The MapInfo SQL syntax is defined in the SQL Reference which ships with MapXtreme and is integrated into Visual Studio after installing MapXtreme. The language is based on SQL3 and has special MapInfo operators defined for spatial analysis. These operators begin with the MI_ prefix.

Data Readers, MemTables and Result Sets

The methods to access data return a data reader or result set. A data reader allows access in a sequential manner and does not store copies of data. It retrieves the data from the data source, except in the case where the data source is cached. Result sets are collections of keys. These keys allow you access back to the original tables and do not create copies of the data.

A MemTable also allows you to store data from various sources into one table. This table type stores data in a combination of memory arrays and temporary disk storage. When data is added, the MemTable makes a copy of the data and does not have a key or pointer back to the original table. These are useful for temporary layers for maps and containers for return values of processes such as a geocoding or routing result. MemTable access and map rendering performance is equivalent to native tables.
Result sets are a great tool when you need access to a defined set of rows and when you need to get data from the source. If the source data may change during your session then this method allows you to see the results if the data source supports concurrent access. Since MemTables are copies of data they are a static set of data rows and will not reflect changes from the original data sources.

**Improving Data Access Performance**

Performance is always an important aspect to any application that accesses data. Consider the following list in your design and development plans for your application.

- Only request the data you need (especially from an RDBMS). This limits the amount of data sent over the connection.
- For web applications, put attribute column information into the workspace file, so that all needed data will be pre-loaded at the MapInfo Session creation time. For an example, see **Overview of the Thematics Sample in Chapter 6 on page 108**.
- Only sort tables if you need an ordered list. This process takes time to read through the entire table to build an order. Also it will be slower if there is no index on the column.
- Only scroll if you need random access to a table. This also builds indexes to speed up access and remember order. Data readers access the data directly with no need to read extra data.
- Use consistent coordinate systems for Join and Search operations. This eliminates the need to convert geometries for every access.
- Use indexed columns for Join / Filter / Sort / Aggregate operations.
- Use CentroidWithin, ContainCentroid, and EnvelopesIntersect prior to actually checking for geometry intersects. These tests are very quick and in most cases eliminate a lot of geometries from your list with little effort.
- Use BeginAccess/EndAccess (especially for file-based tables) when performing multiple queries and/or edits.
- Try to avoid calls such as Area and Buffer in the Where clause because the operation will have to be done each time a new cursor is created.
- Try to avoid calls such as Area and Buffer in the Select list when defining a view or result set for similar reasons.
- Use result sets for intermediate results or operations where you manage keys. These are very light weight and afford quick direct access back to the original data.
Creating Expressions

Expressions are used throughout MapXtreme to describe the exact pieces of information you need to display and analyze in your mapping application. This chapter covers creating expressions for a wide range of product areas, including data access, creating themes, labeling maps and more.

In this chapter:

- Expressions Overview ........................................... 202
- Creating Expressions ............................................. 202
- Where Clause – Boolean Expressions ..................... 203
- Functions In Expressions ....................................... 203
- Expression Examples ............................................. 204
Expressions Overview

Expressions are statements that are used to describe and format data. For example, in English, an expression might read like “a median income of more than $50,000, or “female percent of population.”

Expressions are formed using column names, constants (i.e., specific data values), along with functions and operators that act upon the columns and constants. The operators and functions are defined in the MapInfo SQL Language, developed to support MapXtreme and other MapInfo .NET supported products going forward. For details, see the MapInfo SQL Reference online via the integrated Help in Visual Studio (look for MapInfo SQL Reference in the Dynamic Help contents pane).

Use expressions to make the most of your data. By using expressions you can:

- Show only the columns and rows of data that interest you.
- Derive new columns by calculating new values based on the contents of your existing columns.
- Aggregate data to work with subtotals instead of the entire table.
- Combine data from two or more tables into one results table.

Many of the data sets you will use include more objects and information than necessary for your projects. In many cases it is easier to work with a subset of the complete data product. For example, if you were tracking crime statistics for a certain county by census tract, you would not need the census tracts for the entire state. You would use an expression to extract just the census tracts for the county.

Expressions are used throughout MapXtreme, in the following areas:

- SQL statements (select, insert, update, delete, group by, order by)
- SQL functions that take expressions as an arguments (e.g., the geometry argument in MI_Area() is an expression that returns a geometry object.)
- Adding columns (MapInfo.Data.Table.AddColumn creates a temporary column based on an expression.)
- Feature searches (SearchInfo and SearchInfoFactory)
- Themes (FeatureStyleModifier)
- Labels (LabelModifier)
- InfoTips (FeatureLayer, MapTools)
- Expression dialog

Creating Expressions

The simplest possible expression consists of a constant, such as “2” (numeric example) or “Friday” (text example).

Other simple expressions consist of a column name, for example:

```
POP_2000
STATE
```
When you request specific multiple columns in a select statement, for example, these columns together are known as an expression list.

```sql
Select colA, colB, colC from Table1, Table2
Select colA/2, ColB/ColC from Table1
```

You can also write expressions that perform mathematical operations on your data.

For example, `RENT + UTILITIES` is an expression that adds two columns together. This expression could be used in a SQL statement to find all apartments that have a total cost of less than $800 per month.

**Where Clause – Boolean Expressions**

A Boolean expression is a search condition that results in a value of either True or False. For example, the expression

```
2 < 5
```

is a Boolean expression because the result is True.

All expressions that contain relational operators, such as the less than sign (<), are Boolean. The operators AND, OR, and NOT, are Boolean operators. Boolean expressions are also called comparison expressions, conditional expressions, and relational expressions.

```sql
POP_2000 > 500000
POP_2000 <= POP_1990
PROVINCE <> 'Ontario'
County = 'Columbia' AND VALUE >= 250000
```

Supported operators in MapInfo SQL are defined in the MapInfo SQL Reference online via the integrated Help in Visual Studio (look for MapInfo SQL Reference in the Dynamic Help contents pane).

Boolean expressions are used in the “where clause” of an SQL statement. The where clause is the expression that controls the rows that are returned (the rows that result in True).

For example, the boolean expression in this statement follows WHERE. Only objects in the Europe table that fall within the boundary of France will be returned as True.

```
"SELECT * FROM Europe WHERE MI_Geometry within @France";
```

**Functions In Expressions**

Functions in MapXtreme are used to create even more complex expressions to retrieve data that meets specific criteria. For example, MapInfo SQL supports many of the usual database functions that work with strings, dates, and numbers.

The most powerful functions in MapInfo SQL are those that take advantage of the spatial nature of mapping data. These geographic functions are used to create new geometries, measure area and length, return spatial information, validate spatial relationships among geometries, and others. Supported functions are defined in the MapInfo SQL reference.
Expression Examples

An example of using a function in an expression might be when you wish to look at the area of a table of boundaries, such as school districts. Use the function MI_Area() to return the area of each record in the table.

Additional examples of functions in expressions are found in the Expressions Examples section below.

Expression Examples

The following highlights some uses of expressions in various areas of MapXtreme.

SQL Statement Examples

This example will select all records from the Eurcity_1K table that are within Germany and have a population of over 1 million.

```
Select * from Eurcity_1K WHERE (MI_Geometry MI_Within @Germany) AND Tot_Pop > 1000000
```

The following examples make selections based on Time and Date columns in the table. This example will select all crime records from a "CrimeActivity" table where the crime occurred between 12:00:00 AM and 6:00:00 AM:

```
SELECT * FROM CrimeActivity WHERE CrimeTime BETWEEN '12:00:00 AM' AND '6:00:00 AM'
```

Where CrimeTime is a Time column that stores the time at which the crime occurred.

This example will select employee Names from an "Employee" table who were born before December 31, 1970.

```
SELECT Names FROM Employee WHERE BirthDay < '12/31/1970 12:00:00 AM'
```

Where BirthDay is a Date column that stores the birthdays of employees.

MapInfo SQL Function Example

The following expression uses a MapInfo SQL function to find features within a buffer.

```
Obj CentroidWithin MI_Buffer(Obj, 5, 'km', 'Spherical', 24)
```

This expression uses a MapInfo SQL special keyword reserved for geographic objects called ‘Obj’. This keyword describes the geometry of the object such as its coordinate system and bounds. This keyword is compatible with previous versions of MapX and MapInfo Professional. It is equivalent to the column name MI_Geometry.

Note that km and Spherical are enclosed in single quotes. In MapInfo SQL, string literals must be enclosed in single quotation marks while identifiers such as column names, table names, aliases, etc.) should be enclosed in double quotation marks, but only needed if the parsing logic is unable to correctly parse the identifier. This would include identifiers that have spaces in their names or other special characters.

To find features that fall outside the buffer, the expression would look like:
Chapter 11: Creating Expressions

Add Columns Example

When adding temporary (computed) columns to a table using the AddColumns method, the columns supplied contain an expression that defines how the value for the column is computed. The expression may contain an aggregation function if multiple source records are expected to match up to a single record in the table to which the columns are being added.

The example below uses expressions to represent population density \( \frac{\text{Pop}_1}{\text{MI_Area(Obj, 'sq mi', 'Spherical')}} \). The expressions are preceded by their new column names, PopDensity1990 and PopDensity2000.

VB example:

```vbnet
Public Shared Sub MapInfo_Data_TableAddColumns(ByVal miTable As Table)
    Dim NewCols As Columns = New Columns
        MI_Area(Obj, 'sq mi', 'Spherical')"))
        MI_Area(Obj, 'sq mi', 'Spherical')"))
    miTable.AddColumns(NewCols)
End Sub
```

For more information on adding columns, see Adding Expression Columns to a Table in Chapter 10 on page 175.

Feature Search Example

The following example uses a boolean expression SearchWhere("State='FL'") that, when executed, will return a value of 1 for each row that contains FL.

VB example:

```vbnet
Public Shared Sub MapInfo_Data_SearchInfo(ByVal catalog As Catalog)
    Dim fFlorida As Feature = catalog.SearchForFeature("usa", _
        MapInfo.Data.SearchInfoFactory.SearchWhere("State='FL'"))
    Dim si As SearchInfo = _
        MapInfo.Data.SearchInfoFactory.SearchWithinGeometry(fFlorida._
            Geometry, ContainsType.Centroid)
    Dim fc As IResultSetFeatureCollection = _

    ' Set the map view to show search results
    ' Set the view of the first map.

    ' Add results to selection.
End Sub
```

For more information on the Feature class and search methods, see Features and Feature Collections in Chapter 10 on page 191.
Expression Examples

**Thematic Expression Example**

A pie theme is an object theme that draws a pie chart based on the numeric value of the theme's expression. The expression for the theme is made up of three columns: `Pop_Native`, `Pop_Asian`, `Pop_Other`.

VB example:

```vb
Public Shared Sub MapInfo_Mapping_Thematics_PieTheme(ByVal _
    map As Map)
' Load a map based on one table
    map.Load(New MapTableLoader("world.tab"))
    Dim lyr As FeatureLayer = CType(map.Layers("world"), FeatureLayer)

' Create a new pie theme
    Dim pieTheme As MapInfo.Mapping.Thematics.PieTheme = New _
        MapInfo.Mapping.Thematics.PieTheme(map, lyr.Table, "Pop_Native", _
        "Pop_Asian", "Pop_Other")

' Create an object theme layer based on that pie theme
    Dim thmLayer As ObjectThemeLayer = New ObjectThemeLayer("World _
        Pop Growth Rate", Nothing, pieTheme)

' Add object theme to the map's layer collection.
    map.Layers.Add(thmLayer)
End Sub
```

For more information on themes, see **Using Themes and Legends in Chapter 15 on page 261**.

**Label Expression Example**

Expressions are properties of a label modifier which you use to modify the default properties used to make a label.

The highlighted line in the example below includes an expression to modifies the caption of the labels for the `usa_cap.tab`. The expression says to display the caption in this form: `Point: <name of capital>`.

VB example:

```vb
Public Shared Sub MapInfo_Mapping_OverrideLabelModifier(ByVal _
    modifier As OverrideLabelModifier)
    modifier.Name = "'Capital Name: ' + Capital"
End Sub
```

For more information on labeling, see **Labels in Chapter 13 on page 237**.

**InfoTips Expression Example**

InfoTips are text items that display when a tool hovers over a Feature. Expressions can be used to generate text for InfoTips in a similar fashion to the way they are used for labels. Use the `MapTool.SetInfoTipExpression` static helper function to set the InfoTip. It takes care of creating the InfoTip hashtable and adding the layer entry into the hashtable if it does not already exist. Each tool can be set to have InfoTips enabled or disabled.
The following example will generate a two-line text label to display the Table alias and either the X or Y coordinate for the centroid of the object that the cursor is hovering over.

VB example:

```vbnet
Public Shared Sub MapInfo_Mapping_HowDoICreateExprForInfoTip(ByVal mapControl1 As MapControl)
    MapTool.SetInfoTipExpression(mapControl1.Tools.MapToolProperties, _
        CType(mapControl1.Map.Layers(0), FeatureLayer), "@TableAlias + _
        char(13) + _ 'Centroid X:' + MI_CentroidX(obj) + '  Y:' + _
        MI_CentroidY(obj)"
    End Sub
```

For more information on InfoTips, see Layer Control in Chapter 23 on page 394.
MapXtreme provides Spatial Server Access. This is a powerful feature that allows developers to connect to live data stored in spatial servers, such as MapInfo’s SpatialWare running on Microsoft’s SQL Server, Informix, or the Oracle Spatial databases. Spatial servers allow companies to host their map data in their enterprise database for central management and security. Spatial servers like SpatialWare offer advanced query processing and increased performance on the server for an organization’s spatial data.

In this chapter:

- Accessing Remote Spatial Data ........................................ 210
- Accessing Remote Tables Through a .TAB File ................... 210
- Accessing Remote Tables Without a .TAB File .................... 210
- Mapping DBMS Data with X/Y Columns ............................ 211
- Accessing Data from Oracle ............................................. 211
- DBMS Connection String Format ..................................... 214
- Defining Mappable Tables in Server Table Queries .............. 216
- Accessing Attribute Data ............................................... 218
- Performance Issues ....................................................... 219
- Working with the Cache ................................................. 219
- The MapInfo MapCatalog .............................................. 221
- Adding Rows to the Map Catalog .................................... 223
- Per-Record Styles ....................................................... 225
- Troubleshooting ......................................................... 227
Accessing Remote Spatial Data

You can access data using MapXtreme with different DBMS servers. The servers include:

- IBM Informix Dynamic Server
- MS SQL Server 2000
- MS SQL Server 2005
- MapInfo SpatialWare for IBM Informix
- MapInfo SpatialWare for Microsoft SQL Server 2000
- MapInfo SpatialWare for Microsoft SQL Server 2005
- Oracle

**Note** See Supported Databases in Chapter 2 on page 29 for details on the specific versions that are supported.

You can add a table from data in a DBMS using the TableInfoServer class in the MapInfo.Data namespace.

The details for adding spatial data are included in the following sections. Once you have decided how to handle your data, follow the instructions in the appropriate section.

Accessing Remote Tables Through a .TAB File

A MapXtreme application can access DBMS data “live”, or can open a MapInfo Professional linked table. However, the linked table will be read-only and cannot be refreshed by your application. The data is actually from the remote database and does not reflect the data in the local linked version.

You can create a .TAB file to provide access to remote data. To generate a .TAB file using MapInfo Professional, choose File -> Open a DBMS table.

The .TAB file is a text file; you can create a .tab file using any text editor. Once you have created the .tab file, you can access it the way you access any other MapInfo .TAB file programmatically through the Catalog object or through the Workspace Manager.

Accessing Remote Tables Without a .TAB File

An application does not need a .tab file to access remote data. The following code sample illustrates this process.

**VB example:**

```vbnet
Public Shared Sub MapInfo_Data_TableInfoServer(ByVal connection As _
    MIConnection)
    ' Note: Do not specify any columns. These are determined
    ' dynamically from the query

    Dim ti As TableInfoServer = New TableInfoServer("Provinces")
    ti.ConnectString = "SRVR=ontario;UID=mapx;PWD=mapx"
    ti.Query = "Select * From Provinces"
```
ti.Toolkit = ServerToolkit.Oci

ti.CacheSettings.CacheType = CacheOption.Off ' On is the default
Dim tbl As Table = connection.Catalog.OpenTable(ti)
End Sub

Mapping DBMS Data with X/Y Columns

You can access data from a DBMS table that has X/Y coordinates. You need to create a Map
Catalog and register the tables as SpatialType 4.0 and specify two column names as the
coordinates. The columns should be indexed on the table. Connect to the DBMS via ODBC and
specify the new columns as “Obj” or “MI_Geometry” in your query.

Accessing Data from Oracle

To connect to an Oracle database from a MapXtreme application, the Oracle OCI connectivity client
must be installed and appropriate permissions granted. See your Oracle documentation for detailed
information.

Geometry Conversion

The table below shows the translation from MapXtreme objects to Oracle Spatial (SDO_Geometry).

<table>
<thead>
<tr>
<th>From MapInfo</th>
<th>To Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL geometry</td>
<td>NULL</td>
</tr>
<tr>
<td>Point</td>
<td>1 POINT</td>
</tr>
<tr>
<td>MultiCurve (with IsLegacyLine = true)</td>
<td>2 LINESTRING Geometry contains one line string</td>
</tr>
<tr>
<td>Polygon</td>
<td>3 POLYGON Geometry contains one polygon.</td>
</tr>
<tr>
<td>FeatureGeometryCollection</td>
<td>4 Collection Geometry is a heterogeneous collection of elements.</td>
</tr>
<tr>
<td>MultiPoint</td>
<td>5 MULTIPOINT</td>
</tr>
<tr>
<td>MultiCurve</td>
<td>6 MULTILINESTRING Geometry has multiple line strings.</td>
</tr>
<tr>
<td>MultiPolygon</td>
<td>7 MULTIPOLYGON Geometry has multiple polygons.</td>
</tr>
<tr>
<td>Ellipse</td>
<td>NULL</td>
</tr>
<tr>
<td>LegacyArc</td>
<td>NULL</td>
</tr>
</tbody>
</table>
The table below describes the translation from Oracle (GTYPES) to MapInfo Spatial objects.

<table>
<thead>
<tr>
<th>From Oracle GTYPES</th>
<th>To MapInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  UNKNOWN_GEOMETRY (Spatial ignores this geometry.)</td>
<td></td>
</tr>
<tr>
<td>1  POINT Geometry contains one point.</td>
<td>Point</td>
</tr>
<tr>
<td>2  LINestring Geometry contains one line string.</td>
<td>MultiCurve</td>
</tr>
<tr>
<td>3  POLYGON Geometry contains one polygon.</td>
<td>MultiPolygon</td>
</tr>
<tr>
<td>4  Collection Geometry is a heterogeneous collection of elements.</td>
<td>FeatureGeometryCollection</td>
</tr>
<tr>
<td>5  MULTIPOINT Geometry has multiple points.</td>
<td>MultiPoint</td>
</tr>
<tr>
<td>6  MULTILINestring Geometry has multiple line strings.</td>
<td>MultiCurve</td>
</tr>
<tr>
<td>7  MULTIPOLYGON Geometry has multiple polygons (more than one exterior boundary).</td>
<td>MultiPolygon</td>
</tr>
</tbody>
</table>

**Multi Dimension Geometry Support (Greater than two Dimensions)**

MapXtreme applications can read these data and map them as layers for SDO_GEOMETRY columns containing geometries defined in more than two dimensions. However, only the first two dimensions are used as the x and y.

**SDO_GEOMETRY Arc and Circle Translation**

Circles and circular arcs can be resolved to MultiCurves with a resolution of 25 segments per 360 degree circle.

**Visualization of Non-translatable Oracle Objects**

Oracle Spatial Objects that your MapXtreme application is unable to translate produces a Point object with a default style (a black star) at the location of the SDO_Spatial point, or the first SDO_Spatial ordinate in the ordinate array. This is to enable a visual representation of the non-translatable object in the proper geographic area to which it belongs. Examples of non-translatable
objects are user-defined objects GTypes 0, 4, 5, or invalid sdo_geometries containing unrecognized
GTypes, ETypes, or interpretations. The second class should also fail using
sdo_validate_geometry().

Centroid Support

A MapXtreme application uses the SDO_POINT as the centroid value for polygons. This centroid
feature is used to position labels, and also affects the tool selection of the object. The Oracle
SDO_GEOMETRY.SDO_POINT_TYPE field (if not NULL) is interpreted as the feature centroid if the
point exists inside the region. If the point exists outside of the region, its centroid is calculated as
always.

Note  There is currently no method or tool in MapXtreme to set the centroid of a region feature, but
one may read and use a stored centroid.

Additional Information

The following sdo_geometry values are not always translatable to or from MapInfo objects format
without data loss. Some geometry details could be lost if updated.

- GTYPE 0 – User defined/unknown geometry
- GTYPE 4 – heterogeneous collection of elements

These types are interpreted (when possible) as single point SDO_POINTTYPE values if not already
NULL. They “grab” the first point in the ordered array which would be interpreted as a NULL
geometry.

Note  Any other SDO_GEOMETRY values that contain circles or arc components are translated
into line strings containing 25 lines per 360 degrees.

Oracle Spatial Reference Support (SRID)

An Oracle SDO_GEOMETRY column may be defined with a spatial referencing system. This is
done by providing the Oracle SRID in the USER_SDO_GEOM_METADATA and also by assigning
that SRID in the stored SDO_GEOMETRY values. If a table contains an Oracle Spatial column with
an assigned SRID, your MapXtreme application is able to query and properly interpret the data. The
MapInfo_MapCatalog must contain the same MapInfo Professional CoordSys string as indicated in
the SRID of the data, since it is the Coordsys in the MapInfo_MapCatalog that is currently used to
interpret and update the data.

If the Spatial column does not contain an SRID value, (the value is NULL), your MapXtreme
application is also able to interpret the data via the MapInfo Professional Coordsys defined in the
MapCatalog.

When loading tables that use the Latitude/Longitude coordinate system (Geodetic Data) to Oracle
Spatial, it is important to verify that all geometry coordinates are between (-180, 180) longitude and
(-90, 90) latitude. Geodetic data coordinates beyond that range are not supported in Oracle Spatial
and may cause problems. You can check your data using MapInfo Professional before loading, or
use the Oracle Spatial SDO_GEOM.VALIDATE_LAYER() function on the table after loading it to
Oracle Spatial.
OCI Connection Dialog

The MapInfo.Data.DBMSConnectionCollection class supports a ConnectionFailed event by subscribing to the ConnectionFailedEvent event handler. When fired, this event displays an OCILoginDlg to give the user an opportunity to change the login information and retry the connection to the Oracle database one additional time. This handler is specific to the MapXtreme OCI toolkit. The event is also available through the LoadMapWizard class in the MapInfo.Windows.Dialogs namespace.

DBMS Connection String Format

ODBC Connection String Format

The format of the ODBC connection string is defined by several clauses separated by semicolons (;). Each clause has the form Key=Value. Important keys are listed below.
### Chapter 12: Accessing Data from a DBMS

#### Note
Connection attributes/parameters do not have to be in order and one may use a dialog to get a connection from an existing connection pool to avoid redundant connections. In previous version of our API, if you used a dialog each time to connect to the same database, or if you did not order the connection keywords in the connection string in the documented order, the connection would not be shared and you would get multiple connections.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
</table>
| **DLG=** | A number that controls the display of the connection dialog box:  
0 – Suppresses the connection dialog.  
1 – Displays the connection dialog.  
2 – Displays the connection dialog, but only if needed (i.e., if not all required information was provided) [default]. |

**Note** ASP.NET Applications which use Pooling must have a DLG=0 clause in the connection string for ODBC.

| **DSN=** | Specifies the ODBC data source name. |

**Caution:** If you use the DSN= syntax key, the name that you specify must match the data source name in use on the user’s system. Note that different users might use different names to refer to the same data source. If you cannot know in advance what data source name to use, use the DRIVER= syntax key instead of the DSN= syntax key.

| **DRIVER=** | Specifies the exact driver name of the installed driver. Used instead of the DSN= syntax key. |

**Example:**

```
DRIVER={SQL Server}
```

**Note** Informix 2.80.0861 does not support DRIVER=. Also, it does not apply to Oracle Spatial.

| **UID=** | Specifies the desired UserId for the data source, if required. |
| **PWD=** | Specifies the user’s password for the data source, if required. Passwords do not need to be in the connection string for the two strings to match. If two tables are in the same database, the connection string is the same. |

**Note** Connection attributes/parameters do not have to be in order and one may use a dialog to get a connection from an existing connection pool to avoid redundant connections. In previous version of our API, if you used a dialog each time to connect to the same database, or if you did not order the connection keywords in the connection string in the documented order, the connection would not be shared and you would get multiple connections.

### ODBC Layers and Pooling in Web Applications

When adding remote layers via ODBC to an ASP.NET application that uses pooling, be sure to have the DLG=0 clause in the connection string. This will avoid the display of unnecessary user and password dialogs that can time out. This applies to TAB files and workspaces. The following connection string example shows the highlighted DLG=0 clause.

```
<ConnectString>DRIVER={SQL Server};DATABASE=Devel;Server=Paladin;UID=devel;PWD=devel;QuotedID=Yes;Trusted_Connection=No;Network=DBMSSOCN;Address=PALADIN,1433;DLG=0</ConnectString>
```
When using TableInfoServer and pooling is on, to access SQL Server specify "DLG=SQL_DRIVER_NOPROMPT" in the connection string to avoid throwing a MapInfo.Data.TableException: Unable to open table.

### Oracle Spatial Connection String Format

These are the Oracle Spatial keywords. The string is defined by several clauses separated by semicolons (;). Each clause has the form Key=Value. Important keys are listed in the table below.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRVR=</td>
<td>Reflects the service name for the server set in the Oracle Net8 EasyConfig utility. This is required for Oracle connectivity, but does not apply to ODBC connections.</td>
</tr>
<tr>
<td>UID=</td>
<td>Specifies the desired UserId for the data source, if required.</td>
</tr>
<tr>
<td>PWD=</td>
<td>Specifies the user’s password for the data source, if required.</td>
</tr>
</tbody>
</table>

### Sample Connection Strings

Here are sample connection strings for the Oracle Spatial, IUS/UDO, and SpatialWare ODBC drivers.

**Oracle Spatial connection string:**

```
UID=george;PWD=password;SRVR=OracleSpatial9i
```

**IUS/UDO connection string:**

```
DRIVER={INFORMIX 3.81 32 BIT};UID=informix;PWD=secret;DATABASE=sw;
HOST=HostName;SERVER=ServerName;SERVICE=sqlexec;PROTOCOL=onsoctcp;
```

**SpatialWare for SQL Server string:**

```
DRIVER={SQL SERVER};
SERVER=ServerName;UID=Troll;PWD=secret;Database=GEORGETOWN
```

### Defining Mappable Tables in Server Table Queries

The query you specify for a server table defines the result set of data from your DBMS that represents the data in the table being added. You can formulate a fairly complex query to do powerful server-side analysis that defines a mappable table in MapXtreme. Your MapXtreme application uses this query internally to access the data.

MapXtreme generates several internal queries based on your query to access the data in a map as well as selection/key based queries. The table from which the geometry column is selected must be registered in the MapInfo MapCatalog on the server. MapXtreme requires this to obtain certain metadata about the geometry column such as the coordinate system, spatial type, and default styles.
In order for a query to define a mappable table, the query must contain both a geometry column and a key column. Sometimes for more complex queries on small sets of data (where the spatial indexing or spatial predicate cause the query to fail), you can specify `TableInfoServer.MbrSearch=false` to enable the results to be mapped.

The Geometry Column

If you do not specify a geometry column that your MapXtreme application can recognize, the table is opened, but cannot be added to a map (the table is unmappable). MapXtreme determines the geometry column of the table by looking it up in the MapCatalog and by examining the result set datatype of the column. You can reference the geometry column generically via the pseudo column name “Obj”, or you may refer to the geometry column using its specific column name. This form is required to reference the geometry column for an X/Y mappable layer. You can specify a geometry column via any server-supported geometry function/expression.

Example

```
Select Obj from rdbdata
Select sw_geometry from rdbdata
select sw_member, ST_Buffer(geometry, 66.0, 0.1) from rdbdata
  // a geometry function
Select st_geometry(st_point(72.5, 42.5) from rdbdata
  // a geometry constructor
```

Spatialware function examples:

```
select sw_member, ST_Buffer(sw_geometry, 66.0, 0.1) from rdbdata
select ST_Overlap(flood100.sw_geometry, lake.sw_geometry) from flood100,
  lake where ST_Overlaps(flood100.sw_geometry, lake.sw_geometry)
```

Oracle sdo_buffer example:

```
Select mi_prinx, mdsys.sdo_geom.sdo_buffer(geoloc, (select diminfo from
  sdo_geom_metadata where table_name = 'ALINE'), 20) from aline where prinx = 1
```

Oracle constructor example:

```
Select 1 "mi_prinx",
  mdsys.sdo_geometry(3,null,null,mdsys.sdo_elem_info_array(1,3,3),
  mdsys.sdo_ordinate_array(-79.919909,40.553465,-71.060457,45.363657)) from
dual
```

The Key Column(s)

A key column(s) must be returned in the query to enable it to be opened as a table. This is what enables your MapXtreme application to identify each row in the result set to perform shading, selection, and label operations on the layer.

The key column does not need to be specified in the query in most cases.
Your MapXtreme application can look up and determine the best key column(s) to use in order to uniquely reference a row in the result set, and then add them to the query if they are not present. In most cases, this is the primary key/unique index.

For Oracle Spatial tables, the MI_PRINX may be used.

For some queries, it is not possible for your MapXtreme application to identify the key. This is the case in a query on a view or a synonym. The view or synonym must appear in the MapInfo MapCatalog. They also must be registered as required with the underlying Spatial index system in most cases. Since MapXtreme cannot determine the key on these, a mechanism is provided to allow the application developer/query writer to identify the key column in the result set. The key must be a single column and must be a distinct value in the result set. To identify the column that is to be used as the key column, you can specify column alias of prinx or mi_prinx, (e.g., select custid mi_prinx, custname, Obj from mycust).

Example
Select customer_id mi_prinx, obj from customer_view

The column alias "mi_prinx" is used to identify and use the customer_id column as the key column for the table. You can alternately alias the desired key column in the create view statement to identify the key column automatically for any query on that view.

Example
Create view customer_view as select customer_id mi_prinx, geoloc from customer

In general, if a column name or column alias of prinx, or mi_prinx is found in the result set, that column is used as the key column for the table. This enables the application/query writer to specify the key column they desire.

Accessing Attribute Data

To use all available data columns, specify a query such as Select * From tablename. You are not required to specify * (asterisk); instead, you can designate specifically which columns you want to use. For the best performance, limit your query so that it retrieves only the needed columns.

When you add a DBMS table, for performance sake, you should only specify the columns in the query that you intend to use in your application. These are the spatial column, the key column(s), which are added automatically if you do not specify them, and columns you want to label with, or create a theme from. You may use the pseudo columns “OBJ” for any mappable table to refer to the column(s) containing the spatial data. This is required for a table using the MapMarker MDIGEOADDRESS column on a table with an X/Y column.

You can use any server side expression/function to specify a column. Also, avoid select * from tab in a real application.

The following code example defines a server table using a TableInfoServer and adds a layer to a map using this definition. You can now label or place themes based upon the columns in this table.

VB example:
Public Shared Sub MapInfo_Data_TableInfoServer(ByVal connection As MIConnection)
' Note: Do not specify any columns. These are determined 
' dynamically from the query

Dim ti As TableInfoServer = New TableInfoServer("Provinces")
ti.ConnectionString = "SRVR=ontario;UID=mapx;PWD=mapx"
ti.Query = "Select * From Provinces"
ti.Toolkit = ServerToolkit.Oci

ti.CacheSettings.CacheType = CacheOption.Off ' On is the default
Dim tbl As Table = connection.Catalog.OpenTable(ti)
End Sub

Performance Issues

Establishing a connection with the database server may take several seconds. This is a one-time cost, incurred when the table is first opened.

The map-display speed depends on how much data is retrieved from the server. In some cases, displaying a map from a server is noticeably slower than displaying a map from a local file. Speed also depends on whether your MapXtreme application has already cached the map features that are being displayed.

Working with the Cache

Knowing how to work with Cache in MapXtreme enables you to improve your application’s performance. The sections below describe what the Cache is, how it works in the MapXtreme object model, and the CacheSettings property of the TableInfoServer object.

What Is the Cache?

In place of local files, applications can access MapXtreme features from a remote database. In place of reading these records from the database each time the map needs to be acted upon, your MapXtreme application can temporarily store these records in the cache. This limits the number of calls between the application and the remote database. Records in a server table can be cached to improve the performance of your application (e.g., drawing, thematics, labeling, etc.). Server table data is cached internally as it is read and drawn to the Map window. All subsequent redraws read from this cache instead of going to the server database for the same data. The cache is able to offer significant redraw performance improvement.

There are several settings that developers can use to customize cache usage. The cache can be enabled (or disabled) when the server table is added by specifying the values for the CacheSettings property of the TableInfoServer object and is On by default.
How the Cache Works

For each record that is cached, each attribute data value is stored in memory and each feature object is stored on disk in a temporary Rtree file. For tables with a lot of records and/or a large record size (for example, number of bytes per record for the attribute data), caching may use a significant amount of memory. If an application tries to cache too much data, too much virtual memory usage may be required, which can degrade performance. Applications should be selective about how the cache is utilized. MapXtreme offers a variety of mechanisms for controlling the cache.

The TableInfoServer Object and the CacheSettings Property

When a table is added to the map, the cache is enabled by default but can be further configured using the CacheSettings property of the TableInfo object. This property has four possible values: ON, OFF, ALL, USER, with ON being the default for TableInfoServer, OFF is default for other TableInfo objects.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>A value of 'Off' means that the table will not use the cache at all. All data operations will go directly to the database server.</td>
</tr>
<tr>
<td>ON</td>
<td>Caching is enabled and the table automatically performs caching based on the map view (center/zoom). The user may additionally control the cache through the cache constraint objects. The cached table maintains the record cache in a fashion that best improves standard map operations. The cache is maintained to contain, at a minimum, all the records displayed in the current window of each Map the table is in (and visible). Once an initial map window has been cached, pan and zoom operations that fall entirely within the initial extents of the cache access the cached records and do not need to query the database. If a pan/zoom operation falls outside the cached region, the table adds the new map window MBR (view) to the cache and obtains the missing records from the database server and adds them to the cache. The old map view is not initially discarded; rather, an internal history of previous map views is maintained. To avoid having a cache that grows excessively large, there are controls that can be placed on the table's cache to determine when to discard old cached views (map window MBR regions). These controls are properties of the CacheParameters object, which can be set at the time the table is initially opened. This allows the developer to set limits on the maximum amount of memory or disk space used by the cache, the maximum number of previous map window views to maintain in the history, the maximum number of records to maintain in the cache, and/or the maximum amount of time old map window views are allowed to remain in the cache history. These limits can be used individually or in combination to provide the cache management that best suits the application's needs. ON is the default setting for the CacheParameter setting for a TableInfoServer. For other TableInfo data sources, the default is OFF. For example, TAB files are not cached by default.</td>
</tr>
</tbody>
</table>
Chapter 12: Accessing Data from a DBMS

If you try to cache too much data or too many tables, virtual memory usage may be forced, and performance gain could be lost.

The MapInfo MapCatalog

In order to display data on a map, your MapXtreme application needs to access a special table, known as the MapInfo Map Catalog. One catalog must be created per database before any tables in that database can be viewed as a map layer in a MapXtreme application. The Map Catalog must contain information about the spatial columns in each of the mappable tables you want to access from the database. The MapInfo EasyLoader utility automatically inserts the appropriate row into the MapInfo Map Catalog when the table is uploaded into the database.

Your application can use a Map Catalog that already exists on the server. (This same catalog is shared by various MapInfo client applications). If there is no Map Catalog on the server, you need to create one. MapXtreme supports the storage of style information for individual features in remote databases.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td>A value of USER for the LayerInfo CACHE parameter means that your application creates a cache, but the only records that are placed in the cache are those specified by the application developer. The mechanisms available for specifying which records are placed in the cache are BoundConstraint, FeaturesConstraint, and AllFeaturesConstraint objects. The word constraint implies that these objects are constraining the cache to include the specified records. The BoundsConstraint object can be used to place all records into the cache for which the MBR of the feature intersects the MBR of the constraint. A FeaturesConstraint object can be used to add specific records to the cache. For example, if an analysis is going to be performed that involves multiple steps and/or reads of the Feature or RowValues of the feature, possibly on a set of features returned from a Layer.Search, Layer.SearchWithinDistance, etc., it may be advantageous to place these records into the local cache for the duration of the analysis and remove them when finished. The FeaturesConstraint provides this capability. If an application is going to perform an analytically intensive operation that may hit every record, it may be desirable to temporarily cache the entire set of data for the layer. This is accomplished by using the AllFeaturesConstraint. These cache constraint objects can also be used when the cache is set to ON. In this case, they may add records to the cache but have no effect on the cache’s history of previous map window views. The constraint objects can also be used when the cache is set to OFF or ALL in which case they have no effect. Note The constraint objects have no effect on non-server tables.</td>
</tr>
<tr>
<td>ALL</td>
<td>The entire table is cached. With this option, the table's data is retrieved from the server once and accessed locally from that point forward. To refresh the data in the cache, use the Refresh method on the table. If you try to cache too much data or too many tables, virtual memory usage may be forced, and performance gain could be lost.</td>
</tr>
</tbody>
</table>

Parameter Description
Loading Spatial Data to DBMS

If you have spatial data in the form of a MapInfo table, you can import that data into your DBMS database.

To load data into SQL Server SpatialWare, Informix IDS SpatialWare, and Oracle Spatial, use the MapInfo EasyLoader, that is distributed with MapInfo Professional and available for download from http://www.mapinfo.com. The EasyLoader utility automatically creates a map catalog when you upload a table. If there is no map catalog already present.

Manually Creating a MapInfo MapCatalog

If you are not a MapInfo Professional or EasyLoader user, you will need to create the MapCatalog (or have your database administrator create the MapCatalog) manually, as described below. You only have to create the MapCatalog once per server/database.

1. Create the user MAPINFO in the specific database where the mappable tables are located.
2. Create the table MAPINFO_MAPCATALOG in the database. The Create Table statement needs to be equivalent to the following SQL Create Table statement:

```sql
Create Table MAPINFO_MAPCATALOG (  
SPATIALTYPE Float,  
TABLENAME Char(32),  
OWNERNAME Char(32),  
SPATIALCOLUMN Char(32),  
DB_X_LL Float,  
DB_Y_LL Float,  
DB_X_UR Float,  
DB_Y_UR Float,  
COORDINATESYSTEM Char(254),  
SYMBOL Char(254),  
XCOLUMNNAME Char(32),  
YCOLUMNNAME Char(32),  
RENDITIONTYPE INTEGER,  
RENDITIONCOLUMN CHAR(32),  
RENDITIONTABLE CHAR(32)  
NUMBER_ROWS INTEGER )
```

**Note** It is important that the structure of the table looks exactly like this statement. The only substitution that can be made is for databases that support varchar or text data types; these data types can be substituted for the Char data type.

3. Create a unique index on the TABLENAME and the OWNERNAME, so only one table for each owner can be made mappable.

```sql
create unique index mapcat_i1 on mapinfo.mapinfo_catalog (OwnerName,TableName)
```

4. Grant Select, Update, Insert, and Delete privileges on the MAPINFO_MAPCATALOG. This allows users to make tables mappable.

```sql
grant select, insert, update, delete on mapinfo.mapinfo_mapcatalog to public
```
Adding Rows to the Map Catalog

For each spatial table that you want to access from your application, you need to add a row to the MAPINFO_MAPCATALOG table. If you do not use MapInfo Professional to manage the Map Catalog, you will have to add rows to the MAPINFO_MAPCATALOG table manually. The following table describes the syntax and meaning of each column:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Values to Assign</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPATIALTYPE</td>
<td><strong>MapInfo Spatial Object Format</strong>&lt;br&gt;1: Point layer in X/Y columns indexed&lt;br&gt;with micode (a serialized quadtree key)&lt;br&gt;4: Point layer in X/Y columns&lt;br&gt;5: x: SpatialWare for Oracle&lt;br&gt;6: x: Ingres SOL - Not Supported&lt;br&gt;7: x: Sybase SQS - Not Supported&lt;br&gt;8: x: Oracle SDO version 2 - Not Supported&lt;br&gt;9: x: MapInfo Geocoding DataBlade&lt;br&gt;  SpatialWare Point Module&lt;br&gt;10: x: MapInfo Geocoding DataBlade XY&lt;br&gt;Module&lt;br&gt;11: x: SpatialWare IDS/UDO datablade&lt;br&gt;13: x: Oracle Spatial&lt;br&gt;14: x: SpatialWare for Microsoft SQL Server</td>
<td>14.0 = SQL Server&lt;br&gt;14.1&lt;br&gt;14.2&lt;br&gt;14.3</td>
</tr>
<tr>
<td>TABLENAME</td>
<td>The name of the table.</td>
<td></td>
</tr>
<tr>
<td>OWNERNAME</td>
<td>The owner name of the table.</td>
<td></td>
</tr>
</tbody>
</table>

**Spatial Object Type**

x.0: Points only  
x.1: Lines only  
x.2: Regions only  
x.3: All types supported

**Note**  
This column describes the Spatial Object Format of how the data is stored and indexed and the Spatial Object type(s) supported and not supported in the column. The digits to the left of the decimal point are the Spatial Object Format. The digits to the right represent the type of Spatial Object Type that can be stored in the column.
### Adding Rows to the Map Catalog

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Values to Assign</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPATIALCOLUMN</td>
<td>The name of the column, if any, containing spatial features:</td>
<td>SW_GEOMETRY</td>
</tr>
<tr>
<td></td>
<td>SW_GEOMETRY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(mappable using SpatialWare Type/IDS/UDO)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NO_COLUMN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MI_SQL_MICODE (mappable using MI Code) Or the name of the IDS/UDO, or Oracle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>column that is ST_SPATIAL datatype. Name of the Oracle SDO_GEOMETRY column.</td>
<td></td>
</tr>
<tr>
<td>DB_X_LL</td>
<td>The X coordinate of the lower left corner of the layer’s bounding rectangle, in</td>
<td>-360</td>
</tr>
<tr>
<td></td>
<td>units that are indicated by the COORDINATESYSTEM (see below).</td>
<td></td>
</tr>
<tr>
<td>DB_Y_LL</td>
<td>The lower left bounding Y value.</td>
<td>-90</td>
</tr>
<tr>
<td>DB_X_UR</td>
<td>The upper right bounding X value.</td>
<td>360</td>
</tr>
<tr>
<td>DB_Y_UR</td>
<td>The upper right bounding Y value.</td>
<td>90</td>
</tr>
<tr>
<td>COORDINATESYSTEM</td>
<td>A string representing a MapInfo CoordSys clause (but without the keyword</td>
<td>Earth Projection 1, 0</td>
</tr>
<tr>
<td></td>
<td>CoordSys at the very start), which specifies a map projection, coordinate</td>
<td></td>
</tr>
<tr>
<td></td>
<td>units, etc. For simple Lon/Lat maps, specify Earth Projection 1, 0.</td>
<td></td>
</tr>
<tr>
<td>SYMBOL</td>
<td>A MapInfo Symbol clause (if the layer contains only points); or a Symbol</td>
<td>Symbol(35,0,12) Pen(1,2,0) Pen(1,2,0) Brush(2,255,255)</td>
</tr>
<tr>
<td></td>
<td>clause followed by a Pen clause (indicating styles for linear features)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>followed by another Pen clause (indicating styles for the borders of regions)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>followed by a Brush clause.</td>
<td></td>
</tr>
<tr>
<td>XCOLUMNNAME</td>
<td>For the X/Y mappable tables, specify the name of the column containing X-</td>
<td>NO_COLUMN</td>
</tr>
<tr>
<td></td>
<td>coordinates. If there is no such column (i.e., if this table uses a single</td>
<td></td>
</tr>
<tr>
<td></td>
<td>spatial column instead of a pair of X-Y columns) then specify NO_COLUMN or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>leave empty.</td>
<td></td>
</tr>
<tr>
<td>YCOLUMNNAME</td>
<td>For the X/Y mappable tables, specify the name of the column containing Y-</td>
<td>NO_COLUMN</td>
</tr>
<tr>
<td></td>
<td>coordinates, or specify NO_COLUMN</td>
<td></td>
</tr>
</tbody>
</table>
### Per-Record Styles

Per-record style support brings a feature to spatial database implementations that has long been available in MapInfo TAB files. Specifically, it allows each geometry in a single table to have its own style. For example, a single 'public institution' table in Oracle Spatial can have schools, town halls, libraries, and police departments and each point type would be represented with its own symbol (i.e.,

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Values to Assign</th>
<th>Examples</th>
</tr>
</thead>
</table>
| RENDITIONTYPE     | This indicates how the object style information is applied.  

0 – Indicates that all the objects in the table will have the style specified in the symbol field of the MapCatalog applied to them. No per-record styles are in effect. Objects will be read/updated using the default style for the table.  

1 – Indicates that the table uses per-record styles. The table has a separate column that contains a MapBasic string representation of the style information for each object in the table (the same format that is currently used in the MapCatalog's SYMBOL column). The style column in the table is recorded in RENDITIONCOLUMN.                                                                                                                                 | 0 or 1    |
| RENDITIONCOLUMN   | If RENDITIONTYPE is 1, this field stores the name of the column in the spatial table that contains style information. This column is automatically added to any query against the table and is maintained (updated) as the object is updated. Users should NOT specify this column in their queries as problems can occur with intersect or update statements. Queries which include this column in the select clause (excluding the wildcard character "*") may access the values through the Dataset object. Rows with a NULL value in their style column will have the style from the SYMBOL field of the MapCatalog applied to the object. | MI_SYMBOLOGY |
| RENDITIONTABLE    | Currently not used, but reserved for future use, this column must exist in order for MapXtreme to correctly identify and apply record-level styles.                                                                                                                                                                                                                                                                     | NULL     |
Per-Record Styles

a school symbol for all the schools). Similarly, a single road table in SpatialWare SQL Server may have different road types such that streets are shown as a single pixel black line, secondary roads as a double pixel red line and interstates as parallel red lines.

To use per-record styles, your table must be represented with an entry in the MapCatalog with appropriate settings for RENDITIONTYPE, RENDITIONCOLUMN, and RENDITIONTABLE.

Note If these columns are not present, the table’s default style will be applied to all objects.

Symbol, Pen, Brush Clause Syntax

If you are manually creating a MAPINFO_MAPCATALOG table to provide support for a remote spatial database, you will need to specify a symbol style, and possibly line and fill styles as well.

Specifying Point Styles

Use a Symbol clause to specify point styles. There are three types of Symbol clauses: one for specifying MapInfo 3.0-style symbols; one for specifying TrueType font symbols; and one for specifying bitmap symbols.

<table>
<thead>
<tr>
<th>Symbol Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol(shape, color, size)</td>
<td>Symbol(35, 0, 12)</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>Symbol(shape, color, size, font, fontstyle, rotation)</td>
<td>Symbol(64, 255, 12, &quot;MapInfo Weather&quot;, 17, 0)</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>Symbol(bitmapname, color, size, customstyle)</td>
<td>Symbol(&quot;sign.bmp&quot;, 255, 18, 0)</td>
</tr>
</tbody>
</table>

Specifying Line Styles

Use a Pen clause to specify line styles. In a Map Catalog, you may need to specify two pen clauses: one to specify the appearance of linear features, and another to specify the appearance of region borders.

<table>
<thead>
<tr>
<th>Pen Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pen(thickness, pattern, color)</td>
<td>Pen(1, 2, 0)</td>
</tr>
</tbody>
</table>

Specifying Fill Styles

Use a Brush clause to specify the style for closed features (regions).

<table>
<thead>
<tr>
<th>Brush Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brush(pattern, color, backgroundcolor)</td>
<td>Brush(2, 255, 65535)</td>
</tr>
</tbody>
</table>
Text Objects Limitation

LegacyText objects have their own way of displaying style that is separate from the use of the MI_Style column. Therefore any form of text object needs to be treated differently than other objects. The style for any text object is embedded and a NULL value is inserted into the style column.

Troubleshooting

If you encounter problems with your SpatialWare or Oracle applications, use the following table to help analyze and solve the problem.

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Possible Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The table is not matchable.</td>
<td>Data binding was attempted against a SpatialWare layer.</td>
<td>AddColumns is not currently supported for SpatialWare layers.</td>
</tr>
<tr>
<td>No object was found using the index that you specified.</td>
<td>A query was made against a table that does not exist.</td>
<td>Check that the table name is correct and in the proper case. Also, the table may need to be mappable.</td>
</tr>
<tr>
<td></td>
<td>No spatial object is contained in the result of the spatial query.</td>
<td>Use the EasyLoader Upload utility to make the table a mappable table.</td>
</tr>
<tr>
<td></td>
<td>A query was made against a non-spatial table.</td>
<td>Check the query for possible syntax errors. Also make sure that the result of the query includes the field specified in the spatial column in the MapInfo_MapCatalog.</td>
</tr>
<tr>
<td>Map appears to have incorrect zoom level. For example, the map may be zoomed out too far to identify any geography.</td>
<td>The MBR for a DBMS layer is determined by the MapInfo_MapCatalog table. The table extents in the MapCatalog result in a different zoom level than the one you desire for your output.</td>
<td>Edit the extents (DB_X_LL, DB_X_UR, DB_Y_LL, DB_Y_UR) in the MapInfo_MapCatalog using the MapInfo Professional MBX tool, MISETMBR.MBX.</td>
</tr>
</tbody>
</table>
Adding Mapping Capability to Your Applications

We use the MapInfo.Mapping namespaces to add mapping functionality to your application. This chapter explains how to use the Mapping namespaces to enhance your mapping application.

In this chapter:
- Introduction to the Mapping Namespace .............................................. 230
- Base Mapping Classes ................................................................. 230
- Layers .......................................................................................... 233
- Labels ............................................................................................ 237
- Adornments .................................................................................... 240
- Feature Style Modifiers ................................................................. 242
- Printing Your Map ................................................................. 244
Introduction to the Mapping Namespace

The MapInfo.Mapping namespace contains classes, interfaces, and enumerations for creating, displaying, and exporting maps, layers, modifiers, thematics, legends, and labels.

The Map class is the top level object for a map in a desktop application. Each Map object can contain exactly one map. Each Map has one Adornment collection and one Layer collection. When developing a web application a Map gets attached to a MapExport object in order to export that map image to a bitmap or stream.

As described in Layers in Chapter 3 on page 59, a map is basically a set of layers piled on top of each other. Using the Mapping namespace classes, an application can be designed to manipulate those layers as needed.

Base Mapping Classes

This section discusses the base map classes in use in the MapInfo.Mapping namespace. The diagram below shows a UML representation of the Map hierarchy.

MapExport

The MapExport class is used to export a Map to an image. The properties of this class specify each aspect of the image, such as BorderPen, ExportSize, Format, etc.

MapExport supports several different image formats including: BMP (default), WBMP, WMF, EMF, GIF, J2K, JPG, PNG, PSD, TIFF, TIFFCymk, and TIFFLzw. When performance is an issue, the BMP, GIF, JPG, PNG, and TIFF formats can also be exported using the .NET framework API instead of the LEADTOOLS API. While the export speed may differ, the quality of the images exported is the same for both methods.

To select a format, set the ExportFormat property to one of the above formats (e.g., ExportFormat.WindowsPNG).

Note The LegendExport class also exports files in these formats using the .NET framework and LEADTOOLS APIs.
Example Use of MapExport

If you are using a MapControl, you must clone the map prior to exporting it, as shown in the example below:

VB example:

```vbnet
Public Shared Sub MapInfo_Mapping_MapExport(ByVal mapControl1 As MapControl)
    ' must clone since map is coming from mapcontrol and is linked to it via the HWND
    Dim NewMap As Map = CType(mapControl1.Map.Clone(), Map)
    Dim exportObject As MapExport = New MapExport(NewMap)
    exportObject.ExportSize = New MapInfo.Mapping.ExportSize(2931, 4104)
    exportObject.Format = ExportFormat.Gif
    exportObject.Export("C:\Temp\ExportImage.gif")
End Sub
```

Map

The Map class contains everything that you can put in a map. A Map object is placed in a MapControl object for a desktop application or attached to a MapExport object for a web application.

A Map has the following properties:

- Bounds
- Center
- Zoom
- Scale
- Size
- Rotation
- DisplayTransform
- DisplayCoordSys

Each Map has a Layers collection, which holds all the layers that comprise the map (see Layers on page 233), and a single Adornment collection, which contains all of the map’s adornments. Adornments include Legends, Titles, and Scalebars (see Legends Overview in Chapter 15 on page 272).

MapFactory

The MapFactory class contains the objects used to create maps from Geosets, workspaces, and lists of tables. MapFactory also functions as a container of, and tracks, the collection of all maps created in a particular session.
Example Use of MapFactory

This example creates an empty map 300 by 300 pixels in size using the CreateEmptyMap method of MapFactory.

' Create a new map
Dim map As Map = Session.Current.MapFactory.CreateEmptyMap(New Size(300, 300))

MapLoader

The MapLoader class provides a mechanism to load the layers of a map from Geoset files, XML Workspace files, or TAB files. For each type of map to load, there is a subclass of MapLoader that is used to load the map. These subclasses include MapGeosetLoader, MapWorkspaceLoader, and MapTableLoader.

VB example:

Public Shared Sub MapInfo_Mapping_MapLoad()
' Create an empty map
Dim map As Map =

' Create a maploader. Make sure that Session.Current.TableSearchPath
' points to the folder with the tables in it
Dim tl As MapTableLoader = New MapTableLoader("ocean.tab", "usa.tab", _,
"mexico.tab", "us_hiway.tab")

' Load tables into a map
map.Load(tl)
End Sub

MapViewList, MapView

These classes contain the objects that help to maintain a list of the previous and next views for a Map. Use the MapViewList class as a convenience to step through your stack of map views, displaying them with the MapView class which provides the specifics (Name, Center, and Zoom) of each map view.

MapXtreme provides a VB sample for MapViewList in `<MapXtreme Install Directory>\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features`.

MapControl

The MapControl class contains the objects that allow the user to visualize a map on a screen. Add a MapControl to a Windows Form to view a map. A MapControl also holds the MapTools collection. A version of MapControl exists for web applications as well.

The MapControl class for use in desktop applications is in the MapInfo.Windows.Controls namespace.
Chapter 13: Adding Mapping Capability to Your Applications

The MapControl class for use in web applications is in the MapInfo.WebControls namespace. See Web Control Architecture in Chapter 5 on page 80.

Layers

The following section discusses the Layers objects and classes. The diagram below is a UML representation of the Layers hierarchy.

- **FeatureLayer**
  
  A FeatureLayer is a layer that displays Features from a Table. For example, a layer of region objects representing world countries is a FeatureLayer. FeatureLayers can be native .TAB data, remote RDB, seamless, or raster data.

- **Layers**
  
  Each map has a collection of FeatureLayers represented by the Layers class. The order in the collection is the order in which the layers are drawn. Methods in the collection class include Add, Insert, Move, and Delete. The Layers collection is derived from the LayersBase class. The best way to enumerate through the layers is to use a Layer filter.

  In order to change a layer it first needs to be made editable. This can be done changing its setting in the LayerControl or programmatically by changing its EditMode property. Once a layer is editable, features in that layer can be moved, resized, or deleted.

  **Note** Any edits made to a particular layer take effect immediately so exercise caution when selecting features in an editable layer.

  To implement a filter use the MapLayerFilterFactory class. This class allows you to create a filter from a set of stock filters such as layer types or all visible layers. To create your own filter write a class that implements the interface, IMapLayerFilter.
Layers

**MapLayer**

The MapLayer class is the base class of any layer. This class implements the IMapLayer interface. Properties include Enabled, VisibleZoomRange, Name, and Alias. Use this class to access generic layer properties.

**UserDrawLayer**

The UserDrawLayer is an abstract class that allows you to override the draw method to draw your own layer. It provides an efficient way to add custom objects on top of a MapControl using Windows drawing methods, instead of creating new features using map coordinates and actually adding them to a map.

**ObjectThemeLayer**

The ObjectThemeLayer class contains one of three different types of themes (pie chart, bar chart, and graduated symbol). This layer behaves just like other layers and can be grouped, have visibility set, etc.

**GroupLayer**

This class represents a group of layers that are moved and turned on/off in unison. A GroupLayer is a LayersBase collection combined with an IMapLayer. This object also supports the ability to do AnimationLayers.

If the layer is in a group with a VolatilityHint equal to Animate then only those layers within that group need to be redrawn. If the Layer has a VolatilityHint equal to CacheIfPossible or Normal then all layers need to be redrawn.

**Layer Filters**

The IMapLayersFilter and IMapLayersFilterFactory interfaces provide support for layer enumeration.

**IVisibilityConstraint**

The IVisibilityConstraint is an interface which determines whether a particular object is visible or not. This interface is implemented by a wide number of types, including all Layer types, LabelModifiers, FeatureStyleModifiers, and Themes.

**Code Example: Animation Layer**

The following VB sample from the MapInfo Knowledgebase shows how to set the animation of a layer.

```vbnet
Private Sub btnInitializeObjects_Click(ByVal sender As System.Object, ByVal As System.EventArgs) Handles btnInitializeObjects.Click
Dim Cat As Catalog = MapInfo.Engine.Session.Current.Catalog
```
'Create Temp layer
Dim tblInfoTemp As New TableInfoMemTable("Animation")
Dim tblTemp As Table = Cat.GetTable("Animation")
If IsNothing(tblTemp) = False Then 'Table exists close it
    Cat.CloseTable("Animation")
End If
  tblTemp.Columns.Add(ColumnFactory.CreateStyleColumn())
  tblTemp.Columns.Add(ColumnFactory.CreateStringColumn("Name", 40))
  tblTemp.Columns.Add(ColumnFactory.CreateIntColumn("Level"))

  tblTemp = Cat.CreateTable(tblInfoTemp)

  Dim lyr As New FeatureLayer(tblTemp)
  MapControl1.Map.Layers.Add(lyr)

  'Create Points
  Dim pt As FeatureGeometry = New Point(lyr.CoordSys, New DPoint(-76, 42))
  Dim cs As New CompositeStyle(New SimpleVectorPointStyle(37, System.Drawing.Color.Red, 10))
  Dim ftr As New Feature(tblTemp.TableInfo.Columns)
  ftr.Geometry = pt
  ftr.Style = cs
  ftr.Item("Name") = "Kelly"
  ftr.Item("Dept") = "Sales"
  ftr.Item("Level") = 3
  tblTemp.InsertFeature(ftr)

  Dim pt2 As FeatureGeometry = New Point(lyr.CoordSys, New DPoint(-119, 34))
  Dim cs2 As New CompositeStyle(New SimpleVectorPointStyle(44, System.Drawing.Color.Purple, 10))
  Dim ftr2 As New Feature(tblTemp.TableInfo.Columns)
  ftr2.Geometry = pt2
  ftr2.Style = cs2
  ftr2.Item("Name") = "Greg"
  ftr2.Item("Dept") = "Marketing"
  ftr2.Item("Level") = 2
  tblTemp.InsertFeature(ftr2)
End Sub

Private Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles Timer1.Tick
  Dim tbl As Table = cat.GetTable("Animation")
  If IsNothing(tbl) = False Then 'Update the position of the points
    Dim si As SearchInfo = MapInfo.Data.SearchInfoFactory.SearchWhere("Name= _ 'Kelly'")
    Dim ftr As Feature = cat.SearchForFeature(tbl, si)
Dim si2 As SearchInfo =
    MapInfo.Data.SearchInfoFactory.SearchWhere("Name = 'Greg'")
Dim ftr2 As Feature = cat.SearchForFeature(tbl, si2)

If TimeOfDay.Now.Second Mod 4 = 0 Then
    ftr.Geometry.GeometryEditor.OffsetByXY(-5, -25, DistanceUnit.Mile, _
        DistanceType.Spherical)
    ftr2.Geometry.GeometryEditor.OffsetByXY(0, 25, DistanceUnit.Mile, _
        DistanceType.Spherical)
Else
    ftr.Geometry.GeometryEditor.OffsetByXY(-10, 0, DistanceUnit.Mile, _
        DistanceType.Spherical)
    ftr2.Geometry.GeometryEditor.OffsetByXY(10, 5, DistanceUnit.Mile, _
        DistanceType.Spherical)
End If
ftr.Geometry.EditingComplete()
ftr2.Geometry.EditingComplete()
ftr.Update()
ftr2.Update()

End If

End Sub
Labels

The following section discusses the Labels objects and classes. The diagram below is a UML representation of the Labels hierarchy.

LabelLayer

The LabelLayer class allows the separate ordering of labels and layers. A LabelLayer is a kind of MapLayer and behaves as such. This similarity to MapLayer supports the placement of a LabelLayer anywhere you can put a MapLayer allowing for relative positioning of the Labels compared to the other layers in the map. Each LabelLayer is made up of LayerSources and acts as a collection of those objects.

LabelSource

The LabelSource class graphically displays geographies as labels based on a data source and rules that define how the source is labeled. The LabelSource is added to a LabelLayer on the Map so that it is drawn. The LabelLayer provides positioning within the Map as well as the rules that govern the interaction with other LabelSources. To use LabelSource class, specify a table from where to get the data (MITable), an expression which defines the label text, and other default label properties accordingly.
Labels

LabelModifier

The LabelModifier class is used to modify the default properties used to make a label. When the LabelLayer generates labels for each LabelSource in its Sources collection, it first uses the DefaultLabelProperties to generate each label. It then uses each visible LabelModifier in the Modifiers collection.

ILabelSourceFilter

This interface allows you to enumerate through the collection of LabelSource objects in a LabelLayer filtering on specific rules. You can also implement this interface to define your own filtering rules.

LabelProperties

This class has label property information, such as style, positioning, etc. It supports the representation of sparse label properties. This is very useful when using a LabelModifier to modify only a portion of the label properties. This class also allows you to set prioritization and improved placement of labels.

Generating Labels

The LabelLayer class generates labels when the map draws or when you call the LabelLayer.Refresh method. Each visible label source is considered. You can have more than one layer of labels in a map.

To generate a label for each feature in the source’s table that is within current map view, the LabelLayer class does the following:

1. Uses the DefaultLabelProperties property as the starting set of properties used to build the label.
2. Calls the Modify method of each visible label modifier in the Modifiers collection, if any. This allows each modifier the opportunity for changing the label properties used to build the label.
3. Performs visibility constraint checks to decide whether to keep the label.
   a. Checks for label visibility by comparing the visibility constraints against the current map zoom/scale.
   b. If the label is visible and overlaps and duplicates are not allowed on this label, checks for overlap and duplicates against other existing labels. If any are found, uses Label Priorities to decide which label to keep.
4. Adds the label to the generated labels collection if it succeeds visibility constraints.

Note that Label generation rules only apply within each label layer and not the entire map. For example, if you set the AllowOverlap property to False in all label sources contained within all the label layers in a map, labels from one label layer will still overlap with labels from another label layer because the labels are generated independently between label layers.

Use the LabelLayer.Refresh method if the map has not yet drawn to generate the labels based on the current map view.
Chapter 13: Adding Mapping Capability to Your Applications

Accessing Generated Labels

You can access generated labels through the LabelSource.Labels property. This collection represents the labels that are within the current map view. They do not represent all the labels in the map. The items in the collection change as the view of the map changes.

Label Priorities

Label priorities determine which labels within the same LabelLayer are generated when either AllowOverlap or AllowDuplicates is set to False. Note: as stated in Generating Labels, each label layer is independent of each other, so different overlap or duplicate settings in other layers do not factor in here.

When a label overlaps with or is a duplicate of another label in the same layer, the priorities of both labels are compared to determine which label to keep. The process first compares the priority of each label (known as the inter-label source). The label with higher priority is kept.

MapXtreme provides two levels of priorities for controlling the display of labels: Major and Minor. This allows you to group and subgroup prioritization. For example, you might want cities with large populations to have labels with a higher priority than small towns. However, you can also add a modifier to bump up the major priority of one small town to give it a chance over the major cities.

When the Major priority is set to null (nothing in Visual Basic), the value used is the inverse of the label's LabelSource position within a LabelLayer. The higher the index position the lower the priority. For example, if a LabelSource is at index 3 and there are 10 LabelSources (indices from 0 to 9), the Major priority will be 6 (the inverse of the index based on total).

When the Minor priority is set to null (Nothing in Visual Basic), the value used is based on the Key of the Feature being labeled. The inverse of a numeric representation of the key against the number of rows in the Table is used. For example, if there are 10 rows in the Table, the minor priority of the label for feature in row ID 7 defaults to 3 (10 - 7 = 3). If the key is not numeric the minor priority defaults to 0.

When Major priorities are equal, the label with the higher Minor priority is kept.

To create a priority, use an expression that results in a numeric value. For example, the expression in C# that results in a numeric value could be a field of numeric type or an expression like the ASC value of the first letter in the field’s value:

"(1/Asc(Country))*1e6"

Label Layer Selectability

To control selectability of label layers, use the SelectMapToolProperties.LabelsAreEditable property programmatically or via Workspace Manager's Labels are Editable checkbox.

The LayerHelper.SetSelectable method does not affect the selectability of certain layer types, including Labels, WMS/WFS, Raster and Group layers.
Adornments

Code Example: Creating a LabelLayer

The following example demonstrates the use of the classes associated with Labels.

VB example:

' Open usa table using the data catalog
Dim table As Table = Session.Current.Catalog.OpenTable("usa.tab")

' Create a new map
Dim map As Map = Session.Current.MapFactory.CreateEmptyMap(New Size(300, 300))

' Create a new feature layer that references the table and add it to the map
Dim featureLayer As New FeatureLayer(table)
map.Layers.Add(featureLayer)

' Create a new label layer and add it to the map.
' Note that if you call MapInfo.Mapping.LayersBase.Add" method instead of
' MapInfo.Mapping.LayersBase.Insert method it will automatically position the
' label layer before the feature layer
Dim labelLayer As New MapInfo.Mapping.LabelLayer()
map.Layers.Insert(0, labelLayer)

' Create a new label source that references the table
Dim source As New MapInfo.Mapping.LabelSource(table)

' Change its caption expression to be a specific column from the table called "State_Name"
source.DefaultLabelProperties.Caption = "State_Name"

' Append the label source to the label layer so that it shows on the map
labelLayer.Sources.Append(source)

Adornments

The Adornments class is an unordered collection of a map’s adornments. An adornment is either a Legend, a Title, a Scalebar, or some other user-defined object. Each map contains a single Adornments Collection. Each adornment belongs to a single AdornmentControl.

To create your own adornment, derive a class from the IAdornment interface and the AdornmentControl abstract class.
Legends

Legends are for use in conjunction with themes. See Chapter 15: Using Themes and Legends for more information about themes. Legends are created using the LegendFactory class. A Legend consists of one or more legend frames. Each frame is either a ThemeLegendFrame or a CartographicLegendFrame. Both kind of LegendFrames are created by using the LegendFrameFactory class from the created Legend object. For manipulation and customization of legends and their frames, use classes in the MapInfo.Mapping.Legends namespace. Classes in this namespace include: CartographicLegendFrame, ThemeLegendFrame, LegendFormat, LegendControl, and others.

Note that you can set the size of the Legend with the Legend.Size property, but you cannot set the size of the LegendFrames. The size of the LegendFrame is controlled by the amount of data it contains.

VB example:

```vbnet
Private Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    LegendControl1.Map = MapControl1.Map
    If Not IsPostBack Then
        Dim normalLyr() As MapInfo.Mapping.LayerType = New _
        MapInfo.Mapping.LayerType(1) {}
        normalLyr(0) = MapInfo.Mapping.LayerType.Normal
        Dim filter As MapInfo.Mapping.IMapLayerFilter = _
        MapInfo.Mapping.MapLayerFilterFactory.FilterByLayerType(normalLyr)
        Dim frame As MapInfo.Mapping.Legends.LegendFrame
        Dim NewLegend As MapInfo.Mapping.Legends.Legend = _
        NewLegend.Format.FrameAlignment = _
        For Each ftrLayer In _
            frame = _
            LegendFrame(ftrLayer)
            NewLegend.Frames.Append(frame)
        Next
        LegendControl1.Legend = NewLegend
    Else
        LegendControl1.Legend = MapControl1.Map.Legends(0)
    End If
End Sub
```

ScaleBar Adornment

A scale bar is linear object used to measure map distances in the units of the map (such as kilometers or feet). Use the ScaleBarAdornment class to create a ScaleBarAdornmentControl and add the control to the map. If you attempt to add a ScaleBarAdornment to the map itself, it does display, but without the ScaleBarAdornmentControl, the ScaleBar is not linked to the map itself.
Feature Style Modifiers

VB example:

Public Shared Sub MapInfo_Mapping_ScaleBarAdornment(ByVal mapControl1 As _
MapControl)
    ' Create a scalebar
    Dim sba As ScaleBarAdornment = New ScaleBarAdornment(mapControl1.Map)

    ' Position the scalebar at the lower right corner of map
    Dim x As Integer = mapControl1.Map.Size.Width - sba.Size.Width
    Dim y As Integer = mapControl1.Map.Size.Height - sba.Size.Height
    sba.Location = New System.Drawing.Point(x, y)

    ' Add the control to the map
    Dim sbac As ScaleBarAdornmentControl = New
    ScaleBarAdornmentControl(sba, _
    mapControl1.Map)
    mapControl1.AddAdornment(sba, sbac)
End Sub

Title Adornment

A Title adornment is a text object drawn on the map to represent a map title or to provide a text to clarify other information on the map. In MapXtreme you use the TitleAdornment class to create a title and add it to the map.

VB example:

Public Shared Sub MapInfo_Mapping_TitleAdornment(ByVal mapControl1 As
MapControl)
    ' Create a Titlebar
    Dim ta As New MapInfo.Mapping.TitleAdornment(New
    System.Drawing.Size(100, 50), mapControl1.Map)
    ta.Title = "This is a watermark"
    mapControl1.Map.Adornments.Append(ta)
End Sub

Feature Style Modifiers

Feature style modifiers allow a way of changing/modifying the style of a Feature before it is drawn. These classes use the sparse attribute aspect of CompositeStyle object to only modify the part of the style in which you are interested. Range, Individual Value, and Dot Density themes are style modifiers. Thematics are covered in detail in Chapter 15: Using Themes and Legends. The diagram below shows a UML representation of the Modifier and thematics hierarchy.
FeatureStyleModifier

This is an abstract base class from which all modifiers must derive. The IndividualValue, Ranged, and DotDensity themes are all FeatureStyleModifier objects. You can create your own class that derives from FeatureStyleModifier and override the Modify() method.

FeatureStyleModifiers

The FeatureStyleModifiers class is an ordered collection of FeatureStyleModifier objects contained within each FeatureLayer. Each modifier in the collection is called in order before a feature's geometry is drawn.

FeatureOverrideStyleModifier

The FeatureOverrideStyleModifier is a simple kind of FeatureStyleModifier. This class provides style override functionality at the Layer level. The FeatureOverrideStyleModifier has a composite style and implements the IVisibilityConstraint. This is similar to functionality found in MapX and MapInfo Professional.
Once your application is created you may want to give your user the opportunity to print out their generated map. The MapInfo.Printing namespace is provided to facilitate including printing functionality into any application. This classes in this namespace are built upon the .NET Framework classes for printing so the necessary constructions are similar to any other Windows application. In addition to regular printing we have provided many additional capabilities to optimize the printed version of a created map.

Refer to Appendix E: Printing From MapXtreme Applications for more detailed information about printing from a MapXtreme application.
Finding Locations

The MapInfo.Data.Find namespace contains the classes to use when searching for map features by address, street intersection or name.

In this chapter:

- Functional Overview of Find ............................................. 246
- Overview of the Data.Find Namespace ................................. 248
- Fine Tuning the Find Process ............................................. 254
Find is used when you wish to locate map features by address, street intersection or name. To find features at locations using selection tools or queries, use the Search classes in the Data namespace (see Searching for Features on page 193).

A successful Find operation can result in an exact match, one or more close matches, or no match (failed match). The operation is controllable by a variety of properties and fallback options and are discussed in the Overview of the Data.Find Namespace on page 248. The section below describes how MapXtreme makes a match. The more you understand the Find process, the more you can control the success of the operation.

The Find Process

MapXtreme locates map features by matching an address, street intersection or place name with information in the feature table. For example, you can find 1600 Pennsylvania Avenue in Washington, D.C. provided a table of Washington D.C. streets is open.

Finding a map feature is similar. For example, you can find The White House if you have a mappable table of landmarks that contains the name “The White House” and its geo-referenced (mappable) location. You don’t need to supply the address to find by place name.

To find a street intersection you must provide both street names that make up the intersection.

MapXtreme attempts to find an exact match in which the result is a character by character match to the input address, place name or intersection. If it cannot make an exact match, MapXtreme attempts to find close matches based on its matching rules and your settings. If it cannot find a close match, it returns a failed match. Note that matching is not case-sensitive. Upper and lower case characters are successfully matched with one another.

A street address is normally made up of a street number, name and abbreviations that represent the street prefix, such as North, and suffix, such as Road. Addresses can take a variety of forms and may include additional information such as apartment number or route. In addition, some of the key components like street type may be missing from the input address. MapXtreme looks at each component of the address and applies specific rules to it to find a match.

The following sections describe how MapXtreme handles specific information and conditions including street name, street abbreviations, address numbers, refining tables and results.

Matching to Street Name

A match to a street name is a straightforward character by character evaluation of the address against the information in the search table. For example, MapXtreme returns an exact match for the street name LaSalle if LaSalle is in the table, but may only return a close match if the address is spelled La Salle or LaSal.

Matching to Street Abbreviations

Street abbreviations vary widely in address records. Sometimes the component is missing altogether. In many cases, however, MapXtreme can make an exact match even if there are slight differences between the address and the search table. MapXtreme refers to a standardized address
abbreviation and substitution list, called Mapinfow.abb, to find a suitable match. This list contains standard spellings of street prefix and suffix abbreviations, such as ST for Street and Av for Avenue. You must set a property to tell MapXtreme to use this abbreviations file, but it is a good way to increase the chance of exact matches or find more close matches than without the file.

The table below illustrates variations in address and whether or not it will result in an exact match based on the use of the abbreviations file. The first column contains a street name you want to find. The second column contains the corresponding street name from a source table. The third column says why they do not match. The fourth column indicates whether the particular problem is one that can be corrected by using the abbreviation equivalence file. This table assumes that the addresses are the contents of a single column in a table. While the street number would often be in the same column, we don’t indicate street numbers here because they are handled differently than street names.

<table>
<thead>
<tr>
<th>Address to Find</th>
<th>Search Table Address</th>
<th>Comment</th>
<th>Correctable with Abbreviation File for an Exact Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaSalle Street</td>
<td>LaSalle St</td>
<td>&quot;Street&quot; and &quot;St&quot; do not match.</td>
<td>Yes</td>
</tr>
<tr>
<td>LaSalle Ave</td>
<td>LaSalle Av</td>
<td>&quot;Ave&quot; and &quot;Av&quot; do not match.</td>
<td>Yes</td>
</tr>
<tr>
<td>LaSalle Ave</td>
<td>LaSalle St</td>
<td>&quot;Ave&quot; and &quot;St&quot; do not match.</td>
<td>No</td>
</tr>
<tr>
<td>LaSalle</td>
<td>LaSalle St</td>
<td>&quot;St&quot; is missing from target.</td>
<td>No. If the street abbreviation is missing, MapXtreme does make a guess on what it could be.</td>
</tr>
<tr>
<td>LaSalle St North</td>
<td>LaSalle St</td>
<td>&quot;North&quot; is not in the search table.</td>
<td>No</td>
</tr>
<tr>
<td>LaSalle St North</td>
<td>LaSalle St N</td>
<td>&quot;North&quot; and N do not match.</td>
<td>Yes</td>
</tr>
<tr>
<td>LaSalle St Apt 3</td>
<td>LaSalle St</td>
<td>Apartment number does not match anything in the source.</td>
<td>Yes. The apartment number is ignored.</td>
</tr>
<tr>
<td>Tenth St</td>
<td>10th St</td>
<td>&quot;Tenth&quot; and &quot;10th&quot; do not match.</td>
<td>Yes</td>
</tr>
<tr>
<td>10th Av</td>
<td>Tenth Av</td>
<td>&quot;10th&quot; and &quot;Tenth&quot; do not match.</td>
<td>Yes</td>
</tr>
<tr>
<td>Saint John’s Lane</td>
<td>St John’s Lane</td>
<td>&quot;Saint&quot; and &quot;St&quot; do not match.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

If you find that you have repeated situations that do not match due to abbreviations, you can:

• Edit your addresses to conform closer to the abbreviations file, or
Overview of the Data.Find Namespace

- Edit the abbreviations file with a text editor to add your specific abbreviations. The mapinfow.abb is located in C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x. For more information about editing the abbreviations file, see Fine Tuning the Find Process on page 254.

Matching Address Numbers

MapXtreme can match when the address number precedes the street name (as in North American addresses) or when it follows the street name (common in European addresses). By default MapXtreme assumes the address number precedes the street name. You must set a property when the address number follows the street name.

MapXtreme compares the input address number to a range of address numbers. Typically the table contains address number ranges that correspond to the portion of the street the range covers. Address ranges can be matched to the exact side of the street, since often address ranges are odd-numbered on one side of the street and even-numbered on the other.

In cases where MapXtreme cannot match the address range exactly, you may be willing to settle for a close match in which the closest range would be considered a match. You would use the stricter exact match requirement only for very precise finds. Often that level of precision isn’t necessary. A close match is usually acceptable.

Matching with a Refining Boundary Table

MapXtreme can also find an address in one table in which there may be more than one possible match. To avoid finding the wrong address, you specify a refining table and column to focus the match to a smaller area.

This is useful, for example, when searching a table of streets that cover an entire county and you are looking for Main St. It's likely that more than one town in the county will have a Main St. By providing a refining table of town boundaries, you can specify that you are looking for the Main St. only in Town A.

You can use any type of refining boundary table you wish, for example, postal code boundaries or census regions. Additionally, you can specific a second refining boundary in which to conduct a Find.

Find Results

MapXtreme returns either an exact match, one or more close matches, or a failed match. The results depend on a number of factors including the quality of the input data and the conditions set for the operation. Information is also returned that explains how well each portion of the address matched (or failed to match).

Overview of the Data.Find Namespace

The MapInfo.Data.Find namespace contains classes that enable you to locate a map feature, street address, or street intersection within a mappable table. The properties and methods of the Find class are used to set up a Find. The results of a Find are returned using the FindAddressRange, FindCloseMatch, and FindResult classes.
The following UML diagram illustrates the Find namespace.

Find

The Find object is used to locate a map object, street address, or street intersection within a given mappable table. Find searches the table for a match and returns the result(s) in a FindResult object.

To use Find in MapXtreme, you must have:

- A mappable table (a table that contains geometry objects)
- An indexed column on which to perform the search
- An item to search for, such as a place name, street address or street intersection
- Optionally, a refining table in which to narrow the search to a more specific location.

The Find class provides a number of properties to control the search operation. For example, you can limit the number of close matches to be returned (CloseMatchesMax) or indicate that you wish to use the abbreviation file for increasing the likelihood of a match (UseAbbreviations).

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddressNumberAfterStreet</td>
<td>Specifies whether the address number is located after the street name (for example, “Smith Street 107”).</td>
</tr>
<tr>
<td>ChooseAlternateBoundary</td>
<td>Specifies whether to match a record found in a refining region other than the refining region specified.</td>
</tr>
<tr>
<td>ChooseClosestAddressRange</td>
<td>Specifies whether to use the closest available address number in cases where the address number does not match.</td>
</tr>
<tr>
<td>ChooseClosestObject</td>
<td>Specifies whether or not to find the closest object match when an exact match is not found.</td>
</tr>
</tbody>
</table>
The Find class provides four search methods: two for searching addresses or features with or without a refining boundary, and two for searching street intersections with or without a refining boundary.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CloseMatchesMax</td>
<td>Specifies how many close matches to return if an exact match is not found.</td>
</tr>
<tr>
<td>InsetDistance</td>
<td>A positive value representing how far from the ends of the line to adjust the placement of an address location.</td>
</tr>
<tr>
<td>InsetPercentage</td>
<td>Represents the percentage of the length of the line where the address is to be placed.</td>
</tr>
<tr>
<td>InsetUnit</td>
<td>Represents the distance unit to use for Inset.</td>
</tr>
<tr>
<td>OffsetDistance</td>
<td>Representing the distance to offset the placement of an address location back from the street.</td>
</tr>
<tr>
<td>OffsetUnit</td>
<td>Represents the distance unit to use for Offset.</td>
</tr>
<tr>
<td>UseAbbreviations</td>
<td>Specifies whether substituting abbreviations from the abbreviations file are used to find a match (for example, “Smith Street” is substituted with “Smith St”).</td>
</tr>
<tr>
<td>UseCloseMatches</td>
<td>Specifies whether to return “N” number of close matches if an exact match is not found.</td>
</tr>
<tr>
<td>UseInsetAsPercentage</td>
<td>Specifies whether Inset is being used as a percentage or as a distance.</td>
</tr>
</tbody>
</table>

FindAddressRange

The FindAddressRange object represents an address range item returned from the Find.Search method. The FindAddressRange object is returned as part of the FindResult object when a street address is not found, the address number is not within the minimum/maximum address ranges for a given street, or an address number was not specified.
Chapter 14: Finding Locations

### Code Sample

```csharp
public void GetAddressRangesOnStreetTable()
{
    Table _table;
    _table = Session.Current.Catalog.OpenTable("North_Greenbush.tab");

    Find _find = new Find(_table, _table.TableInfo.Columns[1]);
    FindResult _findResult = _find.Search("Meadow Dr");
    {
        FindAddressRangeEnumerator _enum = _findResult.GetAddressRangeEnumerator();
        FindAddressRange _findAddressRange;
        int _iIndex = 0;

        while (_enum.MoveNext())
        {
            _findAddressRange = _enum.Current;
            Console.WriteLine("_findAddressRange.BeginRange");
            Console.WriteLine("_findAddressRange.EndRange");

            _iIndex++;
        }
    }
    if (_table != null)
    {
        _table.Close();
        _table = null;
    }
}

find.Dispose();
}
```

### FindCloseMatch

The FindCloseMatch object represents a close match item returned from the Find Search method. The object is returned as part of the FindResult object. A close match item is a returned match that closely matches the name of the requested search.

To use this feature, you must first set the UseCloseMatches and CloseMatchesMax properties of the Find object before you execute your search. For example if you tried to search for “Washington Street,” and UseCloseMatches was set to true, a close match would be “Washington Ave.”

### Code Sample

```csharp
public void CloseMatchesOnStreetTable()
{
    Table _table;
    
    Find _find = new Find(_table, _table.TableInfo.Columns[1]);
    FindResult _findResult = _find.Search("Meadow Dr");
    {
        FindAddressRangeEnumerator _enum = _findResult.GetAddressRangeEnumerator();
        FindAddressRange _findAddressRange;
        int _iIndex = 0;

        while (_enum.MoveNext())
        {
            _findAddressRange = _enum.Current;
            Console.WriteLine("_findAddressRange.BeginRange");
            Console.WriteLine("_findAddressRange.EndRange");

            _iIndex++;
        }
    }
    if (_table != null)
    {
        _table.Close();
        _table = null;
    }
}

find.Close();
}
```
_table = Session.Current.Catalog.OpenTable("Rensselaer.tab");

Find _find = new Find(_table, _table.TableInfo.Columns[1]);
_find.UseCloseMatches = true;
_find.CloseMatchesMax = 5;

FindResult _findResult = _find.Search("70 Washington");

if (!(_findResult.ExactMatch) && (_findResult.NameResultCode == FindNameCode.ExactMatchNotFound))
{
    FindCloseMatchEnumerator _enum = _findResult.GetCloseMatchEnumerator();
    FindCloseMatch _findCloseMatch;
    int _iIndex = 0;

    while (_enum.MoveNext())
    {
        _findCloseMatch = _enum.Current;

        Console.WriteLine(_findCloseMatch.Name);
        _iIndex++;
    }
}

if (_table != null)
{
    _table.Close();
    _table = null;
}
find.Dispose();

FindResult
The FindResult class returns information from the Find.Search method in the form of properties that explain what kind of match was made, as shown in the table below. If successful, the FoundKey property contains the key of the object to be located. If successful, the FoundPoint property contains the point of the object located.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddressOutOfRange</td>
<td>Specifies whether the address passed in was out of range.</td>
</tr>
<tr>
<td>AddressResultCode</td>
<td>Identifies the result code for the address part of the search and returns a FindAddressCode enumeration.</td>
</tr>
<tr>
<td>BoundaryResultCode</td>
<td>Identifies the result code for the refining boundary part of the search and returns a FindBoundaryCode enumeration.</td>
</tr>
<tr>
<td>ExactMatch</td>
<td>Specifies whether an exact match was found.</td>
</tr>
</tbody>
</table>
### Property Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FoundKey</td>
<td>Specifies the Key of the object that has been located.</td>
</tr>
<tr>
<td>FoundPoint</td>
<td>Specifies the Point of the object that has been located.</td>
</tr>
<tr>
<td>IntersectionNotFound</td>
<td>Specifies whether the intersection was not found.</td>
</tr>
<tr>
<td>MultipleMatches</td>
<td>Specifies whether multiple matches were found.</td>
</tr>
<tr>
<td>NameResultCode</td>
<td>Identifies the result code for the name being searched on and returns a FindNameCode enumeration.</td>
</tr>
<tr>
<td>ResultCode</td>
<td>If the Find results is an exact match, the value is one. If the Find results in an approximate match, the value is greater than one. If the Find fails to match the address, the result is a negative value.</td>
</tr>
<tr>
<td>UseSubstitution</td>
<td>Specifies whether a substitution was used from the abbreviation file.</td>
</tr>
</tbody>
</table>

### FindAddressCode Enumeration

Identifies the result code for the address part of the search and is returned by the FindResult.AddressResultCode property.

**Note** This result code should only be used when searching for a street or intersection.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExactMatch</td>
<td>An exact match was found.</td>
</tr>
<tr>
<td>SideOfStreetUndetermined</td>
<td>The side of the street was undetermined.</td>
</tr>
<tr>
<td>WithinMinMax</td>
<td>The address number was within minimum/maximum range.</td>
</tr>
<tr>
<td>NotWithinMinMax</td>
<td>The address number was not within minimum/maximum range.</td>
</tr>
<tr>
<td>AddressNumNotSpecified</td>
<td>An address number was not specified.</td>
</tr>
<tr>
<td>StreetsDoNotIntersect</td>
<td>The streets do not intersect.</td>
</tr>
<tr>
<td>NoMapObjectForRowMatched</td>
<td>The row matched does not have a map object.</td>
</tr>
</tbody>
</table>

### FindBoundaryCode Enumeration

Identifies the result code for the refining boundary part of the search and is returned by the FindResult.BoundaryResultCode property. A refining boundary is used to distinguish between multiple features with the same name.

**Note** This result code should only be used when a region is being used to refine the search.
FindNameCode Enumeration

Identifies the result code for the name being searched on and is returned by the FindResult.NameResultCode property.

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExactMatch</td>
<td>An exact match was found.</td>
</tr>
<tr>
<td>FoundInOneOtherRegion</td>
<td>The name was found in only one region other than specified region.</td>
</tr>
<tr>
<td>FoundInMoreThanOneRegion</td>
<td>The name was found in more than one region other than the specified region.</td>
</tr>
<tr>
<td>NoRegionSpecifiedOneMatch</td>
<td>No refining region was specified, and one match was found.</td>
</tr>
<tr>
<td>NoRegionSpecifiedMultipleMatches</td>
<td>No region was specified, and multiple matches were found.</td>
</tr>
<tr>
<td>MultipleMatchesFound</td>
<td>The name was found more than once in the specified region.</td>
</tr>
</tbody>
</table>

Fine Tuning the Find Process

As we stated in the beginning of this chapter, the more you understand about how Find works, the better you can use its properties and your input information to improve the chances for a successful match. This section provides some help with developing strategies for improved results during a Find.
Editing the Mapinfow.abb File

The Mapinfow.abb file is an abbreviations and substitution file that ships with MapXtreme. This file can be used to improve the chances for a Find if the abbreviation in the input address can be found in the abbreviations file. This is covered in the section Matching to Street Abbreviations on page 246. This section covers additional types of information you can include in the file.

You can edit Mapinfow.abb in any text editor or word processor. Open the file and make your additions, adding keywords as necessary. Here is a standard version of the file:

```
!Version 3.0
FIRST 1ST
SECOND 2ND
THIRD 3RD
FOURTH 4TH
FIFTH 5TH
SIXTH 6TH
SEVENTH 7TH
EIGHTH 8TH
NINTH 9TH
TENTH 10TH
NORTH N
SOUTH S
EAST E
WEST W
ALLEY AL
AVENUE AV
AVE AV
BOULEVARD BLVD
BRIDGE BR
CIRCLE CIR
COURT CT
DRIVE DR
EXTENSION EXT
HIGHWAY HWY
INTERSTATE I
LANE LN
MOUNT MT
PARK PK
PARKWAY PKWY
PLACE PL
PLAZA PLZ
POINT PT
RAILROAD RR
ROAD RD
ROUTE RT
SAINT ST
SQUARE SQ
STREET ST
STR ST
TERRACE TER
!EOLNOSPACE
,```
Space-delimited Substitution

Space-delimited simple substitution is the default. What that means is this: MapXtreme compares space-delimited tokens in target addresses with the rows in the address file. A space-delimited token is a string of characters with a space before and a space after. For example, MapXtreme will match “Ave” with “Av” in “Park Ave” but it will not match “Avery Blvd” to “Avry Blvd.” Both street names contain the string “Ave”. But that string is bordered by spaces only in “Park Ave”, not in “Avery Blvd.” In “Avery Blvd”, “Ave” is followed by “r”, not by a space.
All of the entries in the abbreviation file will receive this default interpretation. You can add other items to receive the same treatment. For example, you might want to add the pair “WK WALK” so that MapXtreme knows to interpret “WK” in a target address as though it were “WALK”. Similarly, you might want to add a pair such as: “WAY WY”.

Use the keyword “!SPACE” to indicate space-delimited simple substitution. Entries following “!SPACE” are given the default interpretation (this allows you to arrange the Abb.file contents in some other order). When MapXtreme encounters another keyword, it switches to the indicated interpretation strategy.

**Simple Truncation**

In simple truncation, MapXtreme finds an item in the address and simply ignores it and everything after it. These items do not have to be space-delimited. This strategy is useful for dealing with addresses such as:

- 123 Appian Way, Mail Stop 829
- 7305 Van Zandt # 23

In the first case, you want MapXtreme to ignore the comma and everything after it. In the second case you want MapXtreme to ignore the number sign and everything after it. To deal with such cases add the following to your abbreviation file:

```
!EOLNOSPACE
,
#
```

“!EOLNOSPACE” is the keyword indicating that the following items are to be treated as cases of simple truncation. After that we have one line with a comma and one with a number sign. Whenever MapXtreme encounters a comma or a number sign in an address it will ignore it and everything after. The examples become:

- 123 Appian Way
- 7305 Van Zandt

**Space-Delimited Truncation**

In space-delimited truncation MapXtreme looks for items which are space delimited and eliminates those items and everything following. For example:

- 73 Appian Way Suite 829
- 3033 Van Zandt Room 202

To deal with such cases add the following to your abbreviation file:

```
!EOLSPACE
SUITE
ROOM"
```

!EOLSPACE” is the keyword indicating that the following items are to be treated as cases of simple truncation. After that we have one line with “Suite” and one with a “ROOM”. Whenever MapXtreme encounters those tokens it will truncate the address. The examples become:

- 73 Appian Way
3033 Van Zandt

**Simple Substitution**

MapXtreme uses simple substitution to remove items from an address and otherwise does nothing. Use it to deal with:

433 Van-Rensselaer

91 St Albans'

The goal is to strip out the hyphen and the apostrophe. Make the following entries to the abbreviation file:

```
!NOSPACE
-
'
```

“NOSPACE” is the keyword calling for simple substitution, and the hyphen and apostrophe on the following lines are the tokens to be removed. The examples become:

369 VanRensselaer

91 St Albans

**Legitimate Spaces**

There are cases where you want to indicate a substitution in which the searched for string contains spaces. You can use double quotes in such cases. Place a double quote:

• at the beginning of the line; and
• between the searched for string and the substitution; and
• at the end of the line.

For example, you might want to substitute “STHWY” for “State Highway”. To do that, use the following line:

```
"State Highway"STHWY"
```

This provides a solution to a subtle problem, that of street names which match items in the abbreviation file. For example, “North St” and “Park Av” both have initial strings which match terms in the abbreviation file. Consequently, MapXtreme will substitute “N” for “North” to yield “N St” and “Pk” for “Park” to yield “Pk Av.” You could add the following lines to the Abbreviation file to rectify these substitutions:

```
"N ST"North ST"
"PK AV"PARK AV"
```

Note that these lines have to come after the entries which substitute “N” for “North” and “PK” for “Park”. If they came before, they would have no effect. Thus:

```
... 
... 
NORTH   N 
... 
PK      PK 
```
When MapXtreme encounters NORTH N it will turn NORTH ST into N ST. When it encounters “N ST"NORTH ST” it will then turn N ST into NORTH ST. PARK AV is treated similarly.

**Special Characters**

MapXtreme uses the exclamation point (!), the double quote (") and the backslash (\) as special characters. These characters tell MapXtreme how to treat strings which follow them, but are not themselves ordinarily treated as characters in substitution strings. The exclamation point tells MapXtreme that the string should not be interpreted as an abbreviation. The double quote tells MapXtreme that spaces in the string are legitimate. And the backslash tells MapXtreme to treat a special character as an ordinary character.

When you want to use any of these in a line where they are to be treated as simple characters, precede them by a backslash. Thus:

```
! \\
\"
```

**Adding Lines to the Abbreviation File**

You can add a new item to the file by adding a new row. The order in which you add rows is not significant, except in those cases where you expect one substitution pair to compensate for the effects of another. The number of spaces between the first and second items in a row is not significant either.

**Incorrect Address Ranges**

When an address contains a number range that is not in the source table, MapXtreme will not be able to match it. Such an address might fall into a gap in range numbers or it might be beyond the ends of the ranges. To handle this problem:

1. Set the Find.ChooseClosestAddressRange property to `true` and then perform your search.
2. Resolve the failed matches by reviewing the FindAddressCode enumeration which is returned by the FindResult.AddressOutOfRange property.

It is possible that the address is for a street segment that was added after your source table was made. In that case, edit the source table so that it reflects the full range of addresses for that street.

**Inaccurate Town Names**

MapXtreme’s last step in a Find operation is to determine in which region to place a matched street address. MapXtreme only takes this step if you have so specified when you set up the find process. It is common to use town or city name as the refining boundary. This causes problems because
people often do not use the town name which the Census Bureau assigns to their address. Since almost all electronic maps of the United States are based on Census Bureau maps, this will cause problems.

For example, the address “50 Wolf Rd., Albany, NY” is actually in the town of Colonie. Thus the address town name in the target address will not match the appropriate town in the source file.

One way to handle this is use the ChooseAlternateBoundary property. When this option is enabled, MapXtreme attempts to match an address to whatever boundary that address is in, providing that address is in only one boundary. When the address is in more than one boundary, the Find will fail.

Another way to deal with this problem is to use the ZIP Code as the refining boundary, rather than the town or city name.
Using Themes and Legends

MapXtreme provides you many options for adding thematics and legends to your map. The following sections illustrate the different types of themes and legends available to you and explains how to use them.

In this chapter:
- Thematics Overview ................................................. 262
- GraduatedSymbolTheme ............................................. 263
- PieTheme ............................................................. 265
- BarTheme .............................................................. 266
- RangedTheme ......................................................... 266
- RangedLabelTheme ................................................... 269
- IndividualValueTheme .............................................. 269
- IndividualValueLabelTheme ...................................... 270
- DotDensityTheme .................................................... 271
- Legends Overview ................................................... 272
Thematics Overview

Thematic mapping allows you to present trends in data that would be difficult to see from tabular data. The theme is usually some piece or pieces of your data. You thematically shade a map using data from a data source, such as a native MapInfo table. For example, you can thematically shade a map of the United States based on the average temperature of each state. When you see red, you know it is hot (high number of degrees); where you see blue, it is cold (low number of degrees).

Themes represent your data with shades of color, fill patterns, or symbols. There are many uses for thematic maps to display your data. You create different thematic maps by assigning these colors, patterns, or symbols to map objects according to specific values in your data.

Mapping.Thematics Namespace

The MapInfo.Mapping.Thematics namespace contains classes that implement themes as style overrides on Feature layers and as Object themes. Modifier themes change the style, while object themes add a new layer. All themes implement the ITheme interface.

Modifier Themes

An example of a feature modifier theme are ranged, individual value and dot density thematic maps. They modify existing features in a layer. The following UML diagram gives an overview of the modifier theme hierarchy.
Object Themes

Object themes include graduated symbol, and pie and bar charts. These themes create objects that represent the data values. The following UML diagram gives an overview of the object theme hierarchy.

GraduatedSymbolTheme

A graduated symbol theme is an object theme that contains point features whose symbol sizes are based on the numeric values of the theme’s expression.

For instance, use graduated symbols to show housing income for a particular segment of the population across an area.
GraduatedSymbolTheme

Example of a Graduated Symbol thematic map.

When To Use a Graduated Symbol Theme

Graduated symbols maps work with numeric data only. It makes no sense to create graduated symbols based on the type of cuisine each restaurant serves. However, graduated symbols are appropriate when you want to show the distribution of housing income by city.

VB example:

Public Shared Sub MapInfo_Mapping_Thematics_GraduatedSymbolTheme(ByVal ByVal map As Map)

' Load a map based on one table
map.Load(New MapTableLoader(“world.tab”))
Dim lyr As FeatureLayer = CType(map.Layers(“world”), FeatureLayer)

' Create a new graduated symbol theme
Dim gradTheme As GraduatedSymbolTheme = New _
GraduatedSymbolTheme(lyr.Table, “Pop_Native”)

' Create an object theme layer based on that graduated symbol theme
Dim thmLayer As ObjectThemeLayer = New ObjectThemeLayer(“World Pop _
Growth Rate”, Nothing, gradTheme)

' Add object theme to the map’s layer collection.
map.Layers.Add(thmLayer)

' Adjust how we graduate the size.
gradTheme.GraduateSizeBy = GraduateSizeBy.Constant
thmLayer.RebuildTheme()

End Sub
PieTheme

A pie theme is an object theme containing pie charts with wedges that represent each data value. In pie charts you compare the wedges in a single pie, or examine a particular pie wedge across all of the pies. Pie charts also enable you to compare parts of a whole.

When To Use a Pie Theme

Pie charts are particularly useful for analyzing demographic data. For example, you have a dataset of demographic information for the United States. Your dataset shows the populations of several major demographic groups. Using pie charts, you can show the population of each demographic group, and see what fraction of the pie it makes up in each chart. This enables you to see the distribution of demographic groups on a per state basis, or across the entire United States. You can also look at one demographic group and see how the relative population of the group varies in different states.

VB example:

```vbnet
Public Shared Sub MapInfoMappingThematicsPieTheme(ByVal map As Map)
    ' Load a map based on one table
    map.Load(New MapTableLoader("mexico.tab"))
    Dim lyr As FeatureLayer = CType(map.Layers("mexico"), FeatureLayer)

    ' Create a new pie theme
    Dim pieTheme As MapInfo.Mapping.Thematics.PieTheme = New _
        MapInfo.Mapping.Thematics.PieTheme(map, lyr.Table, "Cars_91", _
        "Buses_91", "Trucks_91")

    ' Create an object theme layer based on that pie theme
    Dim thmLayer As ObjectThemeLayer = New ObjectThemeLayer("Count by _
        Vehicle Type", Nothing, pieTheme)

    ' Add object theme to the map's layer collection.
    map.Layers.Add(thmLayer)

    ' DataValueAtSize is calculated automatically if not specified in the
    ' pie's constructor. But, you can adjust it. If you do so here, you
    ' have to rebuild the theme. You can adjust it before creating the
    ' object theme layer, and that way the pies won't need to be built
    ' twice.

    pieTheme.DataValueAtSize /= 2
    pieTheme.GraduateSizeBy = GraduateSizeBy.Constant
    thmLayer.RebuildTheme()
    End Sub
```
BarTheme

A bar theme is an object theme that contains bar charts with bars that represent each data value. A bar chart is built for every map object (feature) at the centroid of the object, enabling you to analyze several thematic variables in a particular chart by comparing the height of the bars.

When To Use a Bar Theme

Bar themes are useful for examining the same variable across all the features in your map. For example, you have a table of U.S. state boundaries containing female and male population. Using bar charts, you can create a thematic map that displays a two–bar chart for each state: one bar representing female, and the other representing male population. You can compare the population differences of each state, or you can examine several states and compare population differences to the others.

VB example:

```vbnet
Public Shared Sub MapInfo_Mapping_Thematics_BarTheme(ByVal map As Map)
    ' Load a map based on one table.
    map.Load(New MapTableLoader("world.tab"))
    Dim lyr As FeatureLayer = CType(map.Layers("world"), FeatureLayer)

    ' Create a new bar theme.
    Dim barTheme As MapInfo.Mapping.Thematics.BarTheme = New _
    MapInfo.Mapping.Thematics.BarTheme(map, lyr.Table, "Pop_Native", _
    "Pop_Asian", "Pop_Other")

    ' Create an object theme layer based on that bar theme.
    Dim thmLayer As ObjectThemeLayer = New ObjectThemeLayer("World _
    Pop", Nothing, barTheme)

    ' Add object theme to the map’s layer collection.
    map.Layers.Add(thmLayer)

    ' Stack the bars and graduate by a constant amount.
    barTheme.Stacked = True
    barTheme.GraduateSizeBy = GraduateSizeBy.Constant
    thmLayer.RebuildTheme()
End Sub
```

RangedTheme

A ranged theme shows data grouped into ranges (bins) according to specific criteria. In MapXtreme a ranged theme modifies an existing layer to reflect that criteria. It does not create a new layer, as range themes did in previous versions of MapX and MapXtreme. When you create a ranged thematic map, MapXtreme groups all dataset rows into ranges and assigns each row’s object the color, symbol, or line for its corresponding range.
Chapter 15: Using Themes and Legends

When To Use a Ranged Theme

A ranged theme is useful, for example, when you have demographic data for an area. For example, rural male population in Asia can be grouped into bins and shaded by color to indicate the population ranges that exist across the area.

All records in the dataset are assigned to a range and then drawn with a style based on that range. When using a yellow to green color range, for example, the countries with the highest population could be shaded yellow, the lowest shaded green and the intermediate ranges some color in between yellow and green. When the map is displayed, the colors make it readily apparent where the highest and lowest populations exist. (See How to Apply Translucent Effects to Themes on page 411 for instructions on applying translucency effects to a ranged thematic map.)

Ranges are also useful when the size of the region is not directly related to the magnitude of the data values.

Types of Ranged Values

MapXtreme can create ranges automatically using five distribution methods:

- Equal count
- Equal ranges
- Standard Deviation
- Natural break
- Quantile
- Custom

Equal Count

Equal Count has the same number of records in each range. If you want to group 100 records into 4 ranges using Equal Count, MapXtreme computes the ranges so that approximately 25 records fall into each range, depending on the rounding factor you set.
When using Equal Count (or any other range method), it's important to watch out for any extreme data values that might affect your thematic map (in statistics, these values are referred to as outliers).

**Equal Ranges**

Equal Ranges divides records across ranges of equal size. For example, you have a field in your table with data values ranging from 1 to 100. You want to create a thematic map with four equal size ranges. MapX produces ranges 1–25, 26–50, 51–75, and 76–100. Keep in mind that MapXtreme may create ranges with no data records, depending on the distribution of your data.

**Standard Deviation**

When you create ranges using Standard Deviation the middle range breaks at the mean of your values, and the ranges above and below the middle range are one standard deviation above or below the mean.

**Natural Break**

Natural Break is a way to show data that is not evenly distributed. It creates ranges according to an algorithm that uses the average of each range to distribute the data more evenly across the ranges. It distributes the values so that the average of each range is as close as possible to each of the range values in that range. This ensures that the ranges are well-represented by their averages, and that data values within each of the ranges are fairly close together.

**Quantile**

Quantile is a second way to show data that is not evenly distributed. Quantiling uses two variables expressions. For example, use the Quantile distribution method to show the literacy rate as tied to population.

**Custom Ranges**

If none of the distribution methods meet your needs, you can create custom ranges using the method DistributionMethod.CustomRanges,. See the code example in the MapXtreme Developer Reference Help under the MapInfo.Thematics.RangedTheme.Recompute method.

VB example:

```vbnet
Public Shared Sub MapInfo_Mapping_Thematics_RangedTheme(ByVal map As Map)
' Create a ranged theme.
    Dim lyr As FeatureLayer = CType(map.Layers(0), FeatureLayer)

' Add the ranged theme to the layer.
    lyr.Modifiers.Append(theme)
End Sub
```
Chapter 15: Using Themes and Legends

RangedLabelTheme

This class creates a range theme in which labels are drawn with a range style. For a more detailed discussion of ranged themes, see the section RangedTheme on page 266.

When To Use a RangedLabelTheme Class

Ranged label themes are useful when you want use the labels to convey information about what you are labeling. For example, you could use a ranged label theme when labeling city or town populations. The label of a city with a large population would have a larger font than the label of a town with a small population.

VB example:

```vbnet
Public Shared Sub MapInfo_Mapping_Thematics_RangedLabelTheme(ByVal labelSource As MapInfo.Mapping.LabelSource, ByVal columnExpr As String, ByVal themealias As String)
    ' Create new ranged label theme based on the label source of a LabelLayer already in the map. It will use 5 bins of equal range.
    Dim rangedLabelTheme As RangedLabelTheme = New RangedLabelTheme(labelSource.Table, columnExpr, themealias, 5, DistributionMethod.EqualCountPerRange)
    ' Add the label modifier to the label layer.
    Dim labelModifier As MapInfo.Mapping.LabelModifier = CType(rangedLabelTheme, MapInfo.Mapping.LabelModifier)
    labelSource.Modifiers.Insert(0, labelModifier)
End Sub
```

IndividualValueTheme

An Individual Value theme is a modifier theme that show points, lines, or boundaries that are shaded by individual values contained in a particular field of a dataset. You can use both numerical and nominal values in individual values maps. MapXtreme gives each unique value its own distinct style.

For example, use an IndividualValue theme to show zoning classifications for parcels of land. Each zone (commercial, residential, industrial) would display in a different color. Parcels that match the zone classification would be shaded in that color.

When To Use an IndividualValueTheme Class

If you are shading your points, lines, or boundaries using nominal data, you can shade only by individual values. Nominal data is either non-numeric data (e.g., name, type of cuisine served, or brand of automobile sold) or numeric data where the numbers do not represent measurements. For example, nominal data may be a column containing ID numbers.

Dates are considered numeric data and can be used in both ranged and individual values maps.

VB example:
Public Shared Sub MapInfo_Mapping_Thematics_IndividualValueTheme(ByVal map As Map)

    ' Load a map based on one table
    map.Load(New MapTableLoader(“World.tab”))
    Dim fLyr As FeatureLayer = CType(map.Layers(“world”), FeatureLayer)

    ' Create an individual value theme
    Dim thm As IndividualValueTheme = New IndividualValueTheme(fLyr, “Country”, “World Pop”)

    ' Add the theme to the FeatureStyleModifiers list
    fLyr.Modifiers.Append(thm)
End Sub

### IndividualValueLabelTheme

This class creates an individual value thematic which operates on a layer's labels. For a more detailed discussion of individual value themes, see the section [IndividualValueTheme](#) on page 269.

### When To Use an IndividualValueLabelTheme Class

As with ranged label themes, individual value label themes are also useful when you want use the labels to convey information about what you are labeling. When working with street data, for example, you could use an individual value label theme to label different types of roads with different fonts. In this case, a highway would be represented by a label with a different style than the label for a county road.

VB example:

```vb
Public Shared Sub MapInfo_Mapping_Thematics_IndividualValueLabelTheme(ByVal labelSource As MapInfo.Mapping.LabelSource, ByVal columnExpr As String, ByVal themeAlias As String)
    ' Create new individual value label theme
    Dim theme As IndividualValueLabelTheme = New IndividualValueLabelTheme(labelSource.Table, columnExpr, themeAlias)

    ' Add the label modifier to the label layer.
    Dim labelModifier As MapInfo.Mapping.LabelModifier = CType(theme, MapInfo.Mapping.LabelModifier)
    labelSource.Modifiers.Insert(0, labelModifier)
End Sub
```
DotDensityTheme

A dot density theme is a style modifier that draws the fill pattern of a region, using dots based on the numeric value of the theme's expression.

Dot density maps use dots to represent the data value associated with a boundary or region. The total number of dots in a region represents that region's data value. If you have 10,000 senior citizens in a county, and each dot represents 100 senior citizens, there would be 100 dots in the county boundary.

When To Use a DotDensityTheme Class

A dot density theme is useful for showing raw data where one dot represents a large number of something: population, number of fast food restaurants, number of distributors who carry a brand of soda, etc.

For example, if you have a table of population broken down into county boundaries, you could use a dot density theme to show the concentration of people in each county boundary. There are two properties you control for dot density maps. You can specify the value of one dot. For example, to represent 20,000 high school students in Rensselaer County, New York using a dot density theme, you can specify that one dot represents 200 students. When you shade the county, the map would be drawn with 100 dots for that county.

VB example:

Public Shared Sub MapInfo_Mapping_Thematics_DotDensityTheme(ByVal map As Map)

' Load a map based on one table
map.Load(New MapTableLoader("World.tab"))

' Create a dot density theme.
' Add it as a modifier.
Dim lyr As FeatureLayer = map.Layers("World")
Dim thm As MapInfo.Mapping.Thematics.DotDensityTheme = New _
End Sub

Bivariate Thematic Maps

Bivariate thematic mapping uses point or line objects to represent two thematic variables. For example, a star can represent one variable, such as the number of teenagers, while a blue fill for the star represents their annual purchase amounts.

To create a bivariate map in MapX, you create two thematic maps, and layer one over the other so that the objects display two variables.
Types of Maps and Variables
The only types of thematic maps suitable for bivariate mapping are ranged and individual values maps. You can choose between two combinations for a bivariate map, depending on your data:

• two ranged maps
• one ranged map and one individual values map

If you have a non-numeric variable, one of your maps must be an individual values map. You cannot create a bivariate map with two non-numeric variables.

Displaying Attributes
To display two variables within one symbol, it is important to choose a different symbol attribute for each variable. For example, you cannot choose color for both variables because one color will overwrite the other. Choose from the following combinations:

• color and symbol type
• color and size
• size and symbol type

Symbol type should only be used for nominal or non-numeric data, as there is no inherent association between a symbol type and a quantity.

VB example:

```vbnet
Public Shared Sub MapInfo_Mapping_Thematics_RangedThemeConstructor(ByVal lyr As FeatureLayer)
    lyr.Modifiers.Append(thm)
End Sub
```

Legends Overview
The MapInfo.Mapping.Legends namespace contains classes, interfaces and enumerations for creating and displaying thematic and cartographic legends. Legends are a collection of Thematic or Cartographic LegendFrames. Each frame contains a collection of LegendRows. Each LegendRow has text and a style property.

Theme Legends
Theme legends provide a key of colors, symbols, and styles used for themes. This key explains what the colors, symbols, and styles represent.
When to Use a Theme Legend

A theme legend is useful whenever you have a map that contains themes. With a weather map that displays precipitation, rainfall may be represented by varying shades of green. A theme legend would be important in visually explaining that the darkest shade of green represents the highest amount of rainfall while lighter shades represent lower amounts of rainfall.

Cartographic Legends

The cartographic legend class allows you to read and write cartographic legend metadata. The legend identifies each cartographic feature on the map using text and a style from the metadata.

When to Use a Cartographic Legend

A cartographic legend is very useful whenever you have a map that contains objects that represent items on the map. For example, a map with landmarks requires a cartographic legend. Hospitals, schools, churches, and airports would each be represented by a different symbol. The cartographic legend provides a visual explanation of the different types of landmarks that are represented on the map.
VB example:

```vbnet
Public Shared Sub MapInfo_Mapping_Legends_LegendFrameFactoryCreateCartographic_LegendFrame()
    Dim Connection As MapInfo.Data.MIConnection = New _
    MapInfo.Data.MIConnection
    Connection.Open()
    Dim versionNumber As String = String.Format("{0}.{1}", _
    MapInfo.Engine.Session.Current.Catalog.OpenTable("C:\Program _
    Files\MapInfo\MapXtreme" & versionNumber & _
    "\Data\US\USA_CAPS.TAB")

    Dim tab As Table = Connection.Catalog.GetTable("USA_CAPS")

    Dim lyr As MapInfo.Mapping.FeatureLayer = New _
    MapInfo.Mapping.FeatureLayer(tab)

    Dim size As System.Drawing.Size = New System.Drawing.Size(400, 400)
    Dim legendfactory As MapInfo.Mapping.Legends.LegendFactory = _
    MapControl1.Map.Legends
    Dim legend As MapInfo.Mapping.Legends.Legend = _
    legendfactory.CreateLegend(size)

    Dim frame As MapInfo.Mapping.Legends.LegendFrame = __
    graphicLegendFrame(lyr, "State", "State", _
    MapInfo.Mapping.Legends.RowTextSourceType.Expression)
    legend.Frames.Append(frame)
    LegendControl1.Legend = legend
End Sub
```

Formatting a Legend

The MapInfo.Mapping.Legends.LegendFormat class contains properties that control how frames within a legend are drawn. You can control display properties such as alignment, the number of legend frames per row or column, spacing between frames, and whether the legend size and frame positions should be automatically adjusted.

The LegendFormat.FrameAlignment property is used in conjunction with the FramesPerRow for horizontal alignment and FramesPerColumn for vertical alignment.

For example, if your legend contains 10 frames, and you set the FrameAlignment to Horizontal and the FramesPerRow to 5, the legend will display two rows of frames with five frames in each row. If you had 10 frames per row, the 10 frames would display in a single row 10 frames wide.

Similar behavior happens with vertical alignment. If you have 10 frames and you set the FrameAlignment to Vertical and the FramesPerColumn to 5, you will end up with 5 rows of 2 frames (5 rows in 2 columns). The frames were aligned vertically up to 5 frames per column. When FramesPerColumn is set to 10, the legend would contain 10 rows of 1 frame each (10 frames per column).
The default setting for FramesPerRow and FramesPerColumn is 0. The value used is the current number of frames in the row or column as indicated by LegendFrameRows.Count or LegendFrameColumns.Count properties.
Stylizing Your Maps

Styles in MapXtreme affect many components of a mapping application, not just how a map feature looks. Styles are used for labels, text, themes, legends, selection and presentation with many controllable attributes so you can design practically any style you like.

This chapter discusses styles in terms of the MapXtreme framework, specifically the MapInfo.Styles namespace.

In this chapter:

- Overview of the MapInfo.Styles Namespace .............. 278
- Style Descriptions ........................................ 279
- Pre-defined Styles and the StyleRepository Class ........ 282
- Using Styles ............................................. 283
- Overriding Styles ....................................... 284
Overview of the MapInfo.Styles Namespace

The MapInfo.Styles namespace highlights the new Styles object model for MapXtreme. The Style class is the base class of all styles. Classes that derive from Style include AreaStyle, BaseLineStyle, BaseInterior, BasePointStyle, CompositeStyle, Font, RasterStyle, and TextStyle. Derived from BaseLineStyle is the SimpleLineStyle class. Derived from BaseInterior is the SimpleInterior class. Derived from BasePointStyle are the BitmapPointStyle, FontPointStyle, and SimpleVectorPointStyle classes. Additionally, GridStyle is derived from RasterStyle to include grid specific style settings. A StockStyles class is provided to create common style types.

You cannot instantiate the abstract style classes Style, BaseLineStyle, BaseInterior or BasePointStyle. You must create a particular type like SimpleLineStyle, or create them as a CompositeStyle.

Style is its own object; it is no longer stored in an object. Each table containing a geometry column also contains a Style column (alias MI_Style) containing a datatype Style. For style overrides, the Feature class provides FeatureStyleModifier and FeatureOverrideStyleModifiers. The Style object model also makes available several collection classes (style repositories) to hold styles for the style dialog controls.

All style classes support the ability to be applied in a sparse manner. See the section on FeatureOverrideStyleModifiers.

![Diagram of Style Class Hierarchy]

Styles are used in many areas of MapXtreme including the lines, interior fills and point styles that represent geographic features. Styles are also an integral part of labels, text, layouts, themes, overrides, legends and selections. Style properties range from standard color fills, line width and point size, to background effects, bitmaps as point styles, and rotation angle. Practically any style property you can imagine is available for you to incorporate into your application. Styles can be changed globally or per feature, overridden for the current display or made as a permanent change.

MapXtreme comes with a number of sample styles to get you started. There are more than 170 interior fill patterns, approximately 120 line style patterns, and approximately 70 bitmap point style images. You can create bitmap images in any application that can create bitmaps, like MS Paint or
Paint Shop Pro. There are virtually no size limitations on the image; however, the ability of MapXtreme to display it will depend on available memory. The image does not have to be square and can also have up to 24-bit color depth. To make sure the image is displayed at the height and width you want, select the 'Display at actual size' option for BitmapPointStyles. Once the image is created, place it in the CustSymb directory. Custom symbols are located in C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x\CustSymb folder.

Additionally, MapXtreme ships with style controls and dialogs that allow you to rapidly add them to a form or web application. See the ChangeStyles and FeatureStyles sample applications under the Samples folder which bring together all the style classes discussed in this chapter. For more information on form style controls and dialogs see Chapter 8: Desktop Applications, Controls, Dialogs, and Tools. For more on web controls see Chapter 5: Web Applications, Controls, and Tools.

For visual representations of supported style elements, see Appendix F: Style Lookups.

**StyleFactory**

A StyleFactory class is available from MapInfo.Styles that contains methods to generate MapInfo Style objects from various types of style parameters. For example, MapInfo.Styles.StyleFactory.FromMBstring takes a MapBasic style clause as string input and returns a CompositeStyle. See the online Developer Reference for more information and a code example.

**Style Descriptions**

**AreaStyle**

The AreaStyle class contains style properties used for drawing regions. Regions are drawn using a BaseLineStyle and BaseInterior.

**BitmapPointStyle**

The MapInfo BitmapPointStyle class contains style properties for drawing points using custom bitmaps. Use this class for marking point locations. BitmapPointStyle is one of three types of supported point styles, the others being FontPointStyle and SimpleVectorPointStyle.

BitmapPointStyle has a ShowWhiteBackground property; if set to false, any white pixel in the bitmap is transparent. By default, ShowWhiteBackground is set to false. For example, you may wish to use your corporate logo to represent office locations worldwide, but do not want to cover up other map features in the immediate area. There are other settings that control how BitmapPointStyles are displayed. See the BitmapStyles enumeration in the online help for more details.

MapXtreme ships with a number of bitmap point styles to get you started. They are located in the CustSymb directory. You can also add your own bitmap images to this directory. The maximum number of images allowed is 32,767.
CompositeStyle

The CompositeStyle class encompasses the collection of all the style types used for default styles, modifier styles, and layer override styles. The CompositeStyle can also be used to describe styles for the Collection object type. The style types contained are AreaStyle, a BaseLineStyle derived class, TextStyle, a BasePointStyle derived class, RasterStyle and GridStyle. The CompositeStyle can be constructed with any or all of these types, but must contain at least one of the above.

For example, you can create a style override (a FeatureOverrideStyleModifier) to alter the appearance of all features in a layer. Since a single layer can contain points and lines and regions, you might need to specify point, line and area styles when you build your style override. You can specify all necessary style types in one CompositeStyle object, which you then pass to the FeatureOverrideStyleModifier constructor.

SimpleInterior

The MapInfo SimpleInterior class contains style properties used for filling the interior of regions. SimpleInterior attributes include pattern, foreground and background color, and background transparency. The default for a SimpleInterior is a solid white interior.

Font

The MapInfo Font class contains style properties used for drawing text. Attributes for fonts, include bold, italic, underline, strikeout, shadow, halo, all caps, double space, size, foreground and background color. You can also change the font type (e.g., Arial, Times New Roman) and the font size. Note that the outline font property has been removed. To create an outline, use halo with a black background.

FontPointStyle

The FontPointStyle class contains style properties used for drawing points using mainly the MapInfo.Styles.Font class. You can customize the point size, font color, angle of rotation and other font properties. The maximum point size allowed is 240 points.

GridStyle

This is a helper class containing display style information about a grid such as color inflections, null cell color and transparency. A grid is a map of continuous color gradation that represents interpolated data values. For more information on grids see Chapter 18: Working with Rasters and Grids.

RasterStyle

This is a helper class containing display style information for a raster image including brightness, contrast, grayscale (color on/off), transparency, and translucency. For more information on raster images see Chapter 18: Working with Rasters and Grids.
Hillshade

This is a helper class used to store the parameters for hill shading on a grid. Hill shading, also called relief shading, can be added to grid maps to show the effect of a light source on the map. This gives a grid map greater definition, particularly useful for elevation maps. Hill shade properties include the horizontal and vertical angles of the light source, and a vertical scale factor. For more information on grids see Chapter 18: Working with Rasters and Grids.

Inflection

This is used to hold a single inflection point which associates a color with a value. A grid has an array of inflections to represent its colors. A grid map is a map that shows a gradual color change across an area. The blending of one color into the next is due to the inflection. For more information on grids see Chapter 18: Working with Rasters and Grids.

SimpleLineStyle

The SimpleLineStyle class contains style properties used for drawing polylines based on the MapBasic Pen clause. It is used for map features such as streets and cable lines, as well as borders around regions. Attributes to describe SimpleLineStyles include pattern, width (in pixels or points) and color. The default for a SimpleLineStyle is a solid black 1-pixel wide line. Units for SimpleLineStyle are pixels (default) or points.

The LineWidth class in the MapInfo.Styles namespace is a helper class used to define the width and units of a line style.

BasePointStyle

This is an abstract base class for all MapInfo point styles. It cannot be instantiated. SimpleVectorPointStyle, BitmapPointStyle and FontPointStyle derive from this class.

BaseLineStyle

This is an abstract base class for all MapInfo line styles. It cannot be instantiated. SimpleLineStyle derives from this class.

BaseInterior

This is an abstract base class for all MapInfo interior styles. It cannot be instantiated. SimpleInterior derives from this class.

StockStyles

This class contains static methods to create various default style objects, including black, blue, red and white interiors, black, blue, and red lines, hollow interiors and lines, and default fonts and point styles.

SimpleLineStyle redLine = StockStyles.RedLineStyle( );
Pre-defined Styles and the StyleRepository Class

TextStyle

This class contains style properties used for drawing text. It contains a MapInfo.Styles.Font class and a BaseLineStyle derived class for callout lines. The BaseLineStyle is optional (the TextStyle may or may not contain one).

SimpleVectorPointStyle

This class contains style properties for drawing points using MapInfo's 3.0 Compatible proprietary font (MapInfow.fnt ships with MapXtreme). SimpleVectorPointStyle properties include color, point size and the shape code of the actual symbol you wish drawn for the point. The standard set includes symbols 31 through 67.

Note Another symbol font set, called MapInfo Symbol, is a TrueType font set that displays using the FontPointStyle class.

Pre-defined Styles and the StyleRepository Class

MapXtreme ships with a variety of bitmap images covering many themes that can be used as bitmap point styles. More than 170 fill patterns and line styles are also available for use. These are all installed by the application and accessible either through the style dialogs (e.g., LineStyleDlg) or through the various StyleRepository classes.

For visual representations of supported style elements, see Appendix F: Style Lookups.

StyleRepository Class

The StyleRepository class contains style collection classes (VectorSymbolRepository, BitmapSymbolRepository, LineStyleRepository, InteriorStyleRepository), which allows you to iterate through all current styles, as well as reload the collections with new styles from a specific file(s) or directory. The StyleRepository class also contains a repository (TrueTypeFontInfoRepository) that represents information about the TrueType fonts installed on the system.

The VectorSymbolRepository represents the set of symbols from the MapInfo 3.0 Compatible symbol set. The BitmapSymbolRepository represents the set of images currently found in the CustSymb directory. The LineStyleRepository represents the current set of patterns available for line styles. And the InteriorStyleRepository represents the set of interior patterns available.
Using Styles

Styles and Layer Control

Use LayerControl at design or runtime to modify and override styles. See the sample application `LayerControl` for an introduction to the `MapInfo.Windows.Controls.LayerControl` object model. This code sample uses a `LayerControl` object directly on a form; it does not demonstrate using the `LayerControlDlg` dialog box, which is a related, but separate class. Every operation demonstrated in this code sample could be applied to a `LayerControlDlg` object, as well, since the `LayerControlDlg` class exposes a `LayerControl` property.

LayerControl is discussed in Chapter 8: Desktop Applications, Controls, Dialogs, and Tools and in Chapter 23: Workspace Manager.

Creating a Custom Bitmap Style

You can create bitmap images in any application that can create bitmaps, like MS Paint or Paint Shop Pro. There are virtually no size limitations on the image; however, the ability of MapXtreme to display it will depend on available memory. The image does not have to be square and can also have up to 24-bit color depth. To make sure the image is displayed at the height and width you want, select the 'Display at actual size option' for BitmapPointStyles. Once the image is created, place it in the CustSymb directory.
Overriding Styles

Styles can be permanently changed for features by saving the new style to the table. Styles for features can also be changed by overriding the current style with another. This does not change the style permanently. It only changes it for the current display. For example, a ranged theme overrides the style of a region object to shade it. See Using Themes and Legends in Chapter 15 on page 261 for more information.

Style can also be overridden for labels. This section introduces you to the main style override class for features. See more about features and labels in the chapter on the Mapping namespace Layers on page 233 and Labels on page 237.

FeatureOverrideStyleModifiers

This class implements FeatureStyleModifier to provide override style functionality for features. Its Style property is a composite style object that is used to specify what parts of a feature's style to override.

The contents of the style object passed to the Modify() method change dynamically for each feature that is drawn. This increases the drawing speed of layers that contain style modifiers. It is therefore important to make a copy of the style object if you need to use it elsewhere in your application. You should also note that for the CompositeStyles in the Style Stack passed to the FeatureStyleModifier.Modify() method, the Changed event does not fire.

Code Sample: FeatureOverrideStyleModifier

The following sample demonstrates how to use FeatureOverrideStyleModifier and layer FeatureStyleModifiers to change styles of various features within a map.

In this snippet from the \Samples\VisualStudio200x\Desktop\Features\ChangeStyles sample application, we want to override the world capitals layer with a single red symbol, but not change the point size.

VB example:

'Get the layer we want
Dim _lyr As FeatureLayer = Me.mapControll.Map.Layers("worldcap")

'Create a sparse point style
Dim vs As MapInfo.Styles.SimpleVectorPointStyle = New _
SimpleVectorPointStyle

'Just change the color and code and attributes flag to indicate that
vs.Code = 55
vs.PointSize = 25

' And apply to the layer
Dim fsm As FeatureOverrideStyleModifier = New _
FeatureOverrideStyleModifier(Nothing, New _
MapInfo.Styles.CompositeStyle(vs))
_lyr.Modifiers.Append(fsm)
Me.mapControl1.Map.Zoom = New MapInfo.Geometry.Distance(6250, _
MapInfo.Geometry.DistanceUnit.Mile)
End Sub
Spatial Objects and Coordinate Systems

This chapter covers the MapXtreme geometry namespace and provides descriptions and examples for writing applications for creating and manipulating geometry objects.

In this chapter:

- Introduction to MapInfo.Geometry Namespace ............. 288
- Geometries .................................................. 288
- Including Your FeatureGeometry in a Map ............. 294
- Checking for Points in Polygons .......................... 296
- Coordinate Systems ....................................... 297
Introduction to MapInfo.Geometry Namespace

The MapInfo.Geometry namespace is used for creating and manipulating geometry objects, and the coordinate systems in which they are used. Geometry objects are used in maps to represent single points, such as cities (represented as point objects), boundary lines, such as county borders (represented by MultiCurve objects), and regions, such as countries or zip code areas (represented by a MultiPolygon object).

The classes, interfaces, and enumerations in the MapInfo.Geometry namespace define the types representing the geometries and coordinate systems used in displaying geographic features on a map. Interfaces allow for creation and editing of the geometry objects. Methods such as Buffer, Combine, Difference, and Intersection provide object processing on single objects or pairs of objects.

Geometries

The Geometry class allows for the creation, editing, and other manipulation of geometry objects. Classes which inherit from the Geometry class and represent types of Geometry objects include Point, MultiPoint, Polygon, MultiPolygon, Curve, CurveSegment, LineString, and Ring. The following legacy classes are also inherited from the Geometry class: Rectangle, RoundedRectangle, Ellipse, LegacyArc, and LegacyText.

The Geometry class represents the topmost level of the MapInfo Geometry object model. This class is abstract, and cannot be instantiated. All classes that derive from this class contain knowledge concerning their coordinate system. All classes are able to make copies of themselves, and compare themselves to other Geometry objects for equality.
Chapter 17: Spatial Objects and Coordinate Systems

The diagram below shows a UML representation of the new Geometry model.

Geometry Objects

All geometry objects in MapXtreme are created with a specific coordinate system that cannot be changed. If you need to alter the coordinate system of an object you can make a copy of that object in the new coordinate system.

All Geometry objects contain a method for retrieving an interface to an editor that places the object into Edit Mode. Once editing is finished the EditingComplete() method needs to be called to signify that the editing of the object is complete. When the EditingComplete() method is called, the order of the objects contained by the Geometry is reshuffled and all references to them are dropped and need to be re-established in order to access them again.

For example, the user creates a MultiPolygon and then edits the MultiPolygon. If the user inadvertently moves a node of the interior ring to be outside of its containing Polygon the Polygon is no longer valid. When EditComplete is called, all the contained objects within the MultiPolygon are reshuffled, fixing the problem.

The geometry objects in the MapXtreme Object Model are described in the following sections.

FeatureGeometry Objects

The FeatureGeometry class is specifically designed to contain classes that can be placed in tables and that can be parts of Features and FeatureCollections. In order for something to be displayed in a map, it needs to be in a table. FeatureGeometry objects are by definition included in tables. Any
object that is a subclass of Geometry and not a subclass of FeatureGeometry cannot be saved to a
table or included as part of a Feature or FeatureCollection. An exception is thrown, or the program
will not compile, if such an operation is attempted. The Geometry class, like the FeatureGeometry
class is abstract and cannot be instantiated.

Point
Points are derived from the FeatureGeometry class and represent a single point on a map. Points
can be contained within a MultiPoint collection and then operated upon collectively.

Use the following example code to model the creation of a Point:

```csharp
using MapInfo.Geometry;
using Mapinfo.Design.Windows;

CoordSys longLatNad83;
CoordSysFactory coordSysFactory = new CoordSysFactory();
longLatNad83 = coordSysFactory.CreateLongLat
    (MapInfo.Geometry.DatumID.NAD83);
DPoint point = new DPoint(0.0, 0.0);
Point pointGeometry = new Point(LongLatNad83, point);
```

MultiPoint
A MultiPoint contains an unordered, disconnected set of Points and is useful for performing multiple
operations on multiple points.

Use the following example code to model the creation of a MultiPoint object:

```csharp
using MapInfo.Geometry;

CoordSys longLatNad83;
CoordSysFactory coordSysFactory = new CoordSysFactory();
longLatNad83 = coordSysFactory.CreateLongLat
    (MapInfo.Geometry.DatumID.NAD83);
MultiPoint multiPointGeometry = new MultiPoint
    (longLatNad83, pointArray);
```

where `pointArray` is an array of DPoints.

MultiCurve
The MultiCurve class is derived from the FeatureGeometry class, and contains a possibly
disconnected set of Curves. These Curves may interact in many ways; they can be connected or
disconnected, and can intersect or overlap each other.

Although the Geometry Object Model supports multiple CurveSegments for each Curve, the current
version of the MapInfo engine is limited to having one CurveSegment per Curve that is part of a
FeatureGeometry (i.e., MultiCurve). This limitation derives from the current TAB file format, which
remains largely unchanged for this version of MapInfo. Hence, the limitation concerns
FeatureGeometry objects only.
Upon construction of a MultiCurve where the constructor takes a Curve or Curves which may contain multiple CurveSegments per Curve, the actual Curves contained in the constructed MultiCurve are altered to always contain only one CurveSegment per Curve. Currently, the only types of CurveSegments that exist are LineStrings. Curves containing multiple LineString CurveSegments have the LineStrings combined to form one large LineString.

Upon completion of editing (signified by calling EditingComplete()), any Curve which was added to the MultiCurve and contained multiple CurveSegments is altered in a similar manner as noted above to produce Curves containing single CurveSegments. This limitation, of Curves contained in MultiCurves always containing only a single CurveSegment, should be removed in future versions of MapInfo as new types of CurveSegments are introduced (e.g., EllipticalArcs, CircularArcs, and Splines), and the TAB file format is altered. Also, during construction and on completion of editing, any empty Curves are automatically removed from the MultiCurve.

The order of the Curves in a MultiCurve may be altered during construction, as compared to the array of Curves passed to the constructor, and upon completion of editing. Due to this, plus the removal of empty Curves, and limitations in the current implementation, any references to Curves contained in a MultiCurve prior to and during editing may no longer be valid after editing is completed (i.e., after EditingComplete() is called). If these objects are referenced, they throw an ObjectDisposedException. After editing, the parts of a FeatureGeometry should be reacquired to obtain a valid reference.

Line objects made up of two points that exist in MapInfo TAB files become MultiCurve FeatureGeometry objects. They can be detected as two-point Lines by using the IsLegacyLine property of the MultiCurve:

See the Developer Reference for a code example of creating and editing a MultiCurve object.

**LineStrings**

A LineString is a directed collection of sequential points that are connected in a linear manner. Any two consecutive points in the LineString are connected by a straight line. LineStrings can be part of Curves or Rings, or they can exist as a stand-alone Geometry. LineStrings that are part of Curves or Rings inherit the coordinate system of their container. Stand-alone LineStrings can be empty. A LineString that is contained in a Curve or Ring that is not in Edit Mode cannot be empty, and must contain at least two points.

See the Developer Reference for a code example.

**Rectangle**

A Rectangle Geometry contains two points representing the lower left hand and upper right hand corners of the Rectangle. The other two points are implied. Rectangles are always axis aligned, and always appear to be rectangular in shape, regardless of the coordinate system, and are not projected. They do not contain any warping that may be represented by the coordinate system.

See the Developer Reference for a code example.
**RoundedRectangle**

A RoundedRectangle behaves exactly like a Rectangle but is displayed with the corners appearing rounded as a display-time only feature. The corners display as quarter circles and the radius of the circle is controlled by the CornerRadius parameter.

Because RoundedRectangle objects, like rectangle objects, are defined by two points and always display axis-aligned and unprojected, they are designed to be used primarily for cosmetic display purposes. While many operations are available using Rectangle objects (e.g., Combine), internally, a MultiPolygon copy of the Rectangle is used for these operations. The resulting MultiPolygon contains 5 points (with the first and last points being identical), and are effected by the coordinate system. In some instances, the converted Rectangle may no longer appear rectangular. Use the CreateMultiPolygon method to convert a RoundedRectangle to a FeatureGeometry object.

See the Developer Reference for a code example.

**Ellipse**

The Ellipse is inscribed in an axis-aligned rectangle defined by a DRect. The DRect is defined by two points, the opposite corners of a rectangle, with the other two corners of the rectangle implied. The Ellipse displays as if it were unprojected, regardless of the coordinate system, and any skew that may be represented by the coordinate system.

Because Ellipse objects are defined by two points and always display axis-aligned and unprojected, they are designed to be used primarily for cosmetic display purposes. While many operations are available using Ellipse objects, internally, a MultiPolygon copy of the Ellipse is used for these operations. The resulting MultiPolygon is effected by the coordinate system and in some cases may no longer appear as a perfect ellipse.

See the Developer Reference for a code example.

**LegacyArc**

The LegacyArc object is a portion of an Ellipse and is defined by a DRect, a start angle, and an end angle. The Ellipse is constructed to be inscribed in the rectangle defined by the DRect. The rectangle, in which the Ellipse is inscribed, is axis-aligned and is always rectangular regardless of the coordinate system used. The angles are measured in degrees with zero being along the positive X-axis and positive angles being in the counterclockwise direction. The angles are only stored to a tenth of a degree resolution with values between 0.0 and 360.0.

Because LegacyArc objects are defined by two points (for the DRect) and angles, and are always displayed axis aligned, they are designed to be used primarily for cosmetic display purposes. While many operations are available using LegacyArc objects, internally, a MultiCurve copy of the LegacyArc is used for these operations. This can sometimes lead to unexpected results.

See the Developer Reference for a code example.
LegacyText

The LegacyText object is the MapInfo Professional equivalent of a text object. If a given database does not support Text the LegacyText object can be lost when using that format. LegacyText objects are placed within a geographically-sized rectangle with a lower-left anchor point specified. The point-size of the text is based upon what fits best within the rectangle.

LegacyText objects do not fit nicely into the Geometry model. Several methods available on the Geometry FeatureGeometry classes, such as Combine, make no sense for LegacyText and will throw a NotSupportedException. Text objects do exist in MapInfo native TAB files in the Geometry column. The LegacyText class provides a way to access these objects. Refer to online reference for specific behaviors of LegacyText objects.

Geometry Objects

Geometry objects that are not also FeatureGeometry objects need to be converted to a suitable FeatureGeometry object to be displayed on a map. Most FeatureGeometry classes contain constructors that take appropriate Geometry objects and create new FeatureGeometry objects:

```csharp
using MapInfo.Geometry;

Curve curve = new Curve(csys, lineString);
MultiCurve multiCurve = new MultiCurve(curve.CoordSys, curve);
```

The code above creates the Curve using parameters defined elsewhere in the code of a CoordSys (csys) and a LineString (lineString). A new MultiCurve is then created using the CoordSys property of the Curve and the Curve itself.

In the example above, as in all FeatureGeometries created from objects, a copy of the original object is created because the reference cannot be shared.

Curve

The Curve class inherits from the CurveSegmentList class, and represents a contiguous linear Geometry. Curves contain a collection of CurveSegments that must remain contiguous. This class is included in the model to allow for future expansion and is part of the OGC standards.

Use the following example code to model the creation of a Curve:

```csharp
using MapInfo.Geometry;

DPoint[] points = new DPoint[4];

points[0]= new DPoint(-88.135215,43.998892);
points[1]= new DPoint(-104.875119,43.998892);
points[2]= new DPoint(-120.242895,47.048364);
points[3]= new DPoint(-89.135215 46.998892);

LineString lineString = new LineString(csys, points);
Curve curve = new Curve(csys, lineString);
```
Including Your FeatureGeometry in a Map

CurveSegments
At present a CurveSegment can only be a LineString. The class is designed to expand in future iterations of the product to include Spline, CircularArc, and EllipticalArc CurveSegments. Curves and Rings are comprised of CurveSegments.

Rings
A Ring is a collection of CurveSegments which must remain contiguous and closed.

Use the following example code to model the creation of a Ring:

```csharp
using MapInfo.Geometry;

dPoints = new DPoint[102];
dPoints[0] = new DPoint(-109.171279,49.214879);
dPoints[1] = new DPoint(-109.169283,49.241794);
...
dPoints[101] = new DPoint(-109.171279,49.214879);
Ring newRing = new Ring(longLatNad83, CurveSegmentType.Linear, dPoints);
```

Polygon
A Polygon is an object made up of Rings. A polygon must have at least a single Ring which defines the exterior boundary of the Polygon. Other Rings can be included inside which then define holes in the Polygon. Once a Ring is placed inside of another Ring the object becomes a MultiPolygon.

Use the following example code to model the creation of a Polygon.

```csharp
using MapInfo.Geometry;

DPoint[][] points = new DPoint[1][];
points[0] = polyPointArrays[0];
Polygon polygon = new Polygon
    (longLatNad83, CurveSegmentType.Linear, polyPointArrays[0]);
```

Including Your FeatureGeometry in a Map

Once a geometry is created you then need to add it to a map, allowing you to display it, select it, label it, or perform any other map-related operations on it.

```csharp
Public Shared Sub MapInfo_Mapping_HowDoICreateFeatureAddToMap(ByVal
    mapControl1 As MapControl, ByVal connection As MIConnection, ByVal x As Double, ByVal y As Double)
    Dim map As Map = mapControl1.Map
    'uses wldcty25 as a template
    Dim table As Table = _

    ' create a temp table and add a featurelayer for it
    Dim coordSys As CoordSys = map.GetDisplayCoordSys()
```
Dim tableInfo As TableInfoMemTable = New TableInfoMemTable("temp")
tableInfo.Temporary = True

' add Geometry column
Dim column As Column

' specify coordsys for object column
column = New GeometryColumn(coordSys)
column.Alias = "MI_Geometry"
column.DataType = MIDbType.FeatureGeometry
tableInfo.Columns.Add(column)

' add style column
column = New Column
column.Alias = "MI_Style"
column.DataType = MIDbType.Style
tableInfo.Columns.Add(column)

Dim pointTable As Table = _
Session.Current.Catalog.CreateTable(tableInfo)

' Set the location and display style of the point
Dim Geometry As FeatureGeometry = _
New MapInfo.Geometry.Point(coordSys, x, y)
Dim vStyle As SimpleVectorPointStyle = _
New SimpleVectorPointStyle(37, Color.Red, 14)
Dim cStyle As CompositeStyle = _
New MapInfo.Styles.CompositeStyle(vStyle)

' Update the table with the location and style of the new feature
Dim cmd As MICommand = connection.CreateCommand()
cmd.Parameters.Add("Geometry", MIDbType.FeatureGeometry)
cmd.Parameters.Add("style", MIDbType.Style)
cmd.CommandText = "Insert Into temp (MI_Geometry,MI_Style) values _ 
(Geometry,style)"
cmd.Prepare()
cmd.Parameters(0).Value = Geometry
cmd.Parameters(1).Value = cStyle
Dim nchanged As Integer = cmd.ExecuteNonQuery()
cmd.Dispose()

' add the table to the map
map.Layers.Add(New MapInfo.Mapping.FeatureLayer(pointTable))
End Sub
The following code example shows how to determine whether a point is inside the boundary of a FeatureGeometry (Multipolygon), on the boundary line or falls outside of it.

```vbscript
Public Shared Sub MapInfoGeometryContainsPoint()
    Dim coordSys As CoordSys = coordSysFactory.CreateLongLat(MapInfo.Geometry.DatumID.NAD83)

    Dim points(6) As DPoint
    points(0) = New DPoint(-0.705036, -0.122302)
    points(1) = New DPoint(-0.446043, 0.486811)
    points(2) = New DPoint(0.235012, 0.36211)
    points(3) = New DPoint(0.422062, -0.304556)
    points(4) = New DPoint(-0.244604, -0.71223)
    points(5) = New DPoint(-0.705036, -0.122302)

    Dim multiCurve As MultiCurve = New MultiCurve(coordSys,CurveSegmentType.Linear,points)
    Dim multiPolygon As MultiPolygon = New MultiPolygon(coordSys,CurveSegmentType.Linear,points)

    Dim insidePoint As DPoint = New DPoint(-0.115108,0.160671)
    Dim boundaryPoint As DPoint = New DPoint(-0.446043,0.486811)
    Dim outsidePoint As DPoint = New DPoint(-1.103118,0.021583)

    If multiPolygon.ContainsPoint(insidePoint) Then
        Console.WriteLine("Points inside area inclosed by closed (GeometryDimension 2) objects are contained")
    End If
    If Not multiCurve.ContainsPoint(insidePoint) Then
        Console.WriteLine("But this is not true for linear (GeometryDimension 1) objects")
    End If
    If multiPolygon.ContainsPoint(boundaryPoint) Then
        Console.WriteLine("Points on the boundary of closed objects are contained")
    End If
    If multiCurve.ContainsPoint(boundaryPoint) Then
        Console.WriteLine("Points lying on linear objects are contained")
    End If
    If Not multiPolygon.ContainsPoint(outsidePoint) Then
        Console.WriteLine("Point completely outside closed objects are not contained")
    End If
    If Not multiCurve.ContainsPoint(outsidePoint) Then
        Console.WriteLine("Point completely outside linear objects are not contained")
    End If
End Sub
```
Chapter 17: Spatial Objects and Coordinate Systems

Coordinate Systems

Coordinate systems describe the domain in which a particular object or set of objects reside. The coordinate system allows for the delineation, in specific terms, of the object or objects being described. The CoordSys classes contain methods, properties and interfaces that allow for the creation, manipulation, and editing of coordinate systems.

When Geometries are created, they are created in a particular coordinate system specified in the creation of the object. Objects cannot change the coordinate system in which they were created. They can only be copied into another coordinate system.

The CoordSys class facilitates the creation and manipulation of coordinate systems. The CoordSys class uses an XML version of the projection file (C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x\MapInfoCoordinateSystemSet.xml).

The CoordSysFactory object contains registered coordinate systems. CoordSys definitions can be registered by loading one or more XML projection files or by using the RegisterCoordSys, or RegisterCoordSysInfo methods. Create CoordSys objects from the factory, or code-codespace (EPSG, SRID), PRJ string, MapBasic string, and other Factory creation methods. There are also Military Grid Reference System conversion methods in the CoordSys class.

Creating a CoordSys Object

The following sample code shows the creation of CoordSys objects several different ways: using a MapInfo codespace; through EPSG; as longitude/latitude from a PRJ string; from a MapBasic string; and through SRID.

VB example:

```vbnet
Public Shared Sub MapInfoGeometryCreateCoordSys()
    Dim factory As CoordSysFactory = Session.Current.CoordSysFactory

    ' create CoordSys objects from srsName
    Dim csysWGS84 As CoordSys = factory.CreateCoordSys("EPSG:4326")
    Dim csysNAD83 As CoordSys = factory.CreateCoordSys("mapinfo:coordsys 1,74")
    Dim csysNAD27 As CoordSys = factory.CreateCoordSys("SRID:8260")

    ' create CoordSys objects from code/codeSpace
    csysWGS84 = factory.CreateCoordSys("4326", CodeSpace.Epsg)
    csysNAD83 = factory.CreateCoordSys("coordsys 1,74", CodeSpace.MapInfo)
    csysNAD27 = factory.CreateCoordSys("8260", CodeSpace.Srid)

    ' create CoordSys objects from user-defined parameters
    Dim dat As Datum = factory.CreateDatum(DatumID.WGS84)
    csysWGS84 = factory.CreateCoordSys(CoordSysType.LongLat, _
        dat, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, nothing)
    dat = factory.CreateDatum(DatumID.NAD83)
    csysNAD83 = factory.CreateCoordSys(CoordSysType.LongLat, _
        dat, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, nothing)
    dat = factory.CreateDatum(DatumID.NAD27ContinentalUS)
```
csysNAD27 = factory.CreateCoordSys(CoordSysType.LongLat, _
    dat, 0, 0, 0, 0, 0, 0, 0, 0, 0, nothing)
' create Long/Lat coordinate system
csysWGS84 = factory.CreateLongLat(DatumID.WGS84)
csysNAD83 = factory.CreateLongLat(DatumID.NAD83)
csysNAD27 = factory.CreateLongLat(DatumID.NAD27ContinentalUS)

' create from MapBasic string
Dim csysRGF93 As CoordSys = _
    factory.CreateFromMapBasicString("CoordSys Earth Projection 3, _
    33, ""m"", 3, 46.5, 44, 49, 700000, 6600000")
' create from PRJ string
csysNAD83 = factory.CreateFromPrjString("1, 74")
Sub

Changing the Coordinate System of a Geometry Object

The next example illustrates how to convert a Geometry object from one coordinate system to another.

VB example:

Public Shared Sub MapInfoGeometryCoordSys(ByRef coordSys As _
    CoordSys, ByRef points() As DPoint, ByRef alternateCoordSys _
    as CoordSys
' All Geometry constructors require a CoordSys parameter
' Note that the points array is assumed to be in coordSys
    Dim lineString As LineString = New _
        LineString(coordSys, points)

' The Geometry has a reference to the CoordSys used during
' construction. Unlike the coordinate data represented by the
' points array, the CoordSys' is not copied
    If ReferenceEquals(coordSys, lineString.CoordSys) Then
        Console.WriteLine("Geometry objects hold a reference to _
            the CoordSys used during construction")
    End If

' if you want to convert the object to another coordinate
' system, you need to make a new copy using one of the copy methods
    If Not coordSys.Equals(alternateCoordSys) Then
        Dim newGeometry as MapInfo.Geometry.Geometry = _
            lineString.Copy(alternateCoordSys)
    End If
End Sub
Determining the Coordinate System of a Map in MapControl

The following code examples shows how to determine the coordinate system for a Map object.

VB example:

```vbnet
Public Shared Sub __
    MapInfo_Mapping_FeatureViewerGetDisplayCoordSys(ByVal map As Map)
    'Load the Default Projection File so that we can get the name.

    'Get the Coordinate System object for current map in the MapControl
    Dim mapCoordSys As MapInfo.Geometry.CoordSys = __
        map.GetDisplayCoordSys()

    'Assign the name of the Coordinate System to a string variable.
    'note: the CoordSysName function will return a blank string if the
    'Coordinate System is not found in the current CoordinateSystemSet
    ' (loaded by the LoadDefaultProjectionFile above).
    Dim mapCoordSysName As String = __
    End Sub
```

Adding Coordinate Systems to MapXtreme

If the MapInfoCoordinateSystemSet.xml file does not contain a coordinate system to match your needs, you may add it to MapXtreme. This feature supports adding EPSG codes and SRID codes to extend MapXtreme's capabilities.

EPGS codes represent a collection of coordinate systems (known as codespaces) maintained in the EPSG Geodetic Parameter Dataset under the auspices of the International Association of Oil & Gas Producers (OPG). The OPG's Survey and Positioning Committee took over this responsibility from the European Petroleum Survey Group in 2005.

SRID codes are unique spatial reference numbers that refer to codespaces for Oracle Spatial tables. (MapXtreme supports a third codespace called MapInfo.)

MapXtreme provides you with many of the common EPSG and SRID mappings. If you need to register a different EPSG or SRID code to a particular coordinate system, this feature provides you with two methods to do so.

To extend MapXtreme's ability to use any EPSG or SRID codespace, you may add the information programatically, in which case, the coordinate system information will only last as long as the MapXtreme Session. Or you may add it to the Web.config file for your web application or app.config file for a desktop application as a more permanent solution. Each is discussed below.
Register EPSG and SRID Codes Programatically

The MapInfo.Geometry.CoordSysFactory class contains methods that allow you to register EPSG and SRID codes to a specified coordinate system.

RegisterEPSGCode() and RegisterSRIDCode() each take two parameters: one being the EPSG or SRID code that represents the codespace, the second is the coordinate system information that first parameter will map to.

The following example demonstrates registering a fictional code with the Long/Lat NAD83 coordinate system.

VB example:

```vbnet
Public Shared Sub MapInfo_Geometry_RegisterEPSGCode()
    Dim factory As CoordSysFactory = Session.Current.CoordSysFactory
    ' create CoordSys objects from srsName
    Dim csysNAD83 As CoordSys = factory.CreateCoordSys("mapinfo:coordsys 1,74")
    ' 9998 is a fictional code for demonstration purposes
    Try
        factory.RegisterEPSGCode(9998, csysNAD83)
    Catch ae As ApplicationException
        'code already exists. Codes cannot be duplicated
    End Try
End Sub
```

**Note** If the EPSG or SRID code already exists, an exception will be thrown indicating this fact.

To determine if a coordinate system for the MapInfo, EPSG or SRID codespace is already supported, call this method:


This method returns the first (or only) occurrence of the codespace that matches or null, if it does not exist.

Similarly, to return the first SRSName in the list that matches the input codespace, call this method:


An SRSName (Spatial Reference System) represents the name of a coordinate reference system written in GML (Geography Markup Language). This is typically, a friendly name for the coordinate system, not a list of parameter values.

To get a list of all the codes and coordinate systems that are mapped to a particular coordinate system, MapXtreme provides the following methods:

- MapInfo.Geometry.CoordSys.Codes(codeSpace)
- MapInfo.Geometry.CoordSys.SrsNames(codeSpace)

Keep in mind that the coordinate system information you added programmatically, will only be maintained during the lifetime of the MapXtreme Session.
Register EPSG and SRID Codes to a Web or Desktop Configuration File

The second, and more permanent, way to add EPSG or SRID codes to MapXtreme, is by adding the information to your web application’s Web.config file or your desktop application app.config file. The code below in bold shows the information you must copy and paste into your config file. An explanation of the code follows.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <configSections>
        <section name="MapInfo.CoreEngine"
            type="MapInfo.Engine.ConfigSectionHandler, MapInfo.CoreEngine,
            Version=4.0.0.468, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1" />
    </configSections>
    <appSettings>
        <add key="MapInfo.Engine.Session.UseCallContext" value="false"/>
    </appSettings>
    <MapInfo.CoreEngine>
        <EPSG_Code_Mappings>
            <EPSG_Code_Mapping>
                <srsName>My Custom CRS</srsName>
                <srsID>
                    <code>coordsys 8,74,8,-110.0833333333,47.5,0.9999375,2624666.667,328083.3333</code>
                    <codeSpace>mapinfo</codeSpace>
                    <remarks>My Custom CRS</remarks>
                </srsID>
                <EPSG_Codes>
                    <EPSG_Code>9987</EPSG_Code>
                    <EPSG_Code>9988</EPSG_Code>
                    <EPSG_Code>9989</EPSG_Code>
                </EPSG_Codes>
            </EPSG_Code_Mapping>
        </EPSG_Code_Mappings>
        <SRID_Code_Mappings>
            <SRID_Code_Mapping>
                <srsName>My Custom CRS</srsName>
                <srsID>
                    <code>coordsys 8,74,8,-114.0833333333,47.5,0.9999375,2624666.667,328083.3333</code>
                    <codeSpace>mapinfo</codeSpace>
                    <remarks>My Custom CRS</remarks>
                </srsID>
                <SRID_Codes>
                    <SRID_Code>9990</SRID_Code>
                    <SRID_Code>9991</SRID_Code>
                    <SRID_Code>9992</SRID_Code>
                </SRID_Codes>
            </SRID_Code_Mapping>
        </SRID_Code_Mappings>
    </MapInfo.CoreEngine>
</configuration>
```

1. If your desktop application does not have an app.config file, you can create one by adding it to your project from the Visual Studio Application Configuration File template.
The code above shows that there are two sections of information to add. One is to identify the correct CoreEngine dll and version number, the second is to add elements for EPSG and SRID code mappings.

An EPSG code mapping consists of an SRSName, SRS ID, and EPSG code(s). The SRSID is further defined by the parameters and codespace for the coordinate system.

An SRID code mapping is similar to the EPSG code mapping, except it refers to the Oracle Spatial identification number.

For more information on coordinate systems, see Appendix H: Elements of a Coordinate System. For information on the MapInfo codespace, see Appendix G: Defining the MapInfo Codespace.

National Transformation v. 2 (NTv2)

The National Transformation v. 2 (NTv2) algorithm and grid shift file format, developed by the Geodetic Survey Division of Geomatics Canada, enables you to convert data from the NAD 27 reference system to the NAD 84 reference system. Grid shift files used with the algorithm contain one or more rectangular grids that indicate the coordinate differences between NAD 27 and NAD 83.

The National Transformation was originally designed to convert Canadian data from NAD 27 to NAD 84. This format was later adopted for datum conversion in Australia, New Zealand, and Germany.

The NT v. 2 algorithm has been implemented in MapXtreme 6.7.1 to support these transformations. Grid shift files have also been added for the following countries:

- Canada
- Australia
- New Zealand
- Germany

These files are located in the MapXtreme Common Files folder. The default location of this file is:

C:\Program Files\Common Files\MapInfo\MapXtreme\6.7.1

In addition to the algorithm and grid shift files, an XML configuration file, NTv2.xml, has been installed to enable you to extend the NTv2 algorithm to support additional datum transformations. You can add new grid shift files to the configuration file, or you can turn the NTv2 algorithm on or off for particular grid shift files. The NTv2.xml file is located in the MapXtreme Common Files folder.

The NTv2 algorithm and grid shift files for Canada are protected under the following copyright:

---

1. To determine the correct version number for the MapInfo.CoreEngine assembly, from the Start menu, choose Run and type Assembly in the Run dialog. Every registered MapXtreme assembly is listed in the global assembly cache list.
The next sections describe the supported transformations for each country and how to use the configuration file.

**Canada**

The following grid shift files are included for Canada:

- NTV2_0.GSB—used for converting NAD 1927 to NAD 1983
- MAY76V20.GSB—used for converting NAD 1927 (Definition 1976) to NAD 1983 for Ontario

These files convert between the NAD 1927 or NAD 1927 (Definition 1976) datums and the NAD 1983 datum. The NAD 1927 datum (Definition 1976) coordinate system is a readjustment of NAD 1927 for Ontario.

Detailed information about the algorithm, software, and grid shift files can be downloaded from the Geodetic Survey Division of Geomatics Canada web site:

http://www.geod.nrcan.gc.ca/apps/ntv2/index_e.php

**Australia**

The following grid shift files are included for Australia:

- A66_National.gsb—used for converting AGD 1966 to GDA94.
- National_84.gsb—used for converting AGD 1984 to GDA94

The Australian grid shift files convert between either the AGD 1966 or AGD 1984 datums and the GDA 1994 datum.

Detailed information about conversion and grid shift files can be downloaded from the Intergovernmental Committee on Surveying and Mapping (ICSM) web site:


**New Zealand**

The following grid shift file is included for New Zealand:

- nzgd2kgrid0005.gsb—used for converting the NZGD49 datum to the NZGD2000 datum.

The New Zealand grid shift file converts from the NZGD49 datum to the NZGD2000 datum.

Detailed information and an online converter can be found at the Land Information New Zealand (LINZ) web site:


**Germany**

The following grid shift file is included for Germany:

- BETA2007.gsb—used for conversion from DHDN datum to ETRS89
For more information and to download the official grid shift file, see the Information and Service System for European Coordinate Reference Systems (CRS EU) web site:

http://crs.bkg.bund.de/crseu/crs/descrtrans/BeTA/de_dhdn2etrs_beta.php

Configuration File

The NTv2 configuration file (NTv2.xml) is an XML file that you can edit to add new grid shift files for other datum conversions and turn NTv2 on or off for particular grid shift files.

To add new grid shift files, you must specify three parameters:

- grid shift file name
- source datum
- destination datum

You can define a datum in various ways. For example, when both the source and destination datums are defined in MapXtreme, the datums are defined in the configuration file by the numbers assigned to them in the MapXtremeCoordinateSystemSet.xml file. In this example for New Zealand, the XML in the configuration file will look like this (application of bold text is for emphasis only):

```xml
- <NTv2Conversion>
  <Description>NTv2 Conversion for New Zealand</Description>
  <GridFile>NZGD2KGRID0005.GSB</GridFile>
  <Enabled>true</Enabled>
- <SourceDatum>
  <DatumID>1010</DatumID>
</SourceDatum>
- <DestinationDatum>
  <DatumID>117</DatumID>
</DestinationDatum>
</NTv2Conversion>
```

Sometimes, however, datums are not that easily defined. You may need to supply datum shift values (and sometimes even ranges for them) in order to define a datum. For example, the sample XML below shows the definition of the NAD 27 datum for Canada. Here, the ellipsoid, shift values for x,y,z, and their ranges have been used to define the datum (application of bold text is for emphasis only):

```xml
- <NTv2Conversion>
  <Description>NAD 1927 to NAD 1983 conversion for Canada</Description>
  <GridFile>NTV2_0.GSB</GridFile>
  <Enabled>true</Enabled>
- <SourceDatum>
  <EllipsoidID>7</EllipsoidID>
  <ShiftX>-8</ShiftX>
  <ShiftY>150.5</ShiftY>
  <ShiftZ>186</ShiftZ>
  <dShiftX>17</dShiftX>
  <dShiftY>19.5</dShiftY>
  <dShiftZ>8</dShiftZ>
</SourceDatum>
- <DestinationDatum>
```
<DatumID>74</DatumID>
</DestinationDatum>
</NTv2Conversion>
Working with Rasters and Grids

The MapInfo.Raster namespace contains all of the classes that control the use and display of raster and grid images in MapXtreme. Rasters are computer graphics that are composed of pixels that render a whole image. Many satellite images are rendered as raster images. Grid images are thematic maps that show a continuous gradation of color to represent interpolated information.

In this chapter:
- Overview of the MapInfo.Raster Namespace ........................................... .308
- Raster Images ............................................................................................ .308
- Raster Handlers ......................................................................................... .311
- Configuring Custom Raster Handlers ....................................................... .312
- Grid Images ............................................................................................... .313
Overview of the MapInfo.Raster Namespace

The MapInfo.Raster namespace exposes the full functionality of Pitney Bowes MapInfo’s C/C++ Raster and Grid Engine APIs. Raster images are bitmaps that provide useful background and reference layers to maps. Grid images are a type of theme that shows a continuous gradation of color across the image. The gradation represents an interpretation of the underlaying data. Grid images are rasters that have data associated with them. A common example is an elevation map. Rasters do not have any underlying data.

Raster Images

Raster images make excellent backgrounds for maps. For example, aerial photographs that show real-world detail such as buildings, refineries, and vegetation are well-suited as base layers for a map. Scanned paper maps are another example of a raster image. Use a raster image as a base layer and overlay vector data such as street networks, point locations representing customers, and postal boundaries, to create useful and visually appealing maps.

Raster images used with vector data must be registered so that known geographic points on the image coincide with the same features on the vector data. Additionally, company logos and other art you wish to display in MapXtreme must be registered to some location on earth, even though they are not true geo-referenced data. Many raster images available today come with a registration file. Examples include GeoTIFF, ADRG, ASRP, CADRГ, and CIB. To register a raster image, you can bring it into MapInfo Professional and register it there. The registration information is stored in a .TAB file.
Below are the raster image formats supported in MapXtreme:

- TIFF and GeoTIFF (*.tif)
- MrSID (*.sid)
- ECW (*.ecw)
- Spot (*.bil)
- JPEG (*.jpg)
- JPEG2000 (*.jp2, *.j2K)
- PCX (*.pcx)
- GIF (*.gif)
- Windows Bitmap (*.bmp)
- PNG (*.png)
- Photoshop (*.psd)
- Targa (*.tga)
- Windows Metafile (*.wmf)
- Windows Enhance Metafile (*.emf)
- Vertical Mapper Continuous Grid (*.grd)
- Vertical Mapper Classified Grid (*.grc)
- ADRG - ARC Digitized Raster Graphics (*.gen)
- ASRP - ARC Standard Raster Product (various file extensions)
- CADRG - Compressed ARC Digitized Raster Graphics (*.gen)
- CIB - Controlled Image Base (various file extensions)
- NITF - National Imagery Transmission Format (*.ntf)

Additional raster formats may be supported on a system if a custom raster handler is installed.

**Raster Classes**

The main classes for raster images are MapInfo.Raster.RasterInfo and RasterRead. Style information is handled by MapInfo.Style.RasterStyle.

RasterInfo provides information about the height and width of the image in pixels, raster format, color depth, and registration information. See the RasterInfo sample application provided in .\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features\RasterInfo

RasterRead is the class that reads the raster image and style information in order for the image to be rendered.

RasterStyle is concerned with how the raster looks. You can control brightness and contrast, display a color image as grayscale, set the transparency and translucency. MapXtreme supports transparency for one color per image. That means that everywhere that color exists in the image it will be invisible, which allows the layer below the image to show through. Translucency is the degree of transparency for the entire image. If you need a layer below a raster image to show through, set a high translucency (100 percent is transparent).

Raster columns are read only so you cannot permanently change their style. However you can programmatically set and get the image's attributes such as brightness, grayscale, and translucency. See the RasterStyle class in the online Developer Reference.
Raster Images and Coordinate Systems

When you display a raster, grid or WMS image as a map layer, MapXtreme automatically sets the rotation and projection of all the vector map layers, so that they match the rotation and projection of the raster image.

If a map includes more than one raster, grid or WMS image layer, MapXtreme automatically displays the map in the projection specified by the most visible raster image. The coordinate system could then change as the map view changes (due to zooming or panning) if a different image with a different projection becomes the most visible. In this case, you cannot change the map's display coordinate system.

Raster Reprojection

Raster reprojection enables you to change the cartographic projection of a raster layer in a map. Many types of raster images can be reoperated. Some examples include satellite and aerophoto images, scanned maps, as well as grids, seamless raster tables, and WMS data.

You can control the reprojection of both raster and vector layers. When you add either a raster or vector layer to a map, the new layer is reprojected into the current map window projection.

When you change the projection of a map window that contains a combination of vector and raster layers, all the layers, both raster and vector, can be reprojected to the new map window projection.

You can access raster reprojection settings in the MapXtreme API, as well as in the user interface of the Workspace Manager’s Layer Control. For information using raster reprojection in the API, see the MapXtreme Developer Reference. For information on the raster reprojection user interface, see Raster Reprojection in Chapter 23 on page 400.

Reprojected Images in Workspaces

The raster reprojection properties can be read into and from a workspace.

MapXtreme can also write raster reprojection information from MapInfo Professional to an .mws workspace. This capability enables you to create your workspace in MapInfo Professional, and then load it into MapXtreme.

Once your workspace is loaded into MapXtreme, you can modify the raster reprojection via the API or the through the user interface in the Workspace Manager’s Layer Control.

Note MapXtreme versions 6.7.1 and earlier ignore raster reprojection parameters.

Raster Image Limitations

- You cannot select any features a raster layer.
- You cannot search for features a raster layer.

Code Sample: Adding a Raster Image to a Map

Adding a raster image to a map is the same as adding any other layer.

C# example:
Table MyTable = Session.Current.Catalog.OpenTable("MyRaster.tab");
FeatureLayer MyLayer = new FeatureLayer(MyTable);
MyMap.Layers.Add(MyLayer);

VB example:
Dim MyTable As Table = _ Session.Current.Catalog.OpenTable("MyRaster.tab")
Dim MyLayer As FeatureLayer = New FeatureLayer(MyTable)
MyMap.Layers.Add(MyLayer)

Raster Handlers

MapXtreme can use one of many different libraries to load a raster image. When a raster image is loaded by MapXtreme, it searches through these DLLs and checks if the given file can be read by that DLL. Once a DLL/Raster format match is made, MapXtreme knows which DLL does the format handling for the file. The Format handlers are named “xxxxxxxx.RHx”. The base part of the name is based on the format. The extension always begins with RH, but can end in any letter (A-Z). When searching for a format handler, MapXtreme searches the format alphabetically, starting with RHA and continuing until RHZ. This process allows MapXtreme to prioritize which handlers are used. For example, SPOT files should be checked for before any other formats since they are just raw data that can be confused with other formats. The SPOT handler's extension is RHD. The Halo format handlers are named RHV. The LEADTOOLS format handlers are named RHX.

LEADTOOLS Win32 Pro provided by LEAD Technologies, Inc. and HALO Imaging libraries provided by Media Cybernetics are included with MapXtreme. LEADTOOLS loads the entire raster image into memory at the time the image is referenced in MapXtreme. This means that the image takes longer to load, but panning and zooming is faster. HALO only loads into memory what it needs to display, so it loads the image faster, but panning and zooming is slower. By default, HALO is attempted first. There may also be some formats which are not supported by either library and therefore need their own separate library. The raster handlers include within MapXtreme are bundled inside the MapInfo.CoreEngine.dll and so are not accessible or modifiable by the developer. Therefore, the order of the checking of the handlers is not able to be altered. For example, if checking a JPEG, which both Halo and LEADTOOLS can handle, Halo always prevails because of its *.rhv (over LEADTOOLS *.rhx) extension.

There are a few raster formats that we do not support in multi-threaded applications (e.g., in an ASP.NET application). These formats are as follows:

<table>
<thead>
<tr>
<th>Reading</th>
<th>Export</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical Mapper GRD, GRC</td>
<td>JPEG 2000 (Win2k only) from LeadTools</td>
</tr>
<tr>
<td>JPEG from Halo</td>
<td>TIFF CMYK from LeadTools</td>
</tr>
<tr>
<td>The LeadTools JPEG handler is threadsafe)</td>
<td></td>
</tr>
<tr>
<td>TIFF w/jpeg compression from Halo</td>
<td></td>
</tr>
<tr>
<td>The LeadTools TIFF handler is threadsafe</td>
<td></td>
</tr>
</tbody>
</table>
Configuring Custom Raster Handlers

You can configure your MapXtreme application to use a different raster handler than the ones included in the distribution of MapXtreme or to support an entirely new raster type. You can also change the precedence in which raster handlers are used.

In the default installation of MapXtreme, all raster handlers are placed in `<program files\\Common Files\\MapInfo\\MapXtreme\\6.x.x\\RasterGridHandlers`. This is also the location of the file mirasteru.dll. This is the recommended installation location of any other raster handlers that you use in your application. If you use the default location, no other configuration steps are necessary.

If you want to put your custom raster handler in a non-default location, then you must specify the location of your custom raster handler in an application configuration file for a desktop application, or in the Web.config file for a web application. To do this, define a `<Path>` or a `<SpecialPath>` element under `<ApplicationDataPaths>`, then copy the selected raster handler to that folder.

For example, to configure a desktop application custom raster handler in the non-default MyAppData directory, you could use the following .config file.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="MapInfo.CoreEngine" 
      type="MapInfo.Engine.ConfigSectionHandler, MapInfo.CoreEngine, 
      Version=4.0.0.350, Culture=neutral, 
      PublicKeyToken=93e298a0f6b95eb1" />
  </configSections>
  <MapInfo.CoreEngine>
    <ApplicationDataPaths>
      <SpecialPath>
        <Personal>MyAppData</Personal>
      </SpecialPath>
      <Path>c:\MyAppData</Path>
    </ApplicationDataPaths>
  </MapInfo.CoreEngine>
</configuration>
```

In this example, the `<Personal>` tag is a special place defined in the user’s My Documents. This would refer to a folder called MyAppData in My Documents. Within the `<ApplicationDataPaths>` tag, use either the `<SpecialPath>` or the `<Path>` tags. Using both would mean that the raster handler could be put in either "MyAppData" folder. While this would not be an error, it probably is not what you intend to accomplish. Since this is a non-default configuration, you must also place the mirasteru.dll in that same directory.

Use the `<SpecialPath>` syntax if the application data is stored in a location relative to a .NET Framework special system folder. For example, if your application data is stored in a directory named MyAppData located under the “My Documents” directory, then the entry in the configuration file could be:

```xml
<Personal>MyAppData</Personal>
```

where “Personal” is the value of the .NET Framework enumeration Environment.SpecialFolder that represents the “My Documents” directory.
This configuration method can also be used to change the preferred raster handler for file types that can be managed by more than one raster handler. For example, a JPEG file can be handled by either Halo or LEADTOOLS (both of which are bundled in MapXtreme). Normally, Halo has priority because its *.rhv file extension alphabetically precedes the LEADTOOLS *.rhx extension. However, because MapXtreme first looks in any <ApplicationDataPaths> defined in the configuration file, it will locate a specified raster handler before looking for a handler in the default [CommonFiles] folder. So, for example, you could copy a LEADTOOLS *.RHX handler in the defined folder and configure MapXtreme to find and use that handler first.

Alternatively, you can rename file extensions in the [CommonFiles] folder so that the desired raster handler appears first in the alphabetic list. However, this will affect all applications developed with MapXtreme and may produce unintended side effects. Also, if file extensions are changed in this manner, the renamed raster handlers will not be deleted if MapXtreme is uninstalled. For these reasons, you may want to use the configuration method to change the location and precedence of raster handlers.

Grid Images

As stated previously, grid images show an interpolation of data values across an area. A grid image is created from a data file in which data is measured at evenly-spaced points. The entire map area is converted to a grid in which each grid cell represents a value. Grid values don't have to be interpolated to produce a grid, although the data collection points need to be regularly spaced. MapInfo Professional creates grids by interpolating values using a grid handler.

In addition to the color gradations, grid images can also show hill, or relief, shading. Relief shading allows the grid surface to be shaded according to a virtual light source. The brightness of each grid cell corresponds to the light striking the surface and is adjusted based on its orientation to the light source. This is well suited for elevation grid maps where you can take surface slope and direction into account relative to the direction of the light. Maximum brightness is assigned at points where the sun’s rays are perpendicular to the surface. As the slope faces turn away from the light source, the brightness values are lower.

Supported grid formats include:

- MapInfo Grid (*.mig)
- USGS DEM (*.dem)
- GTOPO30 (*.dem)
- DTED (*.dt0, *.dt1, *.dt2)
- Vertical Mapper Continuous Grids ( *.grd)

Additional grid formats may be supported on a system if a custom grid handler is installed.

1. Can be displayed as either grid or raster. The .TAB file determines whether the image should be drawn using the grid handler or the raster handler.
Grid Classes

The main classes for grid images are MapInfo.Raster.GridInfo, GridRead, ContinuousGridWrite and HillShadeWrite. GridInfo and GridRead are like RasterInfo and RasterRead, in this case, provide the ability to get information about a grid file. Grids are created for a layer of data points using ContinuousGridWrite. HillShadeWrite allows you to add or alter relief shading to the grid. Many of the properties of a grid are inherited from RasterInfo, including the image class, coordinate system, raster control points and minimum bounding rectangle (MBR).

Code Sample: Adding a Grid Image to a Map

Adding a grid image to a map is the same as adding any other layer.

C# example:

```csharp
Table MyTable = Session.Current.Catalog.OpenTable("MyGrid.tab");
FeatureLayer MyLayer = new FeatureLayer(MyTable);
MyMap.Layers.Add(MyLayer);
```

VB example:

```vbnet
Dim MyTable As Table = _ Session.Current.Catalog.OpenTable("MyGrid.tab")
Dim MyLayer As FeatureLayer = New FeatureLayer(MyTable)
MyMap.Layers.Add(MyLayer)
```

Code Sample: Retrieving Data from a Grid Map

This example shows how to open and read information from a grid file.

VB example:

```vbnet
Public Shared Sub MapInfo_Raster_GridRead(ByVal strGridFilename As String)
    Dim strHillshadeFilename As String = _
    MapInfo.Raster.GridRead.DefaultHillshadeFilename(strGridFilename)
    Dim session As ISession

    session = MapInfo.Engine.Session.Current
    Dim gridread As GridRead = New GridRead(strGridFilename, _
        strHillshadeFilename)
    Console.WriteLine(gridread)
End Sub
```

This example shows how to open and read cell values from a grid file.

VB example:

```vbnet
Public Shared Sub MapInfo_Raster_GridReadStartRead(ByVal gridread As GridRead, ByVal strGridFilename As String)
    Dim strHillshadeFilename As String = _
        gridread.DefaultHillshadeFilename(strGridFilename)

    If gridread.StartRead() Then
        Dim x As Integer = 0   ' TODO  - set to a pixel column value
        Dim y As Integer = 0   ' TODO  - set to a pixel row value
```
Dim bIsNull As Boolean
Dim dValue As Double
If gridread.GetValue(x, y, bIsNull, dValue) Then
    If bIsNull Then
        ' read a null cell
        Console.Write("{0,20}", "NULL")
    Else
        ' read a non-null cell, with value == dValue
        Console.Write("{0,20}", dValue)
    End If
End If
gridread.EndRead()
This chapter covers the MapXtreme namespaces for geocoding and provides descriptions and examples for writing applications that will access geocoding servers or services.

In this chapter:

- Overview of the MapInfo.Geocoding Namespace .............. 318
- Main Geocoding Classes. ...................................... 318
- Understanding the Geocoding Model. .......................... 320
- Geocoding a Location ........................................... 322
- Using Constraints for Accurate Geocoding .................. 325
- Understanding Accuracy for Close Matches ................. 328
Overview of the MapInfo.Geocoding Namespace

The MapInfo.Geocoding namespace provides interfaces and classes for geocoding address records. Geocoding is the process of determining the geographic location of a street address, intersection, place or postal code in order to pinpoint the location on a map. The geocoding client sends requests via HTTP to Pitney Bowes MapInfo’s geocoding server or geocoding web service product. The server and web service are not included in the purchase of MapXtreme.

Perhaps you are building a find the nearest application in which your users provide their geographic location, in the form of an address, and receive a map that shows where they are in relation to the closest ATM machine. To perform this task, you need more than an address to display a location on a map. These geographic locations (addresses, buildings, points of interest) translate to coordinates on the Earth’s surface. You need the Latitude/Longitude, or coordinate, values to pinpoint and map the location. The MapXtreme geocoding client gives you these values determined (geocoded) from a place name, street address, postal code, or from the intersection of two streets.

In addition to returning the coordinate values, the geocoding client can return a complete description of the place, which is useful when only partial information is known. The geocoding client may return zero, one, or more responses to a request depending on the request preferences, the method being employed, the specified preferences, and the match criteria.

The MapXtreme geocoding client also has Gazetteer type functionality (world/country/city Geocoder), that geocodes a partial address. For example, an address containing only a city, state, place name, landmark, or airport.

Using the geocoding capabilities, you get those coordinates by matching the input address to one that already has the coordinates. You supply one or more addresses in a single request document. Incomplete addresses can be processed as well. The server returns a complete address that you can use to clean up your input addresses. The server returns zero, one or more responses to a service request, depending on the quality of the data and the request preferences. Responses typically include a list of one or more potential matches, called candidates, and result codes that indicate the positional accuracy of the candidates. Your application will then need to resolve the response by choosing the appropriate address candidate to retrieve the coordinates.

Main Geocoding Classes

Geocoding is supported using either a MapInfo geocoding server or the Envinsa Location Utility Service. One of these resources needs to be deployed and made available via a URL. To create the appropriate geocoding client, use the GeocodeClientFactory class. The interface for both clients is similar as they both use the same classes for geocode requests, preferences, and responses.

The main interface for using the geocoding client is called MapInfo.Geocoding.IGeocodeClient. This interface defines a geocoding client object. It has a single method, Geocode, that takes a request as input and returns a response containing candidate addresses (addresses that are a potential match for the input address). Other classes include: GeocodeRequest, GeocodeResponse, GeocodeClientFactory, GeocodingConstraints, AddressCandidates, GeocodeMatchCode, and CandidateAddress.
The following diagram illustrates the interfaces and classes that make up the Geocoding namespace:

**GeocodeRequest**

The GeocodeRequest class sends a request to the geocoding server or service. Its properties include AddressList, a list of addresses to be geocoded, and Length, the number of items in the address list. Input addresses can include a variety of information, including street address or intersection, primary and secondary postal code, and country code.

**GeocodeResponse**

The GeocodeResponse class contains a response for each address in the GeocodeRequest. The response contains address candidates, which are those addresses considered to be possible matches for the input address. Note that a GeocodeRequest that contains multiple addresses will yield a GeocodeResponse object that includes a list of candidate addresses for each input address.

**GeocodeClientFactory**

This class returns an IGeocodeClient which you use to send a GeocodeRequest and receive a GeocodeResponse. The method GetMmjHttpClient uses the geocoding client that speaks to an instance of the MapInfo MapMarker Java servlet. Use the method GetEnvinsaGeocodeClient to send requests to the Location Utility Service in MapInfo Envinsa 4.0.
Understanding the Geocoding Model

GeocodingConstraints

This class contains the preferences that can be set when geocoding. All are set/get properties that return true or false (default).

AddressCandidates

This class is a list of possible address matches that is returned in a GeocodeResponse.

BaseGeocodeMatchCode and GeocodeMatchCode

These are classes that implement the IGeocodeMatchCode interface. The interface exposes a ResultCode property that is a string that describes how well the match was made. BaseGeocodeMatchCode is returned when a request is sent to the Envinsa Location Utility Service. GeocodeMatchCode extends BaseGeocodeMatchCode. It is returned when a geocode request is sent to a MapMarker server. GeocodeMatchCode includes additional convenience properties to determine which parts of the address matched. Among them, StreetAddressMatch and MunicipalityMatch return a value of true if the candidate matched the street and municipality. For details on result codes see Understanding Accuracy for Close Matches on page 328.

CandidateAddress

This class defines an address that was geocoded. Its properties include Address, the address for the geocoded address, GeocodeMatchCode, which explains how well the address was geocoded, and Point, the geometry that represents the candidate address.

Understanding the Geocoding Model

The geocoding client is based on a model of relative matching that is governed by a set of weights that scores each portion of the address against candidate records (possible matches) in the data. The resulting scores are summed and the candidate’s total score is used to determine the best match or matches. A close match is made when there is a candidate that scores well above other candidates. In addition, the matching routine uses a set of geocoding preferences to determine whether certain matching conditions are required or relaxed. For example, the default preferences include relaxing a match on postal code, but requiring a match on the house number and street name. This gives the best return of hit rate with the fewest erroneous matches (false positives) and the best performance.

Geocoding Trade-offs

With a relative matching system such as the geocoding client, there are trade-offs that must be considered in light of how you use the geocoded data. Consider questions like the following:

- What level of matching accuracy are you looking for (unique address match, close match)?
- What level of geographic accuracy do you need for your geocoded points (street level, ZIP Code centroid)?
• Is your goal to geocode as many records as possible?

The answers to these questions are driven by how you intend to use the geocoded records. For example, perhaps you are determining the location of a new retail store and need to know the distribution of current and potential customers. In this case, you want to geocode as many of these customers as possible and do not need an exact street address match for each one. Geocoding to postal code centroid is fine for your analysis.

On the other hand, if you, as a utility service coordinator, need to know where your customers are in relation to neighborhood gas lines, the positional accuracy of each customer is of critical importance to you. Geocoding to street level with strict matching preferences is your best strategy.

A Few Words About Addresses

The quality of the address data is of utmost importance in geocoding. A clean input address that follows standard address conventions for the locale, will yield better results than incomplete or poorly formed addresses.

This section is an overview of address quality for both the input addresses as well as the matching reference address. To get the most out of geocoding, it's important to understand the address structure of your data and the data to which you are matching.

Input Addresses

Input addresses are those addresses you wish to geocode. These are addresses that do not contain geographic coordinates and thus cannot be located on a map.

Input addresses are made up of several components that the geocode server examines to determine a match. Address number, street name and street prefixes and suffixes can be contained in an input address. Prefixes are commonly directional in nature, such as North or South and exist only for some addresses. Suffixes are typically the type of street, Street, Road, Avenue Other information you might find in an input address is an apartment or route number.

Input addresses can contain a full or partial address and can contain spelling variations. For example, LaSalle Street as input will match to LaSalle St. However LaSalle without a street suffix may not because the address could really be LaSalle Ave.

The MapXtreme geocoding client can use so-called “dirty” data and incomplete addresses. This client accommodates both U.S. and non-U.S. addresses, provided the server contains the appropriate reference addresses.

Reference Addresses

Reference addresses are those that the server matches against the input addresses. These records contain the geographic coordinates required to locate the address on a map. Reference addresses are maintained on the server and are updated regularly through maintenance programs to accommodate new addresses. They have been standardized based on conventions of the locale. For example, in North America, street numbers precede the street name. In other parts of the world, the street number may follow the street name.
Geocoding a Location

Typically the reference address contains the essential components: address number, street name and prefix, if one exists, and suffix. For non-U.S. locales, additional information such as apartment number, may not be contained in the reference address and thus, ignored in the input address.

What are Custom User Dictionaries?

There may be situations where the MapInfo Address Dictionary, provided with the server or service, does not contain the desired coverage for the areas or regions in which you are interested. Also users may have their own custom data they may prefer to use (for example, the data for all the company stores in the country). For these purposes users can use their own dictionaries that contain this kind of data.

The geocode client supports the use of custom dictionaries. A custom dictionary is a table of streets and address ranges that you match ungeocoded records against. You can use as many custom dictionaries as you need when geocoding. Refer to the server or service documentation for more information on creating custom user dictionaries.

What is Geocoding World?

The geocoding client allows you to geocode in different countries by specifying the country code in the input address. The geocode world component is different from the MapMarker-based component, which provides one country or area per component. The geocode world component provides city and postal code data coverage for 238 countries or areas, and street level data coverage for nine countries. Note that for both options access to data is licensed, and requires having a valid data license file. For information on Geocoding World, please contact your MapInfo sales representative.

Geocoding a Location

Geocoding a location helps visualize your data relationships. For example, once address records are geocoded, they can be used by the routing client to display them as driving directions between two addresses (locations). Once data has geographic references, spatial searches can be performed to answer questions such as "Find all customers within 10 miles of this location." All geocode requests can:

- Geocode an incomplete address and return a complete set of address information (a normalized address).
- Indicate the number of exact or close matches in the response for a particular address supplied in the geocoding request.
- Process one or more addresses in a single geocoding request.
- Provide information on the quality of the result by using a match code.

Using the GeocodeRequest class you can send a request to the geocoding server or service. Its properties include AddressList (a list of addresses to be geocoded), Length (the number of items in the address list), and GeocodeContraints (see Using Constraints for Accurate Geocoding on page 325 for more information). As part of the input address, you can include a variety of
information, including street address, intersections, primary and secondary postal codes, place names, and a country code. The level of information included in the input address will determine the level and accuracy of the geocode result candidates.

Using the geocoding client, you can perform various levels of geocoding:

**Street Address Geocoding**

You can use geocoded street address information to display on a map or to perform spatial searches and queries. For example, this is useful for displaying store locations and the customers who are part of a store loyalty program on a map to determine market regions. For street address geocoding, use the Address class to specify the available information for the input address.

**How do I Geocode an Address?**

The following example shows how to geocode a street address in C# using the geocoding client. There are three sections to a geocode request: Define the parameters, create the street and address objects, and create the geocode request.

See also a Geocode sample application provided in ..\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features\Geocode\cs.

**Define your Parameters**

To populate the address object you must define your address parameters. In addition to the address parameters, you must also define the server or service URL.

```csharp
//Define the server URL
String MMJUrl = "";

//Define the address parameters
String streetName = "One Global View";
String cityName = "Troy";
String stateName = "NY";
String zipCode = "12180";
String countryCode = "USA";
//The following are optional address parameters
String directionalPrefix = "";
String typePrefix = "";
String suffix = "";
String directionalSuffix = "";
```

**Create the Address**

Populate the address by creating the Street, StreetAddress, and Address objects.

```csharp
//Create a Street object
MapInfo.Geocoding.Street street =
    new MapInfo.Geocoding.Street(
        directionalPrefix,
        typePrefix,
        streetName,
```
directionalSuffix);

//Create a StreetAddress object
MapInfo.Geocoding.StreetAddress streetAddress =
    new MapInfo.Geocoding.StreetAddress(street);

//Create an Address object
MapInfo.Geocoding.Address address =
    new MapInfo.Geocoding.Address(streetAddress, countryCode);
address.PrimaryPostalCode = zipCode;
address.PlaceList =
    new MapInfo.Geocoding.Place[
    {new MapInfo.Geocoding.Place(
        cityName,
        MapInfo.Geocoding.NamedPlaceClassification.Municipality),
    new MapInfo.Geocoding.Place(
        stateName,
        MapInfo.Geocoding.NamedPlaceClassification.CountrySubdivision)}];

Create the Geocode Request

Once the address object is created, add the address to a list and use the list to create the geocode request. Using the GeocodeClientFactory, define whether the request is being sent to the geocode server or service. Use the GetMmjHttpClient method for a request sent to the geocode server or GetEnvinsaLocationUtilityService method for a request sent to the geocode service.

//Add the address to a list of addresses
MapInfo.Geocoding.Address[] addressList = {address};

//Create the geocode request
MapInfo.Geocoding.GeocodeRequest geoRequest =
    new MapInfo.Geocoding.GeocodeRequest(addressList);

//Create the geocoding client
MapInfo.Geocoding.IGeocodeClient geoClient =
    MapInfo.Geocoding.GeocodeClientFactory.GetMmjHttpClient(MMJUrl);

//Send the request and get the response
MapInfo.Geocoding.GeocodeResponse geoResponse =
    geoClient.Geocode(geoRequest);

Street Intersection Geocoding

Street intersections can be made up of different address types or address elements, and the request may have different levels of accuracy or preferences. For example, a mobile subscriber could use street intersection geocoding to enter the nearest street intersection to view a map of their location on the mobile device. For street intersection geocoding, use the StreetIntersection class and specify a Street and an IntersectingStreet property.
Postal Code Geocoding

Postal code centroids represent the centre of a postal code region. Being able to locate these centroids on a map lets you perform demographic analysis to find market regions to target for advertising or direct mail. For postal code level geocoding, use the Address class specifying only the PrimaryPostalCode information.

Gazetteer Type Geocoding

As part of the geocoding client, you can perform a world, country, or city geocode operation that can find the position of a partial address containing only the city, state, or a place name. The following operations are supported:

- Search, based on country, country subdivision, city, city subdivision, landmark, or airport to return the position.
- The pattern search capability supports a wildcard character (*) as a search value in one or more fields to return all available values for that particular field.

Batch Geocoding

Batch geocoding is supported using the IGeocodeClient.BatchRequests property. This property sets the number of addresses that will be sent to the server for a single operation. The defaults number of addresses is 25. The BatchRequest property must be greater than 0 and less than 500. When submitting addresses to the IGeocodeClient.Geocode() method, you may exceed the number specified in the BatchRequest property. If this occurs, the geocode operation containing these addresses will be broken down into multiple operations, where each operation contains no more than the number specified in the BatchRequest property.

Note A small number for BatchRequest (e.g., 2) will slow performance. A high number for BatchRequest (e.g., 500) may result in memory issues. The performance of batch geocoding is dependant on system configuration.

Using Constraints for Accurate Geocoding

The geocode client allows you to set the match restrictions, fallback, and multiple match settings when geocoding input addresses using the GeocodeConstraints class. Here you set the preferences to be as strict or relaxed as you need. The settings that give you the best compromise among match rate, accuracy, and performance, are:

- Require an exact match for house number and street name.
- Do not fall back to postal code centroid.
- Return close matches only.

The strictest matching conditions require an exact match on house number, street name, postal code and no fallback to postal code centroids. The server or service, in essence, looks for an exact street address match within the postal code named in your input address.
What are the Match Constraints?

The match constraints control how to match a given input address, the number or candidates returned for each address, and what auxiliary information will be returned. An input address may not exactly match a real world address possibly due to incomplete or incorrect information. The GeocodeConstraints class contains a series of properties to determine which pieces of the address are considered most important, and what to do when an exact match is not found.

The IGeocodeClient.DefaultGeocodeConstraints property can be used to retrieve the defaults from the server or service if possible. The geocode client will use the defaults from the server or service if no GeocodeConstraints are supplied.

Each of the following preferences impact the level of record matching that the geocode server or service employs:

- **CloseMatchesOnly** – Returns only those geocoded results that are close match candidates. The address candidates are ranked according to how closely the input address matches the MustMatchXXX preferences. Only those candidates flagged as close matches (true) are returned. The default is false.

- **MustMatchAddressNumber** – Only candidates matching the address number are considered close. The default is true.

- **MustMatchMainAddress** – Only candidates matching the street name are considered close. This does not prevent Soundex matches from being considered. (For example, "Muller Strasse" for "Mueller Strasse" may still be considered close). It does prevent expanded street name manipulation (misspelling), such as considering juxtaposed letters (For example, "Muleler Strasse" for Mueller Strasse"), if the input street does not generate a close match. The default is true.

- **MustMatchCountry** – Only candidates matching Country are considered close. The default is true. It is recommended that this preference is not changed.

- **MustMatchCountrySubdivision** – Only candidates matching Country Subdivision are considered close. The default is true. CountrySubdivision may be different in each country. For example, in USA, it is state, in CAN, it is province.

- **MustMatchCountrySecondarySubdivision** – Only candidates matching Country Secondary Subdivision are considered close. The default is false. CountrySecondarySubdivision are different for each country. For example, in the USA it is county, in Canada, it is Census Division.

- **MustMatchMunicipality** – Only candidates matching Municipality are considered close. The default is true. An example of a Municipality is a town or city.

- **MustMatchMunicipalitySubdivision** – Only candidates matching Municipality Subdivision are considered close. The default is false. An example of a MunicipalitySubdivision is an MSA (Metropolitan Statistical Area), or a borough of a city (such as Bronx or Brooklyn in New York City).

- **MustMatchPostalCode** – Only candidates matching postal code are considered close. The default is false.

- **MustMatchInput** – Only candidates matching all input preferences are considered close, regardless of what is set for that particular preference. The default is false.

- **FallbackToGeographic** – Determines if a street geocode request which produces no candidates, the geocode is performed at the centroid of a geographic region. The resulting point is located at the geographic centroid of the area where it is possible to obtain the highest level of accuracy. The default value is set to false.
Chapter 19: Geocoding

- **FallbackToPostal** – Determines if a street geocode request which produces no close candidates, the geocode is performed on the postal code centroid. The default value is set to false.

- **OffsetFromStreet** – Determines the distance from the street segment at which the address is positioned. This value is used to prevent addresses across the street from each other from being given the same point. For example, a house with number 50 on a segment that spans from 40 to 60 is interpolated as the midpoint of the segment. The OffsetFromStreet positions the point perpendicular from the road, so that houses on the left side of the street appear on the left, and the houses on the right side of the street appear on the right. The default value for OffsetFromStreet is 25 meters. The Distance class is used to specify the OffsetFromStreet by defining the value and unit of measure.

- **OffsetFromCorner** – Determines the distance from the end (corner) of a street segment at which the address is positioned. This value is used to prevent addresses at street corners from being given the same point of the intersection. OffsetFromCorner positions the point parallel to the street (based on the street address) a distance along the segment. For example, a house with number 40 on a segment that spans from 40 to 60 is positioned so that its point is not located at the street intersection. The default value for OffsetFromCorner is 25 meters. The Distance class is used to specify the OffsetFromCorner by defining the value and unit of measure.

- **MaxCandidates** – Defines the maximum number of candidates returned in a response. The actual number of candidates returned may be less than this maximum value. A value of -1 is used to mean return all candidates that meet the other constraints. The default is 3 when using the MapInfo geocoding server. The default is -1 (all) when using the Envinsa Location Utility Service.

- **MaxRanges** – This preference controls whether a geocode operation returns the exact number for a street address or a range of numbers. If the MaxRanges is set to 0, then no ranges are returned and only the exact address number is returned. If the MaxRanges is set to a value greater than 0, then that number of street address numbers is returned. The default value for MaxRanges is 0. For example, if MaxRanges is set to 5, then the service returns up to 5 matches within the address range. A value of -1 returns all suitable matches.

- **MaxRangeUnits** – Defines the maximum number of candidate range units to be returned per candidate range. The default value for MaxRangeUnits is zero (0).

- **GeocodeType** – Determines the type of geocoding when available from the server. Refer to your server or service documentation for information on geocode type support.

- **LocalGeocodeConstraints** – Allows you to set country specific constraints which are specific to that country and that geocoder. These values are always Key/Value pairs. Refer to your server or service documentation for information on available local constraints.

- **DictionaryUsage** – Specifies if the server address dictionary is to be used by itself, if a configured user dictionary is to be used by itself, or if both of them are to be used. If both are to be used, a preference of one over the other can be requested. By default the AD_AND_UD option is used. This option indicates that the server address dictionary and any user dictionaries that are configured should be used. For information on creating and accessing user dictionaries, refer to your server or service documentation.

- **UseCASSRules** – Specifies if CASS level geocoding should be used by the server (US only). CASS certification is a process by which a table of mailing addresses is standardized to meet U.S. Postal Service® (USPS) requirements for bulk mailing discounts. The geocoding client performs this address standardization under strict matching conditions set by the U.S. Postal Service while it geocodes your records. You must deploy a version of the US geocode data component that supports CASS in order to use CASS geocoding. Contact your MapInfo sales representative to obtain a current version in order to comply with CASS address standardization. The default is false.
Impact of Relaxing Match Constraints

Relaxing the conditions broaden the area in which it searches for a match. For example, by relaxing the postal code, a search will be performed for candidates outside the postal code but within the city of your input address. Consider the following when you are determining the settings for geocoding:

Relaxing Postal Codes

When postal codes are relaxed, a search is performed on a wider area for a match. While this results in slower performance, the match rate is higher because the request does not need to match exactly when it compares match candidates.

Relaxing Subdivisions and Municipalities

When these are relaxed, a search is performed on the street address matched to the particular postal code, and considers other cities or subdivisions that do not match the name, but do match the postal code.

Relaxing Street Name

A search is performed that looks at all candidates with names that sound like the input address or that are spelled improperly. This slows down the performance. On the plus side, since more candidates are examined, the match rate increases. If your table is indexed, the time difference between performance and match rate is reduced.

Relaxing House Number

Performance is not significantly affected when the house number setting is relaxed. It does, however, affect the type of match if the candidate address corresponds to a segment that does not contain any ranges. The type of match can also be affected when the house number range for a candidate does not contain the input house number. If you are relaxing the house number, it is recommended that you set the maximum ranges to be returned to a value higher than 0.

Understanding Accuracy for Close Matches

The ranking of the results is based on the close match of the request. For the GeocodeMatchCode class, the geocoding server or service returns a result code for every record it attempts to match based on the combination of GeocodeConstraints in the request and server or service configuration. The code represents the success or failure of the geocoding operation and conveys information about the quality of the match. The codes fall into five major categories:

• Single Close Match (S Category)
• Best Match from Multiple Candidates (M Category)
• Postal Code Centroid Matches (Z Category)
• Geographic Centroid Matches (G category)
• Non-Match Codes

For either S or M category result codes, each character of the code tells how precisely the geocode operation matched each address component.
Chapter 19: Geocoding

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>House number</td>
<td>115</td>
</tr>
<tr>
<td>P</td>
<td>Street prefix direction</td>
<td>North</td>
</tr>
<tr>
<td>N</td>
<td>Street name</td>
<td>Main</td>
</tr>
<tr>
<td>T</td>
<td>Street type</td>
<td>Place</td>
</tr>
<tr>
<td>S</td>
<td>Street suffix direction</td>
<td>SE</td>
</tr>
<tr>
<td>C</td>
<td>City name</td>
<td>New York</td>
</tr>
<tr>
<td>Z</td>
<td>Postal code</td>
<td>80302</td>
</tr>
<tr>
<td>A</td>
<td>Address Dictionary</td>
<td>A</td>
</tr>
</tbody>
</table>

The following example is a close match candidate:

S5HPNTSCZA

The following example is not a close match candidate:

S5-PNTSC-A

A dash in the result code indicates a conflict. If the postal code in the request conflicts with one found by the geocode in the data (input was 28031, value found is 28013), the resultant string contains a dash instead of a Z (For example, S5HPNTSC-A).

**Single Close Match (S Category)**

Matches in the S category indicate that the record was matched to a single address candidate. The first character (S) reflects that the geocoding server found a street address that matches the record. The second position in the code reflects the positional accuracy of the resulting point for the geocoded record.

- S1 – single close match, point located at postal code centroid
- S2 – single close match, point located at ZIP+2 centroid (US centric)
- S3 – single close match, point located at ZIP+4 centroid (US centric)
- S4 – point located at the center of a shape point path (shape points define the shape of the street polyline)
- S5 – point located at a street address position
- S6 – point located at centroid of geometry postal code. (For example, large buildings having their own codes). This represents the highest accuracy available
- SX – point located at street intersection
- SO – no coordinates available (very rare occurrence)
Best Match from Multiple Candidates (M Category)

Matches in the M category indicate that there is more than one close match candidate for the record and the server or service has chosen the best one of those candidates. As in the S category, the second position in the code of M category matches the positional accuracy of the resulting point object.

- M1 – multiple close matches, point located at postal code centroid
- M2 – multiple close matches, point located at ZIP+2 centroid (US centric)
- M3 – multiple close matches, point located at ZIP+4 centroid (US centric)
- M4 – multiple close matches, point located at the center of a shape point path (shape points define the shape of the street polyline)
- M5 – multiple close matches, point located at a street address position (highest accuracy available)
- M6 – multiple close matches, point located at point postal code location
- MX – multiple close matches, point located at street intersection
- M0 – multiple close matches, no coordinates available

Postal Code Centroid Matches (Z Category)

Matches in the Z category indicate that no street match was made, either: 1) because there is no close match and you allowed the server or service to fall back to postal code centroid; 2) the address is a P.O. Box or rural address; or 3) you set the server or service to match to postal code centroids. The resulting point is located at the postal code centroid with four possible accuracy levels.

- Z1 – Postal code centroid match
- Z2 – ZIP+2 centroid match (US centric)
- Z3 – ZIP+4 centroid match (highest accuracy available, US centric)
- Z6 – Postal code centroid match for point ZIP
- Z0 – Postal code match, no coordinates available (very rare)

Note: Point ZIPs are ZIP Codes without an area. These include P.O. Box ZIPs and Unique ZIPs (single site, building, or organization). When using a non US address dictionary, the results in this category will still be a Z.

Geographic Centroid Matches (G category)

Matches in the G category indicate that no street match was made. This takes place when no close match is found, and the server or service is configured to fall back to the geographic centroid. The resulting point is located at the geographic centroid with the following possible accuracy levels.

- G0 – areaName0 centroid (country)
- G1 – areaName1 centroid (country subdivision)
- G2 – areaName2 centroid (country secondary subdivision)
- G3 – areaName3 centroid (municipality)
- G4 – areaName4 centroid (municipality subdivision)
Non-Match Codes

The following result codes indicate no match was made:

N – No close match. These records can be re-geocoded interactively using MapInfo geocoding products or during subsequent automatic passes under different matching conditions.
NX – No close match for street intersections.
ND – The server or service could not find the Address Dictionary for the given postal code or municipality/country subdivision. These records can also be re-geocoded once the Address Dictionary is available.
NG – The user marked these records during geocoding as non-geocodable. The server or service does not attempt to match these records again until the code is removed.
Routing

This chapter covers the MapXtreme namespace for Routing and provides descriptions and examples for writing applications that will access routing servers or services.

In this chapter:

- Overview of MapInfo.Routing Namespace .................. 334
- Calculating Routes .............................................. 335
- Advanced Route Options ...................................... 338
- Iso Routing (Drive-Time and Drive-Distance) ............... 342
- Updating a Request Using Routing Data ....................... 347
Overview of MapInfo.Routing Namespace

The MapInfo.Routing namespace provides an interface and classes for incorporating a MapXtreme routing client into your application, in a similar fashion to the geocoding client. The routing client sends requests via HTTP to Pitney Bowes MapInfo’s routing server or routing web service product. The server and web service are not included in the purchase of MapXtreme.

Using the routing capabilities, you can determine the shortest or fastest path (route) between two points, with the ability to add various advanced routing options. You can return the route geometry (layout of the route for map display), turn-by-turn directions, and a route summary of total distance and time. Additionally, you can create routes that include intermediate points and preferences that avoid certain road features, such as highways or certain locations.

The routing capabilities include important analytical tools, allowing you to calculate drive-time and drive-distance (isoChrone and isoDistance), and the ability to use transient updates to avoid certain road types or to recalculate a route. The routing client functionality is divided into four logical groups:

- Calculating Routes
- Advanced Route Options
- Iso Routing (Drive-Time and Drive-Distance)
- Updating a Request using Routing Data

Main Routing Classes

Routing is supported using either a MapInfo routing server or the Envinsa Route service. One of these resources needs to be deployed and made available via a URL. To create the appropriate routing client, use the RouteClientFactory class. The interface for both clients is similar as they both use the same classes for route requests, preferences, and responses.

Using the routing client you can determine the shortest or fastest path (route) between two locations, add intermediate locations (ViaPoint class), avoid locations or road types (AvoidList, PointExclude, SegmentExclude classes), get driving directions (RouteInstructionsRequest class), get the route geometry (RouteGeometryRequest class), and customize your routes using numerous preferences resulting in more meaningful analysis. The routing client includes important analytical tools, allowing you to calculate drive-time and drive-distance (isoChrone and isoDistance), and the ability to use transient updates to avoid certain road types or to recalculate a route. The routing client functionality is divided into three logical groups:

1. Calculating routes using the RouteRequest class for point-to-point and multi-point routes, and the MatrixRouteRequest class for matrix routes.
2. Calculating drive-time and drive-distance using the IsoChroneRequest class for drive-time and the IsoDistanceRequest class for drive-distance.
3. Calculating routes with transient updates to produce more accurate and meaningful routes using the TransientUpdate class.
Calculating Routes

There are three types of route calculations (determining shortest or fastest route) that can be performed: simple routes (point-to-point), multi-point routes, and matrix route requests.

Point-to-Point Routing

One of the most common routing requirements is for simple driving directions where a customer inputs two addresses, two points, or any combination, and gets back a route. The route will be calculated from a specified start point to a specified end point.

At the simplest level of point-to-point routing, the response includes a route summary. This summary includes the distance and time of the route. There are numerous additions you can make to a point-to-point route request to return additional information in the response. These additions are:

- Routing Preferences
- Driving Directions
- Route Geometry
- Avoiding Points, Features, and Segments
Calculating Routes

- Time-Based Routing
- Updating a Request using Routing Data

A point-to-point route is created using the RouteRequest class. Refer to the Developer Reference for API-level syntax and descriptions.

The following code sample is an example of a simple point-to-point route:

**VB example:**

```vbnet
Public Shared Sub New_RouteRequest()
    ' Create the start point and end point
    Dim coordSys As MapInfo.Geometry.CoordSys = _
        Session.Current.CoordSysFactory.CreateFromPrjString("1, 104")
    Dim dpt1 As MapInfo.Geometry.DPoint = _
        New MapInfo.Geometry.DPoint(-74, 42)
    Dim dpt2 As MapInfo.Geometry.DPoint = _
        New MapInfo.Geometry.DPoint(-74, 41)
    Dim startPoint As MapInfo.Geometry.Point = _
        New MapInfo.Geometry.Point(coordSys, dpt1)
    Dim endPoint As MapInfo.Geometry.Point = _
        New MapInfo.Geometry.Point(coordSys, dpt2)
    ' Create the point list
    Dim pointList As WayPointList = _
        New WayPointList(startPoint, endPoint)
    ' Create the route plan object
    Dim plan As RoutePlan = _
        New RoutePlan(pointList)
    ' Create the request object
    Dim request As RouteRequest = _
        New RouteRequest(plan, DistanceUnit.Mile)
End Sub
```

**Multi-Point Routing**

Multi-point routing is the ability to route via points. The routing server will find the shortest time or distance through all the points. This can be used for complex directions. These directions are a requirement for customers who wish to plan daily drop-offs, or customer visits such as distribution and repair companies, or general field sales. Typically, a start and end point are defined plus a series of stops are added into the calculation. There is no difference between multi-point route and point-to-point route except for the via points between the start and end locations.

Similar to point-to-point routes, there are numerous additions you can make to a multi-point route request to return additional information in the response. These additions are:

- **Routing Preferences on page 338**
- **Driving Directions on page 339**
- **Route Geometry on page 340**
- **Avoiding Points, Features, and Segments on page 340**
- **Time-Based Routing on page 341**
- **Updating a Request Using Routing Data on page 347**
A multi-point route is created using the RouteRequest class. To add intermediate points to a route, WayPointList must consist of at least one ViaPoint. Refer to the Developer Reference for API-level syntax and descriptions.

The following code sample shows how to create a WayPointList consisting of a startPoint, endPoint, and ViaPoint:

**VB example:**

```vbnet
Public Shared Sub New_WayPointList()
    ' Create a start point and end point
    Dim dpt1 As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint(-74, 42)
    Dim dpt2 As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint(-73, 42)
    Dim dpt3 As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint(-73.562, 42)
    Dim startPoint As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt1)
    Dim endPoint As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt2)
    Dim intermediatePoint As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt3)

    ' Create the intermediate point object
    Dim point As ViaPoint = New ViaPoint(intermediatePoint)
    ' Create the collection
    Dim pointList As ViaPointList = New ViaPointList
    ' Append the object to the collection.
    pointList.Append(point)
    ' Create the WayPointList object
    Dim list As WayPointList = New WayPointList(startPoint, endPoint, pointList)
End Sub
```

**Matrix Routing**

Matrix routing allows you to find the shortest or fastest paths between a number of start points and a number of end points, and return the route costs. The costs are the total time and distance of the individual routes. The matrix route feature is extremely useful for situations where you have 3 start points and 50 destination points, where you want to find the shortest paths between all the start points and all the destinations, and return the routes’ costs.

The order of the points and number of sequences in the response are determined by the number of start and end points in the request. For example, if two start points and two end points are specified in the request, the response will contain the following order of sequences (where S is start and E is end): S1 to E1, S1 to E2, S2 to E1, S2 to E2.
Typically, this calculation can be used to determine service response time and coverages for specific services such as a fire house or police station. You may require these calculations to ensure they can provide adequate coverage for service level agreements such as having one or more people who can respond to an incident within 20 minutes of first notification. At the simplest level of matrix routing, a list of sequences will be included in the response. Each sequence includes the start and end point, as well as the distance, and time. There are numerous additions you can make to a matrix route request to return additional information in the response. These additions are:

- **Routing Preferences on page 338**
- **Avoiding Points, Features, and Segments on page 340**
- **Updating a Request Using Routing Data on page 347**

A matrix route is created using the MatrixRouteRequest class. The matrix route request must consist of a MatrixRouteCost and may include a MatrixRoutingPreferences. Refer to the Developer Reference for API-level syntax and descriptions.

### Advanced Route Options

There are numerous advanced capabilities that can be added to your routing applications. This section highlights these important capabilities used to customize your point-to-point, multi-point, and matrix route requests.

#### Routing Preferences

You can specify various preferences for each type of route you are calculating. All of these preferences are optional, but can be very important in determining the accuracy of the route and helping you create a more meaningful route. You can return the route with fastest time from the start point to the end point or the shortest distance from the start point to the end point. If specified, these preferences will override the route optimization settings defined in the server or service configuration. The preferences may be overridden by server preferences if the server preferences are more restrictive.

The type of route calculation (point-to-point, multi-point, or matrix) determines which route preferences are available to use. The following list outlines the available route preferences:

- **DistanceUnit** - You can specify the distance unit for all of the route calculations. The most common units are Kilometer, Meter, Mile, Yard, and Foot. If a value is not specified, the default setting will be used (Mile).
- **RouteMethod** - Determines the general type of route preference to perform. The route client can either perform Fastest or Shortest route calculations. If a value is not specified, the default setting will be used (Fastest).
- **OptimizeIntermediatePoints** (only used in multi-point routes) - Determines if the via points are to be traversed in the order specified in the request or have the route client find the optimal route using these via points. By default (false) the route client will traverse the specified points in the order in the request. When set to true, the optimal route using the via points is calculated.
- **StopThreshold** (only used in multi-point routes) - Sets the route calculation algorithm stop threshold. The route algorithm will stop calculating and return the current ‘best’ route when the difference in time or distance between the route candidates remaining in the algorithm reaches the stop threshold. Valid values are any positive numbers. For best performance, a positive
number less than one is recommended. Setting the stop threshold is a balancing factor between accuracy and speed. The lower the threshold value on average, the more accurate the result and the longer the route will take to calculate. The stop threshold has a minimal effect on routes with few via points (under 10). The default is 0.01 (1 percent).

- **TimeOut** - Sets the route calculation algorithm time out value. The time out value is in seconds. The default time out is 600 seconds. The route algorithm will stop calculating once the time out value is reached.

- **TimeUnit** - Sets the unit of measure for all routes durations. The default time unit is Minute.

- **TravelPreferences** - Allows you to specify unique routing abilities for modifying road type priority of travel. To add travel preferences you can specify a road type and a road type preference. Routing can be performed on priority basis by setting the RoadTypePreference to High, Medium, Low, or Avoid. The road types with the higher priority will be chosen over those with the lower priorities when the route is calculated. You can also use the Avoid RoadTypePreference to avoid a particular road type. One of the more often used travel preference is to set the FERRY RoadType to Low, which will only use ferry routes if that is the only mean of travel to the destination.

- **TravelTime** (point-to-point and multi-point) - Sets the start or end travel time for the route request. Refer to Time-Based Routing on page 341 for more information on how to define the travel time and other time-based routing options.

- **Updates** - Allows you to include transient updates in your route request that temporarily modify road or segment speed and road types. Refer to Transient Updates on page 348 for more information on including updates.

The RoutingPreferences and MatrixRoutingPreferences classes are the starting point when defining route preferences. Refer to the Developer Reference for syntax and descriptions.

### Driving Directions

Driving directions can be returned for a point-to-point or multi-point route. These instructions list the step-by-step procedure for the route. Each segment of the route will return the step that needs to be performed (for example, Begin, Turn Right), the name of the street, the direction, the distance, and the time. There are numerous options for driving directions including focusing the directions to a particular section of the route, changing the wording and format of the directions, and changing the language for the directions.

There are two places where this can be configured/requested. The first place is in the server or service settings. The second place is using the RouteInstructionsRequest class setting the ReturnDirections property to true.

### Modifying Direction Preferences

The following list describes the available preferences used to modify driving directions:

- **DirectionBreakTurnAngle** - Sets the turn angle value that determines when a street is broken into a new directions string. Sometimes, when following a route a street will make a significant turn while keeping the same name. By using this value, the user can specify the turn angle at which a new direction should be started. Valid values are 0 to 180 degrees. The default is 45 degrees.

- **DirectionsGeneratorName** - Specifies a custom, server-side directions generator to be used to create the route directions. If the server or service is configured with one or more directions generators, this preference may be used to force directions generation to be handled by a
specific generator. For instance, setting “myGenerator” will force the server or service to use the
generator that has been configured with the name “myGenerator”. Use this preference only if
your server or service has been configured with a custom direction generator. Refer to your
server or service documentation for instructions on how to create and use direction generators.

- **Focus** - Specifies the focus of the route. A focused route is a subset of the whole route that
  concentrates on either the beginning or end of the route. A route focused at the start will route
  the user from their origin to (and including) the first major highway. A route focused at the end will
  route the user from the last major highway in the route (including that highway) to the
  destination. If there are no major highways in the route, the focused route will be the same as an
  unfocused route. NONE signifies directions for the whole route will be returned, START signifies
  that the start of the route will be returned, or END signifies that the end of the route will be
  returned. The default is NONE.

- **ShowDistance** - Indicates whether or not to return the distance of a direction in the driving
  directions. The default is true.

- **ShowPrimaryNameOnly** - Specifies whether or not to show only the primary street names. It is
  often the case that a street contains multiple names. This is used to indicate whether all names
  for the street should be shown in the directions or only the primary name. If set to true, only the
  primary name of the street will be used in the directions. If set to false, the primary name and all
  alternate names will be used. The default is false.

- **ShowTime** - Indicates whether or not to return the time it takes to follow a direction in the driving
  directions. The default is true.

- **Style** - Specifies the type of directions to return. The client can return two types of directions,
  normal and terse. Normal is the standard driving direction instructions. For example “Turn left on
  Yonge Blvd and travel Southeast (1 s)”. Terse directions are shorter directions that are more
  suitable for wireless devices. For example “L on Yonge Blvd”. This can be set to NORMAL or
  TERSE. The default setting is NORMAL.

### Setting the Direction Language

You can manipulate the output language of your route instructions by changing the UserLocale
property. The route client uses ISO standard locale code <language>_<country> (for example,
French is fr_FR, German is de_DE). Currently the route client supports EN, FR, DA, DE, FI, NL, NO,
ES, PT, IT, SV. English (en_US) is the default language.

### Route Geometry

The ability to create a route geometry is a vital capability in creating maps and analysis of a route.
The route geometry is the visual representation of the route. By default the route geometry is not
returned. However, you can return the route geometry in the response by setting the
ReturnGeometry property to true.

### Avoiding Points, Features, and Segments

You can avoid or exclude types of roads, points, or route segments. Before determining a route, you
may already know of a road or location that you would like to avoid based on traffic information,
accidents, holidays, rush hour, or any other known factor.
Chapter 20: Routing

There are three types of avoids or excludes: points, features, and segments:

**Avoiding Points**
Currently there are two methods to avoid points along the route. You can use the Points property of the AvoidList class or you can exclude any number of points from a route request using the PointExclude class.

*Note* Using the PointExclude class is the recommended method to avoid points.

**Avoiding Features**
Currently there are two methods to avoid features (road types) along the route. The TravelPreferences property of the RoutingPreferences class can be used to avoid any road type. See **Routing Preferences on page 338** for more information on using the TravelPreference property.

The route client also provides another method to avoid highways when calculating routes. This feature is useful when you want to limit routes to local roads. You can use the Features property of the AvoidList class and define Highway as the avoid feature.

**Avoiding Segments**
You can exclude any number of route segments (ID) from a route request using the SegmentExclude class. The segment ID is a unique identifier assigned to each line, road, or section of the route data. You can determine the segment ID for a particular section of the route by returning them in a previous route response. See **Returning Segment Information on page 347** for more information on how segment ID's are returned in a route response.

*Note* If the route cannot be calculated or there is no way to get a destination using the avoid or exclude, then the route directions will not be returned.

**Time-Based Routing**
Time-based routing is a key component for delivery systems, mobile work forces, and calculating accurate drive times and routes. You can specify a start or end time for your route or a stop duration for each intermediate point in a multi-point route. For example, you can specify that the route should start at location A and pass through locations B, C, and D where you spend five minutes at each talking to customers or loading your trucks. The route will then end at location E.

**Start and End Times**
Part of time-based routing is being able to specify the start or end time for a route. Start and end times in routing are important when other time-of-day preferences are used. For example if the server is configured with data that represents rush hour traffic patterns, the start or end time for your route may be of importance for analysis. The start and end time can be set using the TravelTime class. The following properties are used to control the travel time:

- **DateTime** - Specifies the date and time to either start or end a route calculation.
- **TimeZone** - Specifies the time zone used in the travel time. The time zone is defined using an hour offset value from Greenwich Mean Time (GMT).
Iso Routing (Drive-Time and Drive-Distance)

- **TravelTimePreference** - Specifies if the travel time defined is a start or end time. The two members are defined using StartTravel or EndTravel.

The following code example shows how to set the travel time for the route. This example includes the StartTravel preference, defining the start time (year, month, day, hour (24 hour clock), minute, second) and time zone:

VB example:

```vb
Public Shared Sub New_TravelTime()
    Dim startTime As DateTime = New DateTime(2005, 5, 1, 14, 0, 0)
    Dim timeZone As MapInfo.Routing.TimeZone = New MapInfo.Routing.TimeZone(-4)
    Dim travelTime As MapInfo.Routing.TravelTime = New MapInfo.Routing.TravelTime(startTime, TravelTimePreference.StartTravel, timeZone)
End Sub
```

Stop Times

At any intermediate point during a route you can specify a stop time. This time is added to the overall time along the route. The stop time is particularly useful for defining time that the user is going to spend at a particular location along the route (for example, loading and unloading time of a delivery truck) for a multi-point route. A stop can be added to the ViaPoint class by specifying a Stop and a StopTime property:

- **Stop** - You can specify whether to stop at a viapoint. By default no stops are calculated at viapoints (false). You can specify if you want to stop at the viapoint by defining the Stop equal to true.
- **StopTime** - Adds a stop time to any viapoint along the route. The stop time is defined by the TimeSpan, and will be added to the total time for the route. The default stop time is 0 (zero). A TimeSpan can be represented as a string in the format "[\-]d.hh:mm:ss.ff" where "-" is an optional sign for negative TimeSpan values, the "d" component is days, "hh" is hours, "mm" is minutes, "ss" is seconds, and "ff" is fractions of a second. For example, a TimeSpan defined as "11.13:46:40" is equivalent to 11 days, 13 hours, 46 minutes, and 40 seconds.

Iso Routing (Drive-Time and Drive-Distance)

IsoChrones and isoDistances can be extremely valuable information for making decisions. An isoChrone is a polygon or set of points representing an area that can be traversed in a network from a starting point in a given amount of time. An isoDistance is a polygon or set of points representing the area that is a certain distance from the starting point. They can be used to determine a drive-time or drive-distance boundary from a location. Users in retail, banking, and insurance, can use this service to determine the potential market or risk for any given asset. These boundaries may then be used for further analyses such as determining which prospects in a mailing list are within an iso boundary so they may be notified of new services available or a new store opening.
Creating an IsoChrone (Drive-Time)

When creating an isoChrone, you are looking for one or more polygons/set of nodes which represents an area that can be travelled in a given amount of time (drive-time cost) from a starting location.

The cost specifies the time used to calculate the isoChrone. The cost represents the time it takes to travel from the starting point to the calculated points on the road network. You can specify multiple costs to produce concentric bands (multiple isoChrones) that visually represent different distances that can be reached along the road network. Associated with multiple costs are tags. You can specify a tag, or ID, for each cost in the request, that will identify the appropriate isogram result (geometry) in the response.

There are numerous preferences you can specify in the IsoChronePreferences and IsogramPreferences classes. These preferences allow you to obtain a desired output for your analysis.

**IsoChronePreferences**

The following preferences are available in the IsoChronePreferences class:

- **DefaultAmbientSpeed** - Determines the off-road network speeds where they are not specified. Roads not identified in the network can be driveways or access roads, among others. For instance, if you are at a point with five minutes left on an isoChrone on the off-road network, boundary points would be put at a distance based on the ambient speed and the time left. So, if the ambient speed in this case was 15 miles per hour, boundary points would be put at a distance of 1.25 miles.

- **AmbientSpeedOverride** - Overrides an ambient speed for a specific road type. The ambient speed can be overridden for all road types. For example, it may be set to 30 Mph for major urban roads on the weekends due to busy traffic:

**IsogramPreferences**

The following preferences are available in the IsogramPreferences class:

- **BandingStyle** - Specifies the style of banding to be used in the result. Banding styles are the types of multiple time or distance bands that can be displayed based on multiple costs. The styles include Donut (each boundary is determined by subtracting out the next smallest boundary) or Encompassing (each boundary is determined independent of all others). Multiple isoChrone bands may be requested by specifying multiple cost factors, such as asking for the isoChrone 5 minutes away and 15 minutes away from the same starting point. These end up as approximate concentric bands. The user may choose to show both complete sets of data (Encompassing style, which shows everything) or just the band between the two (Donut style), for everything between 5 and 15 minutes away.

Donut banding can result in routing server problems when two boundaries (times) are almost identical. There are three request settings you can use to avoid this situation:

a. Do not use the maximum off road distance setting if possible with Donut banding. If you must use this setting, set it as large as possible.

b. Do not use a low ambient speed setting with Donut banding.

c. Do not make requests with cost increments that are small relative to the cost. For example, requesting 4, 5, and 6 minute costs (1 minute increments starting at 4 minutes) is not likely to be a problem, but 120, 121, and 122 minute costs may be a problem. The larger the cost the larger the cost increments will need to be.
The above example shows the geometries returned for an Encompassing style banding of cost 5, 10, and 15 minutes from the starting point. Notice how each geometry includes the area of the prior isoChrone.

The above example shows the geometries returned for a Donut style banding of cost 5, 10, and 15 minutes from the starting point. Notice how each geometry begins at the end of the prior geometry, excluding the area of the prior isoChrone.

- **MajorRoadsOnly** - Determines what road network is used in the calculation. The network can include major roads only or all roads. A major road is a main road or highway. If you choose to use major roads, performance will improve, but accuracy may decrease. The images below illustrate the behavior of the major roads option. The image on the left shows MajorRoadsOnly set to true while the image on the right shows MajorRoadsOnly set to false. Notice that the service uses side streets and other types of secondary roads when calculating the iso response if MajorRoadsOnly is set to false.

- **MaxOffRoadDistance** - Specifies the maximum amount of distance to come off the road network when using the ambient speed. The default setting for this property is no limit, allowing off-road travel to occur to the maximum of the isoChrone. The server may not be able to generate a response for a maximum off-road distance set to a very small value.
• **ReturnAccesibleNodes** - Specifies the type of isogram or feature to be returned from an isoChrone request be a set of nodes (MultiPoint). By default, ReturnAccesibleNodes is False. When this property is true, all of the points along the road network that can be reached for the isoChrone calculation will be returned.

• **ReturnGeometry** - Specifies the type of isogram or feature to be returned from an isoChrone request be a geometry (MultiPolygon). By default, ReturnGeometry is true. False will return all of the points along the road network that can be reached for the isoChrone calculation.

• **ReturnHoles** - Specifies whether or not to return holes in the response. Holes are areas within the larger boundary that cannot be reached within the desired time or distance, based on the road network. These pockets of territory are often neighborhoods of local roads that are cumbersome to traverse. Holes can be returned as is, or removed entirely. This setting will only apply if ReturnGeometry is set to true (default). If this property is not specified, the default setting, false, will be used.

• **ReturnIslands** - Specifies whether or not to return islands in the response. Islands are small areas outside the main boundary that can be reached within the desired time or distance. These areas are frequently located off exit ramps of major highways. Islands can be returned as is, or removed entirely. This setting will only apply if ReturnGeometry is set to true (default). If this property is not specified, the default setting, false, will be used.

• **SimplificationFactor** - Specifies the reduction factor for polygon complexity. The simplification factor indicates what percentage of the original points should be returned or that the resulting polygon should be based on. The polygon or set of points may contain many points. The simplification factor is a decimal number between 0 and 1.0 (1 being 100 percent and 0.01 being 1 percent). Lower numbers mean lower storage and lower transmission times. This setting will only apply if geometry is the result type. If this property is not specified, the default setting 0.05 (5 percent) will be used. The images below illustrate the behavior of the simplification factor option. The image on the left shows the simplification factor set to 0.01 while the image on the right shows the simplification factor set to 1.

• **TimeOut** - Specifies the amount of time allowed for the server to create the isogram. The default value for this property is 600 seconds (10 minutes).

### Creating an IsoDistance (Drive-Distance)

When creating an isoDistance, you are looking for one or more polygons/set of nodes which represent the area that can be travelled along a road network, given a distance (drive-distance cost) from a starting location.
The cost specified for an isoDistance request is similar to an isoChrone except the cost value is a measure of distance used to calculate the isogram. The cost represents the distance required to travel from the starting point to the calculated points on the road network. You can specify multiple costs to produce concentric bands (multiple isoDistances) that visually represent different distances that can be reached along the road network. Associated with multiple costs are tags. You can specify a tag, or ID, for each cost in the request, that will identify the appropriate iso result (geometry) in the response.

The following code sample shows how to define an IsoDistanceDefinition. The response will create an isogram (geometry) using the point as the center and a cost in miles \((pt, \text{DistanceUnit.Mile})\) defining the boundary:

**VB example:**

```vbnet
Public Shared Sub New_IsoDistanceDefinition()
    ' Create a point
    Dim dpt As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint (-74, 42)
    Dim pt As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt)
    ' Create a definition
    Dim def As IsoDistanceDefinition = New IsoDistanceDefinition(pt, DistanceUnit.Mile)
End Sub
```

There are numerous preferences you can specify in the IsoDistancePreferences and IsogramPreferences Classes. These preferences allow you to obtain a desired output for your analysis. The preferences specified in the IsogramPreferences Class are the same for both an isoDistance and isoChrone request. For a description of the IsogramPreferences, refer to [IsogramPreferences on page 343](#).

**IsoDistancePreferences**

The following preferences are available in the IsoDistancePreferences Class:

- **DefaultPropagationFactor** - Determines the off-road network percentage of the remaining cost (distance) for which off network travel is allowed when finding the isoDistance boundary. Roads not identified in the network can be driveways or access roads, among others. The propagation factor is a percentage of the cost used to calculate the distance between the starting point and the isoDistance. For example, if you were at a point with five miles left to go on an isoDistance on the off-road network, boundary points would be put at a distance based on the propagation factor and the distance left. So, if the propagation factor was 0.16, boundary points would be put at a distance of 0.8 miles.

- **PropagationFactorOverrides** - Overrides the propagation factor for a specific road type. The propagation factor can be overridden for all road types. For example, it may be set to 0.24 for major urban roads:
Chapter 20: Routing

Updating a Request Using Routing Data

Being able to access and your route data, then use this information to create more accurate and meaningful routes, is a very useful capability. After you create a route and get a route response, you may realize that the result contains a location or segment you don't want to pass, and a road type or segment where you may want to reduce or increase the speed. MapXtreme provides capabilities to return segment information on your route data and use the information in your route data to modify/update route requests (transient updates). This means you can update the routing network without reprocessing the base data.

The transient update process is comprised of two steps:

1. Return the route data segment information in a route response. This information is required to determine the segments used for the road type or speed update. For more information, refer to Returning Segment Information on page 347.

2. Based on information returned in your response, make temporary updates to the route data. This is called transient updates. These changes are submitted within a route request, and are only valid when the server or service is handling that particular request. This capability can be used to include some dynamic traffic/accident data or to set the preferences for a particular road type. For more information, refer to Transient Updates on page 348.

Returning Segment Information

In order to avoid or update the road type or speed data for a road segment you will need to know the segment ID and other road information. MapXtreme provides the ability to return the road segment data including information such as the segment ID, name, distance, road type, speed, travel time, geometries, and other important data.

By default, segment information is not returned in a route request. Use the class, and set ReturnSegmentData to true. You also have the options of defining the units of measure for both the velocity and angle for the returned information. By default the VelocityUnit is Mph (miles per hour) and the AngleUnit is Degree. The segment data is the data that is necessary to create the driving directions and contains detailed information on each segment in the route. The segment information returned includes:

- the street names (including alternate names)
- the languages for the street names
- compass direction
- segment ID
- one way boolean
- roundabout boolean
- toll road boolean
- road type
- speed limit
- turn angle
- time to traverse the segment
- distance along the segment
### Transient Updates

In order to create more accurate and meaningful route calculations, you can make temporary changes to the route data. The changes are submitted within each route request and are only valid when the server or service is handling that particular request. You can use this feature to avoid a particular highway during rush hour or lower the speed of the road segment in the request, representing real life traffic patterns, producing more accurate time based routing. A transient update can be included in any type of route request (point-to-point, multi-point, or matrix route). By making these types of modifications, you have the ability to:

- set the speed of a point, segment, or road type.
- change (increase or decrease) the speed of a point, segment, or road type by some value.
- change (increase or decrease) the speed of a point, segment, or road type by some percentage.
- set the road type for a segment.

When defining speeds in transient updates, there are essentially three types, each speed value defined differently:

- **Speed** - specifies the new speed of the segment.
- **Relative** - specifies an increase or decrease for the speed by a relative value. These values represent a change in speed. For example a value of 10, will increase the default speed by 10, while a value of -10 MPH will decrease the default speed by 10 MPH.
- **Percentage** - specifies an increase or decrease for the speed by a percentage. Values are between -100 and 100. For example a speed of 50 would increase the speed by 50 percent of the default speed (a speed of 30 MPH would increase to 45 MPH), while a speed of -50 would decrease the speed by 50 percent of the default speed (a speed of 30 MPH would decrease to 15 MPH).

### Using Transient Update to Modify a Segment

After you get a route response, you may realize that the result contains a particular route segment or group of segments (roads) where you know traffic is slower or faster, or you may want to update the road type in a request. You can use a transient update to set the speed of the road segments or road types, recalculating the route for more accurate results. Use the following classes to perform the various segment transient updates:

- **SegmentRoadTypeUpdate** - Updates the road type for the segment. The following code sample shows how to specify a new road type for the defined segment ID “S1256”,
  ```vbnet
  RoadType.MAJOR_ROAD_URBAN
  ```
  VB example:
  ```vbnet
  Public Shared Sub New_SegmentRoadTypeUpdate()
      Dim update As SegmentRoadTypeUpdate = New SegmentRoadTypeUpdate("S1256", RoadType.MAJOR_ROAD_URBAN)
  End Sub
  ```

- **SegmentSpeedUpdate** - Updates the speed of the segment with a new speed.
  The following sample shows how to define a new speed of 50 miles per hour (`Velocity(50, VelocityUnit.Mph)`) for the specified route segment (`S1256`):
  ```vbnet
  VB example:
  ```
Public Shared Sub New_SegmentSpeedUpdate()
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(50, VelocityUnit.Mph)
    ' Create the update object
    Dim update As SegmentSpeedUpdate = New SegmentSpeedUpdate("S1256", velocity)
End Sub

• **SegmentRelativeSpeedUpdate** - Updates the speed of the segment with a change in speed.

  The following sample shows how to increase the speed by 5 miles per hour (Velocity(5, VelocityUnit.Mph)) for the specified route segment (T1256):

  VB example:

  Public Shared Sub New_SegmentRelativeSpeedUpdate()
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(5, VelocityUnit.Mph)
    ' Increase the speed by 5 mph.
    Dim update As SegmentRelativeSpeedUpdate = New SegmentRelativeSpeedUpdate("T1256", velocity)
End Sub

• **SegmentPercentageSpeedUpdate** - Updates the speed of the segment with percentage of the default speed.

  The following sample shows how to increase the speed for the specified route segment by 20 percent ("S1256", 20):

  VB example:

  Public Shared Sub New_SegmentPercentageSpeedUpdate()
    ' Increase the speed by 20 percent.
    Dim update As SegmentPercentageSpeedUpdate = New SegmentPercentageSpeedUpdate("S1256", 20)
End Sub

**Using Transient Update to Modify a Point**

After you get a route response, you may realize that the result contains some locations (points) where you know traffic is slower or faster. You can use a transient update to set the speed of the road segments closest to the point, recalculating the route for more accurate results. Use the following classes to perform the various point transient updates:

• **PointSpeedUpdate** - Updates the speed of the closest segment to the point with a new speed.

  The following sample shows how to define a new speed of 50 miles per hour (Velocity(50, VelocityUnit.Mph)) for the road closest to the defined point (pt):

  VB example:

  Public Shared Sub New_PointSpeedUpdate()
    ' Create a point
Dim dpt As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint(-74, 42)
Dim pt As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt)
' Create the velocity object
Dim velocity As Velocity = New Velocity(50, VelocityUnit.Mph)
' Create the update object
Dim update As PointSpeedUpdate = New PointSpeedUpdate(pt, velocity)
End Sub

• **PointRelativeSpeedUpdate** - Updates the speed of the closest segment to the point with a change in speed.

The following sample shows how to increase the speed by 5 miles per hour (Velocity(5, VelocityUnit.Mph)) for the road closest to the defined point (pt):

VB example:

```vbnet
Public Shared Sub New_PointRelativeSpeedUpdate()
' Create a point
Dim dpt As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint(-74, 42)
Dim pt As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt)
' Create the velocity object
Dim velocity As Velocity = New Velocity(5, VelocityUnit.Mph)
' Increase the speed by 5 mph.
Dim update As PointRelativeSpeedUpdate = New PointRelativeSpeedUpdate(pt, velocity)
End Sub
```

• **PointPercentageSpeedUpdate** - Updates the speed of the closest point to the location by a percentage of the default speed.

The following sample shows how to increase the speed for the road closest to the define point by 20 percent (pt, 20):

VB example:

```vbnet
Public Shared Sub New_PointPercentageSpeedUpdate()
' Create a point
Dim dpt As MapInfo.Geometry.DPoint = New MapInfo.Geometry.DPoint(-74, 42)
Dim pt As MapInfo.Geometry.Point = New MapInfo.Geometry.Point(coordSys, dpt)
' Increase the speed by 20 percent.
```
Dim update As PointPercentageSpeedUpdate = New PointPercentageSpeedUpdate(pt, 20)
End Sub

Using Transient Update to Modify Road Type Speeds

Before determining a route, you may already know some travel information (for example, construction, accident, or rush hour), and want to reduce or increase the speed of a particular road type. For example, you can lower the speed for all highways, taking into consideration holiday weekend travel. You also have the option to lower the chance of a particular road type being used in the route calculation. When determining the fastest route, you can decrease the travel speed for a particular road type which will set those roads to a lower priority, and decrease the chance of them being used in the calculation. Use the following classes to perform the various road type transient updates:

- **RoadTypeSpeedUpdate** - Updates the speed of a particular road type with a new speed.

  The following sample shows how to define a new speed of 50 miles per hour (Velocity(50, VelocityUnit.Mph)) for all urban major roads (RoadType.MAJOR_ROAD_URBAN):

  VB example:

  ```vbnet
  Public Shared Sub New_RoadTypeSpeedUpdate()
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(50, VelocityUnit.Mph)
    ' Create the update object
    Dim update As RoadTypeSpeedUpdate = New RoadTypeSpeedUpdate(RoadType.MAJOR_ROAD_URBAN, velocity)
  End Sub
  ```

- **RoadTypeRelativeSpeedUpdate** - Updates the speed of the road type with a change in speed.

  The following sample shows how to increase the speed by 5 miles per hour (Velocity(5, VelocityUnit.Mph)) for all urban major roads (RoadType.MAJOR_ROAD_URBAN):

  VB example:

  ```vbnet
  Public Shared Sub New_RoadTypeRelativeSpeedUpdate()
    ' Create the velocity object
    Dim velocity As Velocity = New Velocity(5, VelocityUnit.Mph)
    ' Increase the speed by 5 mph.
    Dim update As RoadTypeRelativeSpeedUpdate = New RoadTypeRelativeSpeedUpdate(RoadType.MAJOR_ROAD_URBAN, velocity)
  End Sub
  ```

- **RoadTypePercentageSpeedUpdate** - Updates the speed of the road type with percentage of the default speed.

  The following sample shows how to increase the speed for all urban major roads by 20 percent (RoadType.MAJOR_ROAD_URBAN, 20):

  VB example:
Public Shared Sub New_RoadTypePercentageSpeedUpdate()
    ' Increase the speed by 20 percent.
    Dim update As RoadTypePercentageSpeedUpdate = New RoadTypePercentageSpeedUpdate(RoadType.MAJOR_ROAD_URBAN, 20)
End Sub
Web Map Service

MapXtreme provides the ability to deploy and access Web Map Services (WMS) so that you can get WMS maps from a variety of sources or provide maps to others.

In this chapter:

- Introduction to MapXtreme’s Web Map Service .................. 354
- Understanding WMS Operations ................................. 354
- Setting up a WMS Server ........................................ 356
- Configuring Layer Information for a WMS Server .......... 362
Introduction to MapXtreme’s Web Map Service

The MapXtreme Web Map Service (WMS) allows clients to request and servers to deliver spatially referenced map images over the Internet or private intranet. MapXtreme gives you the tools to both deploy a WMS Server for others to query, and supports the incorporation of WMS Client capabilities into your application to request images from a WMS Server. The WMS Server and Client implementations are based on the 1.1.1 OpenGIS® Web Map Service Implementation Specification, which can be found at http://www.opengis.org.

A basic WMS classifies its geo-referenced information holdings into layers and offers predefined styles in which to display those layers. A WMS that conforms to the OGC specification may support the transparent pixel definition for some image formats as well. Transparent pixels allow you to use retrieved WMS images as raster overlays and not solely as the background layer for a map. The quantity and quality of data available is determined by the individual WMS Server.

There is a growing amount of geo-spatial data available, provided by governments, corporations, and other organizations, that users can retrieve to enhance the accuracy and completeness of their maps. Companies with land use and water use data can add elevation and population information from the U.S. Census Bureau, or from a local data provider. Combining traffic pattern data with store location information can provide insight into establishing additional store locations or can optimize marketing and product placement efforts. As a developer, you can customize routing requests to include particular hotel, attraction, or vacation destinations by extending the WMS code. The possibilities are limited only by your need and your imagination.

MapXtreme’s WMS support is presented in the following sections:

- Understanding WMS Operations
- Setting up a WMS Server
- Configuring Layer Information for a WMS Server
- Requesting a WMS Layer

Understanding WMS Operations

MapXtreme WMS follows the 1.1.1 OGC specification for basic WMS service. There are three WMS Server specific operations: GetCapabilities, GetMap, and GetFeatureInfo.

GetCapabilities

Before requesting a WMS map, it is necessary to find out the names of available layers, styles in use, spatial information in use, and other information that the WMS server provides. GetCapabilities is an HTTP request that retrieves service-level metadata over the Internet or intranet, including the server name, the layer names, abstracts about the data and the acceptable request parameters.

In MapXtreme, GetCapabilities is the first step in collecting information from a WMS server. The capabilities are then used to request a map image.

In MapXtreme, you do not call GetCapabilities directly. Your request for a map is contained in a call using an WMSClient that takes a capabilities instance as input.
// build the capabilities
ICapabilities capabilities = WmsClientUtilities.GetCapabilities(url, "1.1.1");

// create the WMS client
WmsClient wmsClient = new WmsClient(capabilities);
wmsClient.AddLayer("WORLD");
wmsClient.Srs = "EPSG:4326";
wmsClient.BGColor = Color.Blue;
wmsClient.MimeType = "image/gif";

GetMap

Once you know the capabilities of the WMS server, a GetMap request is issued to request a map image of one or more of the WMS server’s map layers. Based on the WMS Server capabilities, a GetMap request includes the following.

• Layers—one or more layers of map data
• Styles—display styles for rendering the layers. If this is not specified, default styles will be used.
• A bounding box—the area to be included in the map image
• A coordinate reference system—a numeric representation of the map, also known as the spatial reference system). If this is not specified, the default coordinate reference system is used, as defined in the default style of the layer.
• Output format—a MIME type such as GIF and PNG, for the output map image
• Output size—height and width in pixels of the map image
• Background color—a hexadecimal red-blue-green color value such as 0xFFFFFF (required when transparency is true)
• Transparency—a true/false setting that indicates if the layer can be used as a transparent overlay to other layers.

In MapXtreme, you do not call GetMap directly. Your request for a map is contained in a call to create a TableInfoWMS, which takes a WMSclient as input. The code example below builds on the example in the GetCapabilities section.

// create the table info
TableInfoWms wmsTableInfo = new TableInfoWms("MyWmsTable", wmsClient);

// create the table
Table wmsTable = Session.Current.Catalog.OpenTable(wmsTableInfo);

// creates a FeatureLayer from the table entry
FeatureLayer featLyr = new FeatureLayer(wmsTable);

GetFeatureInfo

Once a GetMap request has been successfully completed the user may want further information about the features included in the map. The GetFeatureInfo operation returns information regarding the layers in the map and the queryable attributes of each layer. This operation is controlled by the WMS Server; the server may not offer this capability. If GetFeatureInfo is employed, it must be issued after a GetMap request is sent, as the X/Y pixel coordinate of a point on that map image is a required input of a GetFeatureInfo request.
Setting up a WMS Server

Before you can use the WMS client functionality in your applications or send HTTP requests, your application must have access to a WMS Server. You have two options: Set up your own WMS Server or send requests to an existing WMS Server.

To set up your own WMS Server you must configure a server connection to IIS and create the XML file necessary for providing the data connection required to host a Web Map Service. We assume you have a working knowledge of WMS and the MapXtreme workspace schema in this section. Information on the workspace schema is located in Appendix D: Understanding the MapInfo Workspace.

If you are only interested in accessing a WMS server to retrieve map layers, see Requesting a WMS Layer on page 364. You do not have to create a WMS Server to use the WMS Client implementation.

The WMS Server runs inside of Microsoft Internet Information Services (IIS). The following are the configuration steps for setting up a WMS Server.

**Step 1:** Create or modify a Web.config file to include the MapXtreme-specific WMS information.

**Step 2:** Create a valid WMS configuration file that contains information about the data you are hosting. This file must validate against the WMS schema file (MXP_WMSConfiguration_1_0.xsd) to avoid errors when you run the WMS Server.

**Step 3:** Configure and test the WMS Server setup.

On the MapXtreme product CD, we provide sample Web.config and WMS configuration files that you can review and modify for your own needs. The Web.config file defines how the ASP process is handled and the WMS server’s elationship to MapXtreme. The WMSSample.xml defines the data sources and layer definitions you want your WMS server to provide.

In addition, MapXtreme provides a WmsClient sample application with an example user interface for interacting with WMS servers. Find it in the MapXtreme install directory under Samples\VS2005_08\Desktop\Features\WmsClient.

The schemas for the MapXtreme workspace and WMS server are also located on the product CD.

**Step 1: Create a Web.config File**

The Web.config is a standard configuration file for a web application. Here the file is modified to provide MapXtreme-specific WMS information and to define how the ASP.NET process will be handled.

1. Create a folder to contain the Web.config and the WMSSample.xml. In this example, the location is called c:\wms.
2. Copy the Web.config and WMSSample.xml from the MapXtreme CD to this folder.
3. Open Web.config in a text editor and modify the <appSettings> line to point to the WMS configuration file.

```xml
<configuration>
  <appSettings>
    <add key="configFile" value="C:\wms\WMSSample.xml"/>
  </appSettings>
</configuration>
```
4. Farther down in the file, update the version number and PublicKeyToken (if necessary) for the MapInfo.Wms.Server and the MapInfo.CoreEngine assemblies. To find this information, from the Start menu, choose Run. In the Run dialog, type assembly to display the list of installed assemblies. In this example, our version number is 6.8.0.530.

```xml
<system.web>
    <httpHandlers>
        <add verb="GET,POST" path="*.ashx" type="MapInfo.Wms.WmsHttpHandler, MapInfo.Wms.Server, Version=6.8.0.530, Culture=neutral, PublicKeyToken=4ac3224575145b20" />
    </httpHandlers>
</system.web>
```

5. Save this file and copy it to the location you created in step 1.

### Step 2: Create a Valid WMS Configuration File for Hosted Data

The WMSSample.xml is a WMS configuration file provided with MapXtreme. This file defines information about your WMS server, including its name, title, abstract, the URL to the WMS Server and the data you want to host.

1. Open WMSSample.xml in a text editor and modify `<mxp-wms:OnlineResource>` line to point to your WMS Server.

```xml
<mxp-wms:Service>
    <mxp-wms:Name>Sample WMS Server</mxp-wms:Name>
    <mxp-wms:Title>Sample WMS Server</mxp-wms:Title>
    <mxp-wms:Abstract>This is a sample WMS server</mxp-wms:Abstract>
    <!-- The following is the URL of your WMS server, here assume it is localhost -->
    <mxp-wms:Fees>$10</mxp-wms:Fees>
</mxp-wms:Service>
```

You can modify other elements as you see fit, including the server name, title, abstract, fees, access constraints, and more.

2. Register your data layers by modifying the paths in the `<DataSourceDefinitionSet>`. Every layer you wish to serve on your WMS needs an entry. The following is a portion of the WMSSample.xml that identifies TAB files for a sample WMS Server:

```xml
<!-- The following data sources reference local TAB files, you need to replace MYPATH with the real path to those tab files. -->
<TABFileDataSourceDefinition id="id1" readOnly="false" xmlns="http://www.mapinfo.com/mxp">
    <DataSourceName>STATES</DataSourceName>
    <FileName>c:\Program Files\MapXtreme\6.x.x\Samples\Data\USA.TAB</FileName>
</TABFileDataSourceDefinition>
```
3. Describe the layers you want to host on your WMS server under the `<mxp-wms:WMSLayer>` section. You will need one entry for every WMS layer you plan to offer on your WMS server. Layers can also be nested so that by requesting a parent layer, all the child layers are included in the response.

The following example is the entry for a single layer called States. Notice it includes an SRS (spatial reference system) for its coordinate system, and a WMS style element, in this case, for an area style made up of a blue fill with red border.

For more information about how to build these entries, see Configuring Layer Information for a WMS Server.

```xml
<mxp-wms:WmsLayer>
  <mxp-wms:Name>States</mxp-wms:Name>
  <mxp-wms:Title>States</mxp-wms:Title>
  <mxp-wms:SRSNameSet>
    <mxp:SRSName>EPSG:4267</mxp:SRSName>
  </mxp-wms:SRSNameSet>
  <mxp-wms:WmsStyleSet>
    <mxp-wms:WmsStyle>
      <mxp-wms:Name>BlueFill</mxp-wms:Name>
      <mxp-wms:Title>Blue Fill</mxp-wms:Title>
      <mxp-wms:Abstract>This is a blue area fill with a red border.</mxp-wms:Abstract>
      <AreaStyle xmlns="http://www.mapinfo.com/mxp">
        <!-- The following defines the red border -->
        <LineStyle stroke="red" width="1" width-unit="mapinfo:imagesize pixel">
          <Pen>mapinfo:pen 2</Pen>
          </LineStyle>
        <!-- The following defines the blue fill -->
        <Interior fill-opacity="0" fill="#id7">
          <Defs>
            <Pattern id="id7">
              <Bitmap uri="mapinfo:brush 2">
                <ColorAdjustmentSet>
                  <ColorAdjustment color-1="nonWhite" color-2="blue" opacity="1"/>
                  <ColorAdjustment color-1="white" opacity="1"/>
                </ColorAdjustmentSet>
              </Bitmap>
            </Pattern>
          </Defs>
        </Interior>
      </AreaStyle>
    </mxp-wms:WmsStyle>
  </mxp-wms:WmsStyleSet>
</mxp-wms:WmsLayer>
```
4. Save WMSSample.xml after you are through adding feature layer information.

Step 3: Configuring and Testing the WMS Server

Once you have edited the Web.config and WMSSample.xml files, you must register your WMS server with Internet Information Services and finally, test your setup.

1. Right-click on your WMS folder (for example, c:\wms) and choose Sharing and Security. From the Web Sharing tab, choose the Share this folder radio button. If you wish to set an alias for your web server, click the Add button and supply a name in the dialog.

2. Open IIS (From the Start menu > All Programs > Control Panel > Administrative Tools > Internet Information Services). Expand the Default Web Site and locate your WMS server (by folder name or by alias, if you use one).

3. Right-click on the Web site and choose Properties. Under the ASP.NET tab, choose 2.0.50727 for the ASP.NET version (The MapXtreme assemblies are compiled under the 2.0 Framework).
4. In the same Properties dialog, under the Directory Security tab, click the Edit button at the top right. In the Authentication Method dialog box, select the Anonymous Access check box. This allows users of your WMS service to skip the username/password authentication process. Click OK twice and close the IIS window.

5. Test your setup by sending a `GetCapabilities` request from a web browser. In the address box type:

   http://localhost/wms/GetFeature.ashx?REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.1.1

   substituting your web server for `localhost`. If you have set an Alias to your web server, be sure to include that in your URL. For example:

   http://localhost/My_WMS/GetFeature.ashx?REQUEST=GetCapabilities&SERVICE=WMS&VERSION=1.1.1

   A successful test will return a web page similar to the illustration below. If the capabilities are not returned, review your configuration files to ensure everything has been entered correctly. Since you are creating the configuration files by hand, it is very easy to include typos or have missing tags.
6. If you make any changes to the WMSSample.xml after you access the WFS Server, you will need to reset IIS in order for your changes to take affect. To reset IIS, from a command prompt type:
   c:\>IISreset

7. To request a test map image, send a **GetMap** request, following the example below:
   http://My_Wms/GetMap.ashx?service=WMS&VERSION=1.1.1&SRS=epsg:4267&REQUEST=GetMap&LAYERS=States&STYLES=&BBOX=-180,0,0,90&WIDTH=800&HEIGHT=600&FORMAT=image/gif
   The WMS server returns a map image of the States layers.
Layers are described in a WMS Server configuration file according to elements laid out in the MXP_WMSConfiguration_1_0.xsd schema. These elements include:

- **Name**–a name of the layer that is used to reference the layer in requests.
- **Title**–a name of the layer that is readable by the user.
- **Abstract**–a longer narrative description of the layer.
- **SRSNameSet**–the spatial reference system(s) that applies to this layer. If more than one SRS can be used on the layer, each one can be listed. This list is unordered and applies to parent and child layers.
- **WmsStyleSet**–An unordered set of pre-defined styles supported by this layer and any child layers. The WmsStyleSet is a collection of WmsStyles, each bearing a name, title, abstract and reference to a style definition.

WMS layers can be made up of a single layer of data or a hierarchical collection of layers. A basic implementation of WMS allows the client to specify which layers it needs, as well as the coordinate system and styles for those layers that will be rendered as a map image.

Your job, as developer of a WMS Server, is to decide how to assemble your data into WMS layers. For example, you may have geographic data broken out into 10 separate .TAB files, such as boundary files, point files and polyline files. In your WMS configuration file, you can represent these 10 layers as a single WMS layer or you can keep them as individual layers. If your data is offered to a WMS client as a single layer, the client will get all 10 layers represented in a single WMS image. In this case, the client cannot separate out the information they do not want.

If, however, you provide the 10 layers as individual layers, the WMS client can choose one, some or all of those layers to be returned in a single image, or perhaps in multiple images, depending on their needs. You must decide when configuring your WMS Server how much flexibility you will offer users.

You can also nest your layers so that by requesting the parent layer, the child layers are included in the map image.

### Using a MapInfo Workspace to Build a WMS Configuration File

To build up the layer information in your WMS configuration file, consider extracting information about each layer from a previously saved MapXtreme Workspace (.MWS).

The following is an excerpt from an .MWS created with MapXtreme Workspace Manager that defines a layer called "World Countries". The .MWS layout is an XML file that follows the schema contained in MXP_Workspace_1_0.xsd.

```xml
<FeatureLayer id="id10" name="World Countries" alias="world" volatile="unknown">
    <Visibility visible="true">
        <VisibleRange enabled="false">
            <ZoomRange uom="mapinfo:length mi" minInclusive="true" maxInclusive="false">0 0</ZoomRange>
        </VisibleRange>
    </Visibility>
    <DataSourceRef ref="id4" />
</FeatureLayer>
```
Now, this same layer information is contained in a WMS configuration file. Notice the information is nearly identical, except that each tag contains "mxp:" to indicate these elements belong to the MapXtreme workspace schema (MXP_Workspace_1_0.xsd).

```xml
<mxp:FeatureLayer id="id10" name="World Countries" alias="world" volatile="unknown">
    <mxp:Visibility visible="true">
        <mxp:VisibleRange enabled="false">
            <mxp:ZoomRange uom="mapinfo:length mi"
                minInclusive="true" maxInclusive="false">0 0</mxp:ZoomRange>
        </mxp:VisibleRange>
    </mxp:Visibility>
    <mxp:DataSourceRef ref="id4" />
</mxp:FeatureLayer>
```

You can build your layers by designing them in Workspace Manager and copying the information into your WMS configuration file. See Chapter 23: Workspace Manager.

**Feature Layers and More**

You are not limited to including Feature layer information in your WMS layers. For example, you can also capture label, themes and style overrides in Workspace Manager and paste those sections in your WMS Configuration file.

Here is a portion of an .MWS that defines a graduated symbol theme for a population layer. Notice that every element that defines the theme is captured here. When you paste this into your WMS configuration file, remember to include in each element `mxp:`

```xml
<ObjectThemeLayer id="id8" name="Graduated Symbol Theme on World Capitals by Cap_Pop" alias="GraduatedSymbolThemeLayer1" volatile="unknown">
    <Visibility visible="true">
        <VisibleRange enabled="false">
            <ZoomRange uom="mapinfo:length mi"
                maxInclusive="false">0 0</ZoomRange>
        </VisibleRange>
    </Visibility>
    <DataSourceRef ref="id4" />
    <FeatureGraduatedSymbolTheme id="id9">
        <Alignment>
            <HorizontalAlignment>center</HorizontalAlignment>
            <VerticalAlignment>center</VerticalAlignment>
        </Alignment>
        <SymbolBaseSize useScale="false">
            <MapScale>1.0</MapScale>
            <DataValueAtSize>20000000</DataValueAtSize>
        </SymbolBaseSize>
        <PositiveSymbol>
            <PointStyle>
                <FontSymbol size="18" size-unit="mapinfo:length pt"
                    stroke="red" stroke-opacity="#1" text="&quot; family="MapInfo 3.0 Compatible" />
            </PointStyle>
        </PositiveSymbol>
        <NegativeSymbol visible="false">
            <PointStyle>
            </PointStyle>
        </NegativeSymbol>
    </FeatureGraduatedSymbolTheme>
</ObjectThemeLayer>
```
Requesting a WMS Layer

You can request a WMS map image programmatically, using the MapXtreme WMS client via MapInfo.Data.TableInfoWMS or through the MapInfo Professional client via File > Open Web Service > Open WMS. In both cases, you can display the returned image in a map as you would other raster layers.

A WMS map request must contain the following elements:

• The layers that should be used in rendering the image.
• The style that each layer should be rendered with.
• The coordinate system that should be used for rendering.
• The geographic area or bounds that the map should represent.
• The image format to be used. The image may have a specified background color or have a transparent background color. Transparency, when available, is useful for overlaying images in a map. An opaque image may only be useful as the bottom most layer in your map.

To learn more about the MapInfo Professional WMS client, please see the documentation provided with MapInfo Professional v7.5 or later.

Code Example: Requesting a WMS Layer

To request a WMS layer programmatically with MapXtreme, follow the code example below.

Note  MapXtreme’s implementation of WMS limits the size of the returned image to 2000 pixels each for width and height.

```csharp
// build the capabilities
ICapabilities capabilities = WmsClientUtilities.GetCapabilities(url, "1.1.1");

// create the WMS client
WmsClient wmsClient = new WmsClient(capabilities);
wmsClient.AddLayer("WORLD");
```
wmsClient.Srs = "EPSG:4326";
wmsClient.BGColor = Color.Blue;
wmsClient.MimeType = "image/gif";

// create the table info
TableInfoWms wmsTableInfo = new TableInfoWms("MyWmsTable", wmsClient);

// create the table
Table wmsTable = Session.Current.Catalog.OpenTable(wmsTableInfo);

// creates a FeatureLayer from the table entry
FeatureLayer featLyr = new FeatureLayer(wmsTable);
Web Feature Service

MapXtreme provides the ability to host and/or access map feature data from internet or private intranet Web Feature Services (WFS).

In this chapter:

- Web Feature Service ........................................ 368
- Understanding WFS Server Operations ............... 368
- Configuring a WFS Server .................................. 373
- Using the MapXtreme WFS Client Programatically . 379
Web Feature Service

MapXtreme provides a Web Feature Service (WFS) implementation to send requests over the Internet or through a private intranet to retrieve geospatial data encoded in Geography Markup Language. While a Web Map Service yields a map image, requests to a WFS Server will generate GML, a form of XML that can capture geographic data.

A basic WFS client can send three kinds of requests to a WFS server. A get capabilities request asks a server to list the geographic data it can provide and the operations that can be applied to that data. A describe feature type request asks a server to describe the data it can provide for a geographic feature. Finally the server can be asked to provide the actual data.

Geographic data from a WFS consists of descriptions about the data. At this point the retrieved data is not viewable as a map layer in MapXtreme. However, using MapXtreme’s WfsClient, the GML output can be converted into a form MapXtreme can work with: MultiFeatureCollections. These feature collections can then be treated like any other FeatureCollection in MapXtreme whereby you can apply themes, labels and perform a variety of analytical operations.

The MapXtreme WFS server implementation complies with the 1.0.0 OpenGIS® Web Feature Service Implementation Specification as a WFS Basic profile. A link to this document can be found at www.opengis.org. MapXtreme’s WFS implementation does not support the Transaction WFS specification at this time.

Requests for features via MapXtreme’s WFS implementation are made with HTTP GET or HTTP POST requests. The response is returned in GML2 by default; however, a request can explicitly ask for a response in GML3.

MapXtreme’s support for WFS consists of two parts—a WFS Server and WFS client. This chapter explains how to configure a WFS server if you want to host your own data for others to access. See Configuring a WFS Server.

If you are interested in accessing data from other WFS servers on the internet or from a private intranet, see Using the MapXtreme WFS Client Programmatically.

The following section describes the WFS Server operations available through MapXtreme WFS.

Understanding WFS Server Operations

There are three WFS Server operations that provide the basis for the MapXtreme WFS server implementation: GetCapabilities, DescribeFeatureType, and GetFeature.

GetCapabilities

A GetCapabilities request is a query of a WFS server to learn more about what the server offers in terms of geographic data and operations that can be performed on that data. The response to a GetCapabilities request is an XML document describing the operations that the WFS supports and a list of all feature types that it can service. You would request the service’s capabilities the first time you access a WFS server.

GetCapabilities is supported via HTTP GET and HTTP POST.
The following is the XML output from a GetCapabilities request. Some of the key elements are shown in **bold text**, including supported capabilities, available feature types, and filtering operations for requesting a subset of a feature type’s data.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<WFS_Capabilities xmlns:ogc="http://www.opengis.net/ogc"
xmlns:miwfs="http://www.mapinfo.com/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.opengis.net/wfs/1.0.0/WFS-capabilities.xsd" version="1.0.0" updateSequence="0"
xmlns="http://www.opengis.net/wfs">
  <Service>
    <Name>Sample WFS Server</Name>
    <Title>Sample WFS Server</Title>
    <OnlineResource>http://localhost/Wfs/GetFeature.ashx</OnlineResource>
  </Service>
  <Capability>
    <Request>
      <GetCapabilities>
        <DCPType>
          <HTTP>
            <Get onlineResource="http://localhost/Wfs/GetFeature.ashx" />
            <Post onlineResource="http://localhost/Wfs/GetFeature.ashx" />
          </HTTP>
        </DCPType>
      </GetCapabilities>
      <DescribeFeatureType>
        <DCPType>
          <HTTP>
            <Get onlineResource="http://localhost/Wfs/GetFeature.ashx" />
            <Post onlineResource="http://localhost/Wfs/GetFeature.ashx" />
          </HTTP>
        </DCPType>
      </DescribeFeatureType>
      <GetFeature>
        <DCPType>
          <HTTP>
            <Post onlineResource="http://localhost/Wfs/GetFeature.ashx" />
          </HTTP>
        </DCPType>
      </GetFeature>
    </Request>
    <FeatureTypeList>
      <Operations>
        <Query />
      </Operations>
    </FeatureTypeList>
  </Capability>
</WFS_Capabilities>
```
- `<FeatureType>`
  - `<Name>miwfs:USA</Name>`
  - `<Title>Title for usa</Title>`
  - `<Abstract>Abstract for USA</Abstract>`
  - `<Keywords>Keywords for USA</Keywords>`
  - `<SRS>epsg:4326</SRS>`
  - `<LatLongBoundingBox minx="-179.62816" miny="18.925255" maxx="-66.951403" maxy="71.42856" />`
- `<ogc:Filter_Capabilities>`
  - `<ogc:SpatialCapabilities>`
    - `<ogc:Spatial_Operators>`
      - `<ogc:BBOX` />
      - `<ogc:Equals` />
      - `<ogc:Disjoint` />
      - `<ogc:Intersect` />
      - `<ogc:Within` />
      - `<ogc:Contains` />
    - `<ogc:Spatial_Operators>`
  - `<ogc:ScalarCapabilities>`
    - `<ogc:Logical_Operators` />
  - `<ogc:Comparison_Operators>`
    - `<ogc:Simple_Comparisons` />
    - `<ogc:NullCheck` />
  - `<ogc:Comparison_Operators>`
  - `<ogc:Arithmetic_Operators>`
    - `<ogc:Simple_Arithmetic` />
  - `<ogc:Arithmetic_Operators>`
- `<ogc:Filter_Capabilities>`

**DescribeFeatureType**

After finding the available feature type with GetCapabilities request, the DescribeFeatureType request asks for detailed information about one or more of the available feature types. In MapXtreme a WFS feature type is represented by a table, and a WFS feature is equivalent to a row of information from a table. The response to a DescribeFeatureType request includes the name of the feature type (the name of the table) and the names and types of the properties (names and types of the columns) in the table. The results are returned in an XML schema document in GML format.

DescribeFeatureType is supported via HTTP GET and HTTP POST. The Schema returned will have the form whereby each property in the feature type becomes an element.

MapXtreme does not return the following column types as properties:

- MIDbType.Binary
- MIDbType.CoordSys
- MIDbType.Grid
- MIDbType.Key
- MIDbType.Raster
The following shows a portion of a DescribeFeatureType response document. In this case, the request is for a feature type called USA. The USA feature type has several properties represented by the geometry column OBJ, and data columns for State and State_Name.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<xs:import namespace="http://www.opengis.net/gml" schemaLocation="http://schemas.opengis.net/gml/2.1.2/feature.xsd" />
<xs:import namespace="http://www.opengis.net/wfs" schemaLocation="http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd" />  
<xs:element name="USA" type="miwfs:USA_Type" substitutionGroup="gml:_Feature" />
<xs:complexType name="USA_Type">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="Obj" minOccurs="0" maxOccurs="1">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="gml:_Geometry" />  
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="State" nillable="false" type="xs:string" minOccurs="0" maxOccurs="1" />
        <xs:element name="State_Name" nillable="false" type="xs:string" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
</schema>
```

GetFeature

Once you know the available feature types and their properties, the final step to retrieving WFS feature information is sending a GetFeature request. This request specifies which feature and properties to fetch. In order to retrieve a subset of features, use filtering operations to constrain the query both spatially and/or non-spatially. See Code Example: Requesting Features Using Filters.

GetFeature is supported via HTTP POST.

Among the OGC-supported parameters, Request and Typename are required. Propertyname, Filter, FeatureID and others are optional. Typename is optional when FeatureID is specified. For further details, see the OGC Web Feature Service Implementation Specification.

The following is a portion of a sample GetFeatureResponse for a feature type called USA.
<?xml version="1.0" encoding="utf-8" ?>
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.0.0/WFS-basic.xsd
http://www.mapinfo.com/wfs http://localhost/Wfs/GetFeature.ashx?REQUEST=DescribeFeatureType&SERVICE=WFS&VERSION=1.0.0&TYPENAME=miwfs:USA">
  <gml:boundedBy>
    <gml:null>inapplicable</gml:null>
  </gml:boundedBy>
  <gml:featureMember>
    <USA>
      <State>AK</State>
    </USA>
    <gml:featureMember>
      <USA>
        <State>AL</State>
      </USA>
    </gml:featureMember>
    <gml:featureMember>
      <USA>
        <State>AR</State>
      </USA>
    </gml:featureMember>
    <gml:featureMember>
      <USA>
        <State>AZ</State>
      </USA>
    </gml:featureMember>
    <gml:featureMember>
      <USA>
        <State>CA</State>
      </USA>
    </gml:featureMember>
    <gml:featureMember>
      <USA>
        <State>CO</State>
      </USA>
    </gml:featureMember>
  </gml:featureMember>
  ... 
</wfs:FeatureCollection>
Configuring a WFS Server

If you have spatial data you wish to make available to others, you first must configure a WFS server to describe what data and capabilities you are offering. There are two configuration files that you need to provide to accomplish this. This discussion assumes you have a working knowledge of schemas and web services.

You do not need to configure a WFS server if you are only interested in accessing someone else's WFS server to retrieve features. See Using the MapXtreme WFS Client Programatically on page 379.

The main configuration steps that are addressed in this section are:

**Step 1:** Create or modify a Web.config file to include the MapXtreme-specific WFS information.

**Step 2:** Create a valid WFS Server configuration file that contains information about the data you are hosting. This file must validate against the WFS schema file (MXP_WFSConfiguration_1_0.xsd) to avoid errors when you run the WFS Server.

**Step 3:** Configure and test the WFS Server setup.

On the MapXtreme product CD, we provide sample Web.config and WFS configuration files that you can use as a guide for creating your own files. The Web.config file defines how the ASP process in handled. The WFSSample.xml defines the data sources and feature definitions you want your WFS server to provide.

The schemas for the MapXtreme workspace and WFS server are also located on the product CD.

**Step 1: Create a Web.config File**

The Web.config is a standard configuration file for a web application. To use it for a MapXtreme WFS server, you must edit it to provide MapXtreme-specific WFS information and to define how the ASP.NET process will be handled.

1. Create a folder to contain the Web.config and the WFSSample.xml. In this example, the location is called c:\wfs.
2. Copy the Web.config and WFSSample.xml from the MapXtreme CD to this folder.
3. Open Web.config in a text editor and modify the <appSettings> line to point to the WFS configuration file.
   ```xml
   <configuration>
   <appSettings>
   <add key="configFile" value="C:\wfs\WFSSample.xml" />
   </appSettings>
   <configuration>
   ```
4. Farther down in the file, update the version number and PublicKeyToken (if necessary) for the MapInfo.Wfs.Server and the MapInfo.CoreEngine assemblies. To find this information on your
system, from the Start menu, choose Run. In the Run dialog, type `assembly` to display the list of installed assemblies. In our example, the version number of the assemblies is 6.8.0.530.

```xml
<system.web>
<httpHandlers>
  <add verb="GET,POST" path="*.ashx"
</httpHandlers>
<httpModules>
  <add type="MapInfo.Engine.WebSessionActivator, MapInfo.CoreEngine, Version=6.8.0.530, Culture=neutral, PublicKeyToken=93e298a0f6b95eb1"
      name="WebSessionActivator" />
</httpModules>

5. Save this file and copy it to the location you created in step 1.

### Step 2: Create a Valid WFS Configuration File for Hosted Features

The WFSSample.xml is a WFS configuration file provided with MapXtreme. This file defines information about your WFS server, including its name, title, abstract, the URL to the WFS Server and the data you want to host.

1. Open WFSSample.xml in a text editor and modify the OnlineResource line to include the URL of your WFS Server. Change `localhost` to what is appropriate for your WFS server.

```xml
<mxp-wfs:Service>
  <mxp-wfs:Name>Sample WFS Server</mxp-wfs:Name>
  <mxp-wfs:Title>Sample WFS Server</mxp-wfs:Title>
  <!-- The following is the URL of the WFS server -->
</mxp-wfs:Service>
```

You can modify other elements as you see fit, including the server name, title, abstract, fees, access constraints and more.

2. Register the tables that the WFS serves by creating a Table element for each table. The value for the `<mxp-wfs:DataSourceDefinition>` id must match the value for `<mxp-wfs:Name>`. You can include the tables in any order.

3. In the `<DataSourceDefinitionSet>`, modify the MYPATH variable to reflect the actual path to your data.

   The following is a portion of the WFSSample.xml that identifies two tables of features.

```xml
<mxp-wfs:Table>
  <mxp-wfs:Name>USA</mxp-wfs:Name>
  <mxp-wfs:Title>Title for usa</mxp-wfs:Title>
  <mxp-wfs:Abstract>Abstract for USA</mxp-wfs:Abstract>
  <mxp-wfs:Keywords>Keywords for USA</mxp-wfs:Keywords>
  <mxp:DataSourceDefinitionSet>
    <TABFileDataSourceDefinition id="USA">
      <mxp:DataSourceName>USA</mxp:DataSourceName>
      <mxp:FileName>MYPATH\USA.TAB</mxp:FileName>
    </TABFileDataSourceDefinition>
  </mxp:DataSourceDefinitionSet>
</mxp-wfs:Table>
```
4. If your WFS will be hosting data that is stored on a RDBMS, specify `<ConnectionSet>` and `<ConnectionMember>` elements, following the example below:

```xml
<mxp:ConnectionSet>  
  <mxp:DBConnection dbType="sqlserver">  
    <mxp:ConnectionName>sqlserver2k</mxp:ConnectionName>  
    <mxp:ODBCConnectionString>DRIVER={SQL Server};DATABASE=YOURDB;Server=YOURSERVER;UID=YOURUSER;PWD=YOURPASSWORD;QUOTEDID=No;Trusted_Connection=No;Network=DBMSSOCN;Address=YOURSERVER,YOURSERVERPORT</mxp:ODBCConnectionString>  
  </mxp:DBConnection>  
...</mxp:ConnectionSet>  

<mxp-wfs:Table>  
  <mxp-wfs:Name>MySQLServerTable</mxp-wfs:Name>  
  <mxp-wfs:Title>Title for MySQLServerTable</mxp-wfs:Title>  
  <mxp-wfs:Abstract>Abstract for MySQLServerTable</mxp-wfs:Abstract>  
  <mxp-wfs:Keywords>Keywords for MySQLServerTable</mxp-wfs:Keywords>  
  <mxp:DataSourceDefinitionSet>  
    <mxp:DBDataSourceDefinition id="MySQLServerTable">  
      <mxp:DataSourceName>MySQLServerTable</mxp:DataSourceName>  
      <mxp:ConnectionMember>  
        <mxp:ConnectionName>my_sqlserver2000_advserver</mxp:ConnectionName>  
      </mxp:ConnectionMember>  
      <mxp:DBQuery>  
        <mxp:Query>select * from MySQLServerTable</mxp:Query>  
      </mxp:DBQuery>  
    </mxp:DBDataSourceDefinition>  
  </mxp:DataSourceDefinitionSet>  
</mxp-wfs:Table>
```
Step 3: Configuring and Testing the WFS Server

Once you have edited the Web.config and WFSSample.xml files, you must register your WFS server with Internet Information Services and finally, test your setup.

1. Right-click on your WFS folder (for example, c:wfs) and choose Sharing and Security. From the Web Sharing tab, choose the Share this folder radio button. If you wish to set an alias for your web server, click the Add button.

2. Open IIS (From the Start menu > All Programs > Control Panel > Administrative Tools > Internet Information Services). Expand the Default Web Site and locate your WFS server (by folder name or by alias, if you set one).

3. Right-click on the Web site and choose Properties. Under the ASP.NET tab, choose 2.0.50727 from the drop-down list for the ASP.NET version (The MapXtreme assemblies are compiled under the 2.0 Framework).

4. In the same Properties dialog, under the Directory Security tab, click the Edit button at the top right. In the Authentication Method dialog box, select the Anonymous Access check box. This
allows users of your WFS service to skip the username/password authentication process. Click OK twice and close the IIS window.

5. Test your setup by sending a **GetCapabilities** request from a web browser. In the address box type:

   http://localhost/wfs/GetFeature.ashx?REQUEST=GetCapabilities&SERVICE=WFS&VERSION=1.0.0

   substituting your web server for localhost. If you have set an Alias to your web server, be sure to include that in your URL. For example:

   http://localhost/My_WFS/GetFeature.ashx?REQUEST=GetCapabilities&SERVICE=WFS&VERSION=1.0.0

   A successful test will return a web page similar to the illustration below. If the capabilities are not returned, review your configuration files to ensure everything has been entered correctly. Since you are creating the configuration files by hand, it is very easy to include typos or have missing tags.
6. To learn about the properties for the returned feature types, send a `DescribeFeatureType` request:

   http://localhost/My_wfs/GetFeature.ashx?REQUEST=DescribeFeatureType&SERVICE=WFS&VERSION=1.0.0&Typename=miwfs:USA

   This request returns a description of the properties for the feature type USA, including two column names: State and State_Name.

7. To request features from the USA table, send a `GetFeature` request.

   http://localhost/My_wfs/GetFeature.ashx?REQUEST=GetFeature&SERVICE=WFS&VERSION=1.0.0&Typename=miwfs:USA&propertyname=miwfs:State_Name

   This request returns features from the USA table as a FeatureCollection. Notice in the URL that we requested only one of the two column properties, State_Name, for the USA table. If we had not specified any property name, all properties in the table would be returned.
Using the MapXtreme WFS Client Programmatically

MapXtreme comes with a WFS Client that you can use programmatically to access data from OGC-compliant Web Feature Services. The MapXtreme WfsClient provides the ability to interact with any WFS 1.0.0 compliant Server at the API level. The WfsClient and related classes are located in the MapInfo.Wfs.Client namespace and uses the WFS Basic portion of the OGC specification. Using the WfsClient class, you can call the following methods on any WFS 1.0.0 compliant server: GetCapabilities, DescribeFeatureType, and GetFeature.

Using Filters in WFS Queries

Filters can be applied to a GetFeature request using the IFilter interface. A filter may either be spatial or non-spatial (scalar). Spatial queries such as Bbox and Within allow you to get features that exist in a certain area. Scalar filters allow you to query against specific properties of a feature type. Compound filters like AND and OR may also be used.

MapXtreme WFS supports the following filter operations.

- Spatial operators: BBox, Equals, Disjoint, Intersects, Within, Contains
- Non-spatial comparison operators: Logical operators: AND, OR, and NOT; PropertyIsEqualTo, PropertyIsGreaterThan, PropertyIsGreaterThanOrEqualTo, PropertyIsLessThan, PropertyIsLessThanOrEqualTo, PropertyIsNotEqualTo,
- Simple arithmetic operators: Add, Div, Mul, Sub

Note The HTTP protocol mandates a URL length for a GetFeature request of no more than 2048 characters. Keep in mind that a filter could easily create a URL much larger than that, which will cause an exception.
Code Example: Requesting Features Using Filters

The following C# code example illustrates how filters on WFS data can be used to return only the data you need. In this case, we query the WFS server to learn what features are available, then request that it return only features that match the population column of WorldCap with a value of 1,000,000 or greater. The output from the GetCapabilities and GetFeatures methods is displayed after the code example.

```csharp
private void doWFS()
{
    string wfsUrl = @"http://localhost/MXTWFS/GetFeatures.ashx";

    /// first we need to get the list of capabilities and list out
    /// the tables available
    ///
    MapInfo.Wfs.Client.WfsCapabilities capabilities =
    MapInfo.Wfs.Client.WfsClient.GetCapabilities(
        MapInfo.Wfs.Client.RequestMethod.GET, wfsUrl);

    MapInfo.Wfs.Client.FeatureTypeList featureTypeList =
    capabilities.FeatureTypeList;

    IList featureTypes = featureTypeList.FeatureTypes;
    foreach (MapInfo.Wfs.Client.FeatureType featureType in
        featureTypes)
    {
        System.Console.WriteLine("FeatureType: " +
            featureType.Name);
    }

    /// Now we can look at a specific table to see its contents
    ///
    string featureTypeName = "miwfs:WorldCap"; // Name taken from
    /// the above output
    MapInfo.Data.MultiFeatureCollection mfc =
    MapInfo.Wfs.Client.WfsClient.GetFeature(wfsUrl,
        new string[]{ featureTypeName }, null, "GML3", -1, null);
    DisplayFeatureCollection(mfc[0]);

    /// Now we can apply a filter
    ///
    MapInfo.Wfs.Client.IFilter filter = new
    MapInfo.Wfs.Client.PropertyIsGreaterThanOrEqualTo(
        new MapInfo.Wfs.Client.PropertyName(wfsUrl, "CAP_POP"),
        new MapInfo.Wfs.Client.Literal("1000000"));

    /// Create the Query container
    ///
    IList queries = new MapInfo.Wfs.Client.Query[]{
        new MapInfo.Wfs.Client.Query(
            new MapInfo.Wfs.Client.TypeName("http://www.mapinfo.com/wfs", "WorldCap"),
            null, filter)
    };
```
/// Run the filter and return the subset.
///
mfc = MapInfo.Wfs.Client.WfsClient.GetFeature(
    wfsUrl,
    queries,
    "GML3",
    -1);

DisplayFeatureCollection(mfc[0]);
}

The output of the above GetCapabilities code is:

FeatureType: miwfs:Ocean
FeatureType: miwfs:WldCty25
FeatureType: miwfs:World
FeatureType: miwfs:WorldCap

The first GetFeature call returns all the features (rows) in WorldCap, a portion of which is shown below. The first line shows the columns for the WorldCap data, the remaining lines show the individual rows of data.

Obj, Capital, Country, Cap_Pop, MI_Style,
Point, Abidjan, IVORY COAST, 2700000, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Abu Dhabi, UNITED ARAB EMIRATES, 722000, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Accra, GHANA, 949000, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Addis Ababa, ETHIOPIA, 1423111, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Agana, GUAM, 132726, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Algiers, ALGERIA, 1483000, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Alma-ata, KAZAKHSTAN, 1108000, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Alofi, NIUE, 3300, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Amman, JORDAN, 936000, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Amsterdam, NETHERLANDS, 694656, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
...

From the filtered GetFeature call, only those rows of data that satisfy PropertyIsGreaterThanOrEqualTo a population of 1,000,000 are returned. Notice Abu Dhabi is not included in the results since its population is listed as 722,000.

Obj, Capital, Country, Cap_Pop, MI_Style,
Point, Abidjan, IVORY COAST, 2700000, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Addis Ababa, ETHIOPIA, 1423111, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Algiers, ALGERIA, 1483000, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Point, Alma-ata, KAZAKHSTAN, 1108000, MapInfo.Styles.SimpleVectorPointStyle: 12 point,
Filtering with a Spatial Operator

The following example shows how to use a spatial query in a GetFeature request. In this case, it is asking for all the rows in the Europe table that satisfy the specified minimum bounding rectangle. This is accomplished using the spatial operator Bbox, a typical spatial operation that most WFS servers support.

```csharp
/// Now lets find rows in an MBR
///
/// WGS84 - Europe
MapInfo.Geometry.DRect bbox = new MapInfo.Geometry.DRect(-11.69, 35.36, 48.77, 65.05);
mfc = MapInfo.Wfs.Client.WfsClient.GetFeature(
    wfsUrl,
    new string[] { featureTypeName },
    bbox,
    null,
    "GML3",
    -1);
DisplayFeatureCollection(mfc[0]);
```

Creating a Map Layer from a WFS Response

If you wish to bring WFS feature data into MapXtreme as a map layer for further analysis, it is necessary for you to run code that parses the GML and creates a MultiFeatureCollection.

Parsing the WFS Response

Parsing the response is not specified in the WFS specification, nor does the specification describe an exact format for the results of a GetFeature request. The WFS specification only states that the request must be at least GML2. Since there are many versions of GML2, and in order for the MapXtreme WFS client to be able to correctly convert the GML2 response from a WFS Server, you must create a parser to convert the GML2 GetFeature response to a MapXtreme MapInfo.Data.MultiFeatureCollection. This is done by implementing the IWfsReader interface and registering that implementation with the MapInfo.Wfs.Client.WfsReaderFactory class. Registration must occur on a per URL basis, that is, if you want to interact with two WFS Servers that have the same GetFeature response, the specific IWfsReader implementation must be registered twice for each server URL. You do not need to implement the entire interface. The MapInfo.Wfs.Client.WfsReader class may be extended to override the ReadGetFeature(XmlReader) method. Any of the other methods may be overridden when interacting with a non-compliant WFS Server.
The following code example illustrates the complete process from requesting features to creating a map layer and displaying the features in a map. The code for this example is contained in the WfsClient sample application located in the \Samples folder under your MapXtreme installation.

```csharp
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Xml.Schema;

using MapInfo.Data;
using MapInfo.Engine;
using MapInfo.Mapping;
using MapInfo.Wfs.Client;

namespace MapInfo.Wfs.Client.Samples {

    /// <summary>
    /// Simple sample to demonstrate how to register a WfsReader to handle requests from a specific Wfs server, get the capabilities of the server, get the schema for a feature type located on the server and getting all of the features from the server.
    /// </summary>
    class SimpleSample {
        private const string URL = "http://www.mapinfo.com/miwfs";
        private const string defaultExportFileName = "WfsClient.gif";

        [STAThread]
        static void Main(string[] args) {

            string exportFileName = defaultExportFileName;
            if (args.Length > 0 && args[0] != null && args[0].Trim().Length != 0) exportFileName = args[0];

            // register URL with a specific WFS reader
            WfsReaderFactory.RegisterHandler(URL, typeof(WfsReader));

            // Get the WFS capabilities of the WFS server using the HTTP GET method.
            try {
                WfsCapabilities Capabilities = WfsClient.GetCapabilities(RequestMethod.GET, URL);
            }
            catch {
                MessageBox.Show("Please check if " + URL + " is a valid WFS URL");
            }
```

---

- **Chapter 22: Web Feature Service**
- **Developer Guide 383**
- **Page dimensions:** 576.0x684.0

---

The following code example illustrates the complete process from requesting features to creating a map layer and displaying the features in a map. The code for this example is contained in the WfsClient sample application located in the \Samples folder under your MapXtreme installation.

```csharp
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Xml.Schema;

using MapInfo.Data;
using MapInfo.Engine;
using MapInfo.Mapping;
using MapInfo.Wfs.Client;

namespace MapInfo.Wfs.Client.Samples {

    /// <summary>
    /// Simple sample to demonstrate how to register a WfsReader to handle requests from a specific Wfs server, get the capabilities of the server, get the schema for a feature type located on the server and getting all of the features from the server.
    /// </summary>
    class SimpleSample {
        private const string URL = "http://www.mapinfo.com/miwfs";
        private const string defaultExportFileName = "WfsClient.gif";

        [STAThread]
        static void Main(string[] args) {

            string exportFileName = defaultExportFileName;
            if (args.Length > 0 && args[0] != null && args[0].Trim().Length != 0) exportFileName = args[0];

            // register URL with a specific WFS reader
            WfsReaderFactory.RegisterHandler(URL, typeof(WfsReader));

            // Get the WFS capabilities of the WFS server using the HTTP GET method.
            try {
                WfsCapabilities Capabilities = WfsClient.GetCapabilities(RequestMethod.GET, URL);
            }
            catch {
                MessageBox.Show("Please check if " + URL + " is a valid WFS URL");
            }
```
// Do something with the the WfsCapabilities here...

// Get the schema for the USA feature type
string[] TypeNames = new string[] { "miwfs:USA" };

// Do something with the schema here...
XmlSchema usaSchema = WfsClient.DescribeFeatureType(URL, TypeNames);

// Get all features from the USA feature type
MultiFeatureCollection usa = WfsClient.GetFeature(URL, TypeNames, null, null, -1, null);
IFeatureCollection fc = usa[0];

// iterate over the Usa MultiFeatureCollection and add each
// IFeatureCollection to a MemTable, etc...
TableInfoMemTable memTableInfo = new TableInfoMemTable("myMemTable");
foreach (Column c in fc.Columns) {
    memTableInfo.Columns.Add(c);
}
Table memTable = Session.Current.Catalog.CreateTable(memTableInfo);
memTable.InsertFeatures(fc);

// create a layer from the MemTable
FeatureLayer featureLayer = new FeatureLayer(memTable);

// create the map and add the layer
Map map = Session.Current.MapFactory.CreateEmptyMap(new Size(500, 500));
map.Layers.Add(featureLayer);

// export the map to a file
using (MapExport mx = new MapExport(map))
{
    mx.Format = ExportFormat.Gif;
    mx.Export(exportFileName);
}

// clean up the map
Session.Current.MapFactory.Remove(map);
The Workspace Manager utility that comes with MapXtreme enables you to create and manage XML-based workspace files (.MWS format). The Workspace Manager’s convenient user interface allows you to easily assemble the tables that make up your map, apply many additional settings, and save the map as a workspace. Your completed workspace is then ready for use in your own applications, or for use in print or file output.

In this chapter:

- Features of the Workspace Manager ......................... 386
- Workspace Format and Contents .............................. 387
- Workspace Manager Menu Commands ......................... 387
- Layer Control .................................................. 394
- Using Workspace Manager Features ......................... 406
Features of the Workspace Manager

The Workspace Manager allows you to control most, but not all, of the settings that can be stored in a workspace file. For example, a workspace can contain information about cartographic legends and adornments; however, the Workspace Manager does not provide any options for creating cartographic legends or adornments. To create cartographic legends or adornments, use the API.

Through the Workspace Manager, you can:

- Load XML workspaces, tables, geosets, and MapInfo workspaces (.mws not .wor files).
- Save workspaces as .MWS.
- Control which tables are opened as part of a workspace.
- Create and load named connections using the Named Connection Manager.
- Add, remove, and view one or more maps.
- Set properties for map and layer visibility, layer and label styles, and themes with the built in layer control.
- Add, remove, or alter custom labels.
- Create group layers, which allow you to organize your layers into logical groupings, so that you can show or hide the entire group with a single click.
- View multiple next and previous map views.
- Use map tools for navigation and manual label placement, and use selection tools to verify that layer selectability settings are correct.
- Preview and print maps.
- Quickly open recent workspaces from the recent file list
- Create translucent effects for maps, labels, and theme layers
Create curved labels for polyline features.
Add a graticule layer to your map.
Reproject raster layers to current map window projection.
Reproject a map window made up of raster and vector layers to a different projection.

Workspace Format and Contents

The workspace file is an XML document (.MWS extension) that contains the locations, descriptions, and metadata of all the maps, tables, layers, and settings that make up the workspace. Because it is XML, the workspace is portable, which means that you can share the workspace with other users working on different computers, on different networks, across locales.

In MapXtreme, the portability of the XML workspace is implemented with named connections. Named connections enable you to define alternative drive, path, and database connection information based on your own environment, so that you can use workspaces created by others. All paths and connection strings are resolved when the workspace is opened. You can set up named connections directly in the Workspace Manager (File > Manage Named Connections). For more information on the XML workspace structure, please see Appendix D: Understanding the MapInfo Workspace.

Workspace Manager Menu Commands

This section explains each menu command available in the Workspace Manager.

File Menu Commands

The commands in the File menu provide all the standard File menu capabilities, such as opening, saving and printing files, as well as some features unique to the Workspace manager. Each command is described below.

New

Creates a new empty map to which you can then add tables using either the Add tool in layer control or the Open Tables command from the File menu. If changes have been made to the current workspace, you will be asked if you want to save the changes before the new workspace is created.

Open Workspace

Opens an existing workspace. If changes have been made to the current workspace, you will be asked if you want to save the changes before the existing workspace is opened.

Open Tables

The Open Tables command enables you to open one or more tables and add them to your map.
Save
Saves your map as a workspace.

Save As
Save a copy of the workspace to a new filename.

Close All Tables
The Close All Tables command closes all of the open tables.

Manage Tables
The Manage Tables command displays a dialog box that lists the tables that make up the map and enables you to open additional tables for possible inclusion in the map. Click Open to display the Open dialog box and open a table. The table you opened is added to the list of open tables in the Manage Tables dialog box. Then you can add the table to the map using the Add tool, which is located over the Workspace Manager layer control window.

To close a table, click a table from the list to highlight it. The Close button is activated. Click Close to close the table. Layers referencing the table you closed are removed from the map.

Manage Named Connections
A named connection describes a connection to a data source using an alias. You can create the following types of named connections: FilePath, DatabaseSource, ODBC, or Oracle OCI. After you specify the connect string or file path, you can save it as XML for later retrieval. You can set a default connection so that when you run Workspace Manager, the connection to your data source is available. Named connections are also saved to the workspace.
Page Setup

The Page Setup command enables you to specify the paper size, orientation, and margins of the printed map. You can also use this option to access printer-specific settings.

Print

The Print command enables you to print your map to paper or file output. In the Print dialog box, specify the printer to use, the page range you want to print if your job is multiple pages, and the number of copies to print. Printer properties enable you to set layout and other options that are specific to the printer you are using. The Print to file check box enables you to print your output to a file.

Print Preview

Use the Print Preview command to see how your output is going to look before you print it.

Recent Workspaces

Recent Workspaces shows a list of recently opened workspaces.

Map Menu Commands

Use the commands in the Map menu to add and remove maps, manipulate the view of the map, and create thematic maps. The view commands are also available in a popup menu. Right-click in the Workspace Manager map window to display the menu.

Add New Map

The Add New Map command enables you to create a new map window using the tables that are currently open.
Select Map

The Select Map command enables you to select which map to view in the Workspace Manager.

Remove Current Map

Removes the current map from the Workspace Manager view and deletes it from the workspace.

Change View

The Change View command enables you to change the current view of the map—that is, what area of the map is currently displayed in the window. You can change the zoom and scale of the map to your own settings. You can also set the center of the map window, or change the rotation angle.

The Change View dialog box allows you to choose the units (miles, kilometers, etc.) for the zoom width and for the center X/Y coordinates (meters, degrees, etc.). Whatever units you choose in the Change View dialog box are also used in the layer control. For example, if you want all distances in the layer control to be displayed in kilometers instead of miles, display the Change View dialog box and choose kilometers from the units list that appears next to the Zoom field.

View Entire Layer

Use the View Entire Layer command to see an entire layer, or all the layers in the map. The View Entire Layer dialog box shows a list of the layers that make up the map. Select the desired layer from the list, or choose All Layers so that all the layers are completely in view, and click OK. The map redraws to display the entire layer.

Previous View

Use the Previous View command to return to the previous view of your map.

Next View

The Next View command is available after you have used Previous View. Use it to redisplay the view of the map that was on the screen before you used the Previous View command.

The Previous View and Next View commands can be used together to toggle back and forth between two views of your map. These commands are also available as tools on the toolbar.
Preserve Scale/Zoom
Use the Preserve Scale and Preserve Zoom commands to keep the zoom and/or the scale the same as you change the size and shape of the map.

Redraw
Use the Redraw command to redraw the map.

View Selection
Use the View Selection command to zoom in or out on a selected object or objects.

Create Thematic
You can create feature themes and label themes via the Map > Create Thematic menu. Feature themes include ranged, individual value, dot density, graduated symbol, and pie and bar charts. Label themes include ranged and individual value.

If your map includes at least one set of labels (which are displayed in a label layer, you can create a label theme. A label theme assigns different label styles (different colors, etc.) to each label, based on the data in your table. For example, use a label theme to show the prominence of locations over others. A ranged label theme groups labels based on a similar data value, such as population. Cities that fall within a certain population range are labeled using one style, while cities in other ranges are labeled in another style, typically a less prominent style to indicate city size without having to label using the population value.

The Create Thematic command opens the Create Theme wizard where you can easily create a theme step by step.

To create a theme:

1. Choose Map > Create Thematic.
   The Create Theme: Step 1 of 3 dialog box appears.
2. Select either a Feature theme or a label theme, and choose the type of theme you want to create.
3. Click Next.
   The Create Theme: Step 2 of 3 dialog box appears.
4. Select the table you want to shade.
5. Choose the data you want to use. Select either a column from the table that contains the data, or select Expression to use an expression to derive the data you want from the table.
6. Click Next.
   The Create Theme: Step 3 of 3 dialog box appears. Here you can customize theme type settings, styles, and the legend. (See How to Apply Translucent Effects to Themes on page 411 for instructions on applying translucency effects to a thematic map.)
7. Click Apply to apply the customized settings.
8. Click OK when you are finished.
Tools Menu Commands

The Workspace Manager’s Tools menu provides access to the map tools via menu commands. These tools enable you to zoom in and out on the map, change the position of the map, and select map objects in various ways. These same tools are also available on the Workspace Manager toolbar. Each tool is explained below.

Zoom In

Use the Zoom In tool to get a closer area view of your map. To zoom in on a map:

1. Choose **Tools>Zoom In** to activate the tool.
   
   Your cursor changes to a magnifying glass with a plus sign in it.

2. Click on your map.
   
   The map redraws at a closer area view, centering itself at the point you clicked.

Zoom Out

Use the Zoom Out tool to get a wider area view of your map. To zoom out on a map:

1. Choose **Tools>Zoom Out** to activate the tool.
   
   Your cursor changes to a magnifying glass with a minus sign in it.

2. Click on your map.
   
   The map redraws at a wider area view, centering itself at the point you clicked.

Pan

Use the Pan tool to reposition your map without changing the zoom level. For example, you might want to redirect the view of your map so that a certain country or city is in the center. To pan your map:

1. Choose **Tools>Pan** to activate the tool.
   
   Your cursor changes to a hand icon.

2. Click on the map, and while holding down the mouse button, drag the map to the desired position.
   
   The map redraws reflecting the new position.

Select

Use the Select tool to select objects one at a time or to select all objects that are generally in the same area.

To select an object using the Select tool:

1. Choose **Tools>Select** to activate the tool.
   
   The cursor changes to an arrow.

2. Click the object on the map you want to select.
   
   The selected object is highlighted.
Radius Select

Use the Radius Select tool to select all objects that fall within a given radius. For example, you have a table of blood donors and a table of blood donation sites. Using the Radius Select tool, you could create a temporary list of blood donors that live within a one-half-mile radius of each blood donation site.

Note that the Radius Select tool selects all objects whose centroid falls within the circle. The object does not have to be completely bounded by the circle. To select objects within a radius:

1. Choose Tools>Radius Select to activate the tool.
   
   The cursor changes to a selection arrow with a small circle underneath it when you move the cursor over the map.

2. Click a place on the map that you would like to use as the center point of your radius search. For example, if you want to select all the fire hydrants that fall within two miles of a fire station, click the fire station and use that as the center point.

3. Hold down the mouse button and drag the mouse away from the center point.
   
   The Workspace Manager draws a circle around the point and reports the radius of the circle in the StatusBar (lower left corner of the screen).

4. When you have the desired radius release the mouse button.
   
   Workspace Manager highlights all map objects that fall within that circle.

Rectangle Select

Use the Rectangle Select tool to select objects within a rectangle. By clicking and dragging using the Rectangle Select tool, you create a dotted rectangle, or marquee box around objects you want to select.

Note that the Rectangle Select tool selects all objects whose centroid falls within the rectangle. The object does not have to be completely bounded by the rectangle. To select objects within a rectangle:

1. Choose Tools>Rectangle Select to activate the tool. The cursor changes to a selection arrow with a small rectangle underneath it when you move the cursor over the map.

2. Click a place on the map outside of the area you want to include in the marquee box.

3. Hold down the mouse button and drag the mouse to form a dotted rectangle around the points you want to select.

4. When you have reached the desired rectangle size release the mouse button.
   
   Workspace Manager highlights all map objects that fall within that rectangle.

Polygon Select

The Polygon Select tool selects map objects within a polygon that you draw on a map.

To select objects with the Polygon Select tool:

1. Choose Tools>Polygon Select to activate the tool.
   
   The cursor changes to a selection arrow with a small polygon underneath it when you move the cursor over the map.

2. Click the map location at which you want to place the first end point of the polygon. Move the cursor over your map in any direction.
   
   Workspace Manager draws a line from the point where you clicked to the cursor.
3. Click to create another endpoint. Continue to move the cursor and click until you have the desired number of sides to your polygon.

4. To close the polygon, make your last click as close as possible to the first click, or double-click on the last point.

Workspace Manager closes the polygon and selects the objects that are within it.

Layer Control

The Workspace Manager application window is divided into two main sections. The layer control window and commands are located on the left, and the map window is on the right. The layer control window consists of the layer tree, which displays the opened map(s) and all of its accompanying layers, and the dialog box tabs that contain map and layer settings.

The layer control features of the Workspace Manager enable you to assemble the layers of your map and apply settings to individual layers or the entire map that govern how the layer(s) or map display.

Layer Control Tools

The tools across the top of the Layer Control allow you to add, move, and remove layers from the layer tree window easily:

- The Add tools allows you to open tables, and insert group layers and label layers into your map.
- The Remove Selected Item tool removes the selected layer from the map.
• The Up and Down toolbar buttons enable you to move layers up and down the layer list, changing the order in which they are displayed.

Layer Tree

The layer control displays a tree showing the map and all layers in the map, including theme, label, and graticule layers. The layer tree allows you to perform these operations:

Map Alias

When you hover the mouse over the Map node (the node at the top of the layer tree), the tooltip shows the map alias. The tooltip aids developers who need to know the map alias (for example, you need to specify a map alias when setting properties on web controls).

Displaying Layers

The check box next to each layer on the layer tree allows you to toggle the visibility of a layer with a single click.

Changing the Layer Order

To change the order of the layers, you can select a layer and click the Up or Down toolbar button. Alternately, you can drag a layer up or down to change its position in the list.

There are several special cases that involve drag-and-drop actions:

• To move a layer into a group layer, drag the layer onto the group.
• To add a new label source to a label layer, drag a layer onto the label layer.

If you do not want to move a layer into a group—if, instead, you want to reposition the layer so that it is located above the group layer—hold down the Shift key before completing the drag-and-drop action. Similarly, if you do not want to add a new label source to a label layer, hold down the Shift key.
Displaying Context Menus

Each map or layer in the layer tree has a context menu. To display a context menu, right-click on the map or layer, or press Shift+F10 to display the menu for the currently-selected layer.

The items on the context menu depend on the type of layer specified. In particular, note the following:

- To add a style override to a layer, display the context menu and choose Add Style Override. Note that each layer can have multiple style overrides, each with a different zoom range; this allows you to set up the map so that points appear to grow larger, and roads appear to grow wider, as you zoom in.
  
  You can also add style overrides to seamless raster layers. When the style override is highlighted, you can access the Raster Image Style dialog from the Visibility tab.

- To rename any item in the layer tree, right-click the item and choose Rename. Alternately, you can press F2 to rename the selected item. Note that renaming a layer in this manner does not rename the original table; the rename operation simply changes the text that is displayed in the layer tree. The information is stored in the workspace file when the workspace is saved.

- To remove a layer in the layer tree, right-click the item and choose Remove. Alternately, you can press the Del key to delete the layer.

- By default, a layer is selectable when it is added to a map. This is controlled by the Selectable check box of the Options tab in Layer Control (see Options on page 399). However, you can designate it as the only selectable layer, and all other layers will be set to unselectable. This is very convenient if you have a map with many layers but only want one layer to have selectable features. To specify only one layer as selectable:
  a. Right-click the layer name in the Layer Control or access the context menu.
  b. Select Make This the Only Selectable Layer. This single map layer now has selectable features. That is, you can use any of the selection tools to select objects on the map. All other map layers will be unselectable.

Layer Control Tabs

The layer control tabs underneath the layer tree provide additional settings and controls that you can apply to the map as well as to each layer in the map. Different tabs control map and layer settings. The tabs that appear depend on whether the map or a layer is highlighted in the layer tree, as well as what kind of layer is highlighted. For example, when you highlight the map in the layer tree, the map tabs appear; when you highlight a layer, the layer tabs appear. If you highlight a theme, label, or graticule layer, tabs that are specific to those layers will appear. The next sections explain the options in each of the tabs.

Map Settings

When you select a map in the layer tree, the following tabs are available: View, Editing, Tools, Style, Coordinate System, and Extents.

View

The View tab enables you to control the overall appearance of the map. You can set the zoom level, scale, a center point (in degrees), and a rotation angle. Click the Apply button to apply your settings.
Editing

The options in the Editing tab enable you to control certain map editing tasks such as the styles used for drawing objects (if your application uses drawing tools), resizing objects, and moving and deleting object nodes.

![Editing tab screenshot]

The style boxes enable you to specify the default styles of any drawing tools that your application uses. Click on a box to open the corresponding style dialog box. The settings you select are saved in the workspace. When a user opens the workspace in an application that uses drawing tools, the application uses these style settings when the user draws objects on the map.

You can also specify whether you want to delete or move an object’s nodes.

Finally, you can specify an Edit mode for the map:

- None—No editing can be done on the map.
- Allow moving and resizing—Objects can be moved and resized.
- Allow node editing—Nodes can be moved or deleted.
- Allow node adding—You can add nodes to objects.

The Edit mode you select applies to all editable layers in the map.

Tools

The Tools tab enables you to control the display of InfoTips, activate Snap to Nodes and set a snap tolerance, and activate Dynamic Selection tools.

The Show InfoTips checkbox controls whether information about the feature displays in a pop-up when you hover over the feature with a select tool. See Options on page 399.

The Dynamic Selection Tools check box controls whether features are selected immediately (while you are using the selection tool) or selected when you release the mouse button to finish using the selection tool:

If the Dynamic Selection Tools check box is not selected, selection tools do not actually select any features on the map until you finish using the tool. For example, the Radius Select tool will not select any features until you specify a radius and release the mouse button.

If the Dynamic Selection Tools check box is selected, features become selected or de-selected dynamically as you drag the mouse. For example, if you use the Radius Select tool, you will see more features become selected as you drag the mouse to enlarge the radius.
If the Snap To Nodes check box is selected, map tools such as the Select tool will automatically search for nodes that are nearby. If a node is nearby, a crosshair will appear to indicate the position of the nearest node. For example, you might want to select the Snap To Node check box if you are using the Radius Search tool, and you want to make sure that the search is centered at the exact location of a point feature on your map. The Snap To Node feature is particularly important in applications that provide drawing tools, because users often need to draw features at the exact location of existing features.

The Snap Tolerance setting specifies how far the tools will search for “snappable” nodes. You can choose which layers use the Snap to Node feature. For example, you might want to turn on Snap To Node, but only have the snap crosshair appear when the cursor is near a feature in a particular layer. To turn the Snap To Node feature on or off for a specific layer, select the layer in the layer tree, then select or clear the Snap To Node check box in the Options tab.

**Style**

The options in the Style tab enable you to control translucency and anti-aliasing properties.

- **Use Anti-Aliasing**—Use this option to smooth jagged edges of lines, curves, and region borders when representing a high-definition rendition at a lower resolution. When you select Use Anti-Aliasing, Enable Translucency is also selected automatically. Whenever Enable Translucency is deselected, Use Anti-Aliasing is automatically deselected.

- **Enable Translucency**—Use this option to allow translucent values in style colors and layers when drawing the map onto the screen, printer, or file export. When translucency is enabled, you can use the translucency trackbar in style dialogs. This property has no effect on raster translucency; however, Enable Translucency must be selected to print translucent raster images.

**Note:** Rendering higher quality maps by enabling translucency and anti-aliasing, particularly in a map with three or more transparent layers, will often result in a slower rendering speed.

**Coordinate System**

The Coordinate System tab indicates the coordinate system of the map, and enables you to change the coordinate system.

To do this:

1. Click the **Coordinate System** button to display the Choose Coordinate System dialog box.
2. Select a coordinate system from the list, and click **OK**.
Chapter 23: Workspace Manager

**Extents**

In the map, the Extents tab shows the extents of the current map view. Click the View Entire Map button to see all of the map.

**Layer Settings**

When you click on a map layer, the following tabs are available at the bottom of the Layer Control.

**Visibility**

Select the Visible check box to make the map layer visible. Selecting the check box next to the layer in the layer tree has the same effect.

Select the Display Within Range check box to specify either a zoom range or scale range in which the layer appears. If you select a zoom range, specify the minimum and maximum zoom in miles. The layer appears within this range. If you select a scale range, specify the closest and farthest scale. The layer appears within this scale range.

You can also select Show Nodes, Show Centroids, and Show Line Direction to display these items on the map layer. Nodes are the points that define segments of a line or multiline or polygon. A centroid is the center of a map object. Line direction is the direction in which the line was drawn (this is helpful on street layers to indicate the proper addressing sequence). Display these elements when you wish to edit map features. The Editable checkbox is located on the Options tab.

**Options**

The Options tab check boxes facilitate editing and customizing a feature layer:

- **Selectable**—When the Selectable check box is selected, objects in the layer can be selected using either the Tool menu commands or the Selecting tools in the toolbar. Clear the Selectable check box for any layer you do not want to select from.
- **Editable**—Select the check box to make the layer editable.
- **Drawing Tools can add features to this layer**—Select this check box if you are preparing this workspace for use in an application that provides drawing tools, and you want the drawing tools to create new features in this layer.
- **Show InfoTips**—Select the Show InfoTips check box to display InfoTips when you hover over map objects in the selected layer. The InfoTip text consists of the result of the expression in the InfoTip Expression field. For example, if the expression is a column in your table, the InfoTips comprise the values from that column. If the expression is a calculation that uses column information in your table, the InfoTips comprise the results of that calculation.
- **Snap to Nodes**—Select to turn the Snap To Node feature on or off for a specific layer, select the layer in the layer tree, then select or clear the Snap To Node check box.

Settings in the Options tab are unaffected by the layer’s visibility. You will be able to see the Options tab settings regardless of whether the layer is currently visible.

**Extents**

For a layer, the Extents tab shows the extents of the selected layer. Click the View Entire Layer button to see all of the layer, or click View Default Area to see the default view of the layer.
Layer Control

**Raster Reprojection**

The Raster Reprojection tab enables you to specify reprojection settings for raster and vector layers in your map.

You can control the reprojection of both raster and vector layers. When you add either a raster or vector layer to a map, the new layer can be reprojected into the current map window projection.

When you change the projection of a map window that contains a combination of vector and raster layers, all the layers, both raster and vector, can be reprojected to the new map window projection.

These settings also exposed in the MapXtreme API. See the RasterReprojection and RasterReprojectionResampling properties in the MapInfo.Mapping.FeatureViewer class in the Developer Reference for more information.

Use the Raster Reprojection check box to turn Raster Reprojection on or off. Select the check box to turn it on; clear it to turn it off. Raster reprojection is off by default, which means that reprojection is not performed when a raster layer is added to a map.

Use the Reprojection Method settings to specify how raster reprojection is performed. Choose one of the following:

- **Optimized**—Raster reprojection for an image is determined by the look of the destination rectangle after transformation into the source image space. If the destination rectangle looks like a "rigorous" rectangle, that is, two sides of the rectangle are parallel to the x-axis and two sides are parallel to the y-axis, then the standard Windows functions stretch the source image in both directions. If the image fails the rectangle test, the reprojection is performed using the specified resampling method.
- **Always**—Raster reprojection is always performed. MapXtreme calculates the image’s coordinates based on a precise formula and then the pixels are resampled using either the Cubic Convolution or Nearest Neighbor methods.

Use the Reprojection Resampling settings to specify a resampling method for the reprojected raster image.

Resampling is a method of restoring the pixel value (usually proportional to the brightness of a spot on the ground object covered by the instant field of view of the image sensor) of the destination raster image based on the source image being a discrete representation of the initial continuous brightness field.

The pixel values of the destination image can be calculated using one of two resampling methods: Cubic Convolution and Nearest Neighbor. Choose one of the following:
• The Cubic Convolution method provides the best "restoration" of pixel values because of their discreteness. In Cubic Convolution, a pixel in the destination image is calculated based on the pixel values in a 4x4 pixel window centered at the basic pixel in the source image. The coordinates of the basic pixel are calculated for every pixel of the destination image based on a special optimized procedure. Pixels are then weighted based on the basic pixel coordinates. In general, we recommend that you use the Cubic Convolution resampling method for aerial images and satellite raster images to get a better image quality.  

• The Nearest Neighbor method replaces the pixel value in the reprojected image with the base pixel value from the source image. This resampling method takes less time to render than the Cubic Convolution method, but it may be less precise. In general, we recommend that you use the Nearest Neighbor resampling method for raster maps, grids, and scanned maps to get faster results.

Information

The Information tab provides information about the selected layer. It gives the name of the table and its path, the type of table, e.g., MapInfo table, and its coordinate system.

Theme Layer Settings

When you click on a theme layer, the following tabs are available at the bottom of the Layer Control.

Visibility

When a theme layer is selected, the Visibility tab options control the display of the selected theme. Select the Visible check box to display the theme layer; clear the check box to turn off the theme display.

Select the Display Within Range check box to specify either a zoom range or scale range in which the theme appears. If you select a zoom range, specify the minimum and maximum zoom distance. Your theme appears within this range. If you select a scale range, specify the closest and farthest scale. Your theme appears within this scale range.

Theme

The Theme tab indicates the type of thematic map and the expression used to obtain the values. The Theme tab also enables you to modify your thematic map. Click Modify Theme to change the styles or legend.

Label Layer Settings

When you click on a label layer in the layer tree, the Visibility tab is available at the bottom of the Layer Control. When you expand the label layer to see the label sources, additional tabs display that control the appearance and content of labels in label sources: AutoLabel, Style, Text, Position, and Rules.

---

Visibility

When a label layer is selected, the options in the Visibility tab control the display of labels. Select the Visible check box to display labels; clear the check box to turn off label display.

Select the Display Within Range check box to specify either a zoom range or scale range in which the labels display. If you select a zoom range, specify the minimum and maximum zoom distance. Your labels display within this range. When you use a zoom range, the maximum value is exclusive—the layer is only visible if the map’s zoom is less than the maximum value. So, if you set the maximum zoom value to 5000 miles, and then you zoom the map to exactly 5000 miles, the layer disappears.

If you select a scale range, specify the closest and farthest scale. Your labels display within this scale range.

Click Clear Label Modifications to return the labels to their default state. This button removes individual labels that were manually added with the Label tool and restores labels to their original position.

AutoLabel

The AutoLabel tab enables you to create and manage the display of autolabels. Select the Create labels automatically check box to generate autolabels for your map. Select the Display Within Range check box to specify either a zoom range or scale range in which the autolabels display. If you select a zoom range, specify the minimum and maximum zoom distance. If you select a scale range, specify the closest and farthest scale. Your autolabels display within this scale range.

Style

The Styles tab controls the style of label text and label lines. For label text, use the Text style box to access the Text Style dialog box. You can specify the font, color, background, and other text effects for the labels. For label lines, use the Line style box to access the Line Style dialog box, where you set the style of the label lines. In the Label Lines group, choose whether you want no label lines, simple lines, or lines with an arrow.

Text

The Text tab enables you to specify an expression that produces the label text from a column or derived information in the table.
Position

Use the settings in the Position tab to set the orientation, offset, and rotation of the labels.

The label’s orientation is the label’s position relative to its anchor point. Click one of the buttons to select an orientation.

Label offset is how far away a label is from its anchor point in pixels.

The label rotation is the angle at which the label is drawn. There are three Rotation settings:

- **Parallel to one segment**—Select this option if you are labeling line features such as highways, and you want each label to be drawn at an angle that will make the label run parallel to the nearest segment of the highway.
- **Parallel to multiple segments**—Select this option if you are labeling line features such as highways, and you want the label text to follow the shape of the highways (that is, you want curved labels). Turning anti-aliasing on via the Style tab in Layer Control will improve the look of the resulting curved labels.
- **Specific angle**—Specify an angle in degrees, such as zero degrees to make all labels horizontal.

Rules

The Rules tab enables you to set certain conditions for displaying labels on your map:

**Allow Duplicate Text** Select the Allow Duplicate Text check box to allow duplicate labels for different objects to display, e.g., Portland, OR and Portland, ME. This option is also used with street maps to label street segments individually.

**Allow Overlapping Text** Select the Allow Overlapping Text check box to allow labels to be drawn on top of each other. Some labels do not display because they overlap labels that have been given higher priority on the map.

**Label Partial Objects** Select the Label Partial Objects check box to label polylines and objects whose centroids are not visible in the Map window.
Per-Label Priority Expression  This expression field is optional. If you leave this expression field blank, features within a single label source are labeled in an unpredictable manner. For example, you might find that some small cities are labeled, while some major cities are not labeled because there is not enough room. If you specify an expression (which must be numeric), then the expression will be calculated for each feature on the visible portion of the map, and features that have a larger value will be given a higher labeling priority. To specify an expression, click the Set button.

For example, suppose you are configuring the labels for the WorldCapitals layer, which contains point features that represent cities. You probably want the cities with the largest population to have the highest labeling priority. In this case, you would specify a labeling expression such as:

cap_pop

The cap_pop column represents the population of each capital city. When you specify a Per-Label Priority Expression of cap_pop, you are specifying that the cities with the largest population should have the highest labeling priority. As a result, the most populous cities will be labeled first, while other cities will be labeled only if there is enough room left over.

Per-Table Priority Expression  This expression field is optional. A label layer can contain multiple label sources; for example, you might have one label source representing a set of labels for World Countries, and another label source representing a set of labels for World Capitals (cities). By default, the label source at the top of the list has the highest priority. If you want to assign a higher priority to a label source, you can either move that label source up (in the layer control's list of label sources), or you can specify a Per-Table Priority Expression for each label source.

For example, if you give the World Countries label source a per-table priority expression of 10, and give the World Capitals label source a per-table priority expression of 5, country labels will have priority over capital city labels.

Graticule Layer Settings

When you click on a graticule layer, the following tabs are available at the bottom of the Layer Control.

Visibility

Controls the visibility of the layer and the zoom range or scale within which the selected layer displays.
Chapter 23: Workspace Manager

Line/Label Properties

Controls the visibility, styles, and labeling of major and minor lines of latitude and longitude.

Line Spacing

Controls automatic or custom spacing (intervals) of major and minor lines of latitude and longitude. Custom intervals can be in degrees (default), feet, or meters.

Extents

Allows you to control the range of latitude and longitude (north, south, east, and west) over which the graticule lines extend. The selections on this tab are enabled only if Custom is selected on the Line Spacing tab.
Using Workspace Manager Features

After making any changes in this tab, click Set Extents to activate the changes.

Using Workspace Manager Features

This section shows you how to apply some of Workspace Manager’s features to enhance the cartographic quality of your maps.

Enhanced Rendering with GDI+ Translucency and Anti-Aliasing

Microsoft Windows GDI+ is the portion of the Windows XP operating system or Windows Server 2003 operating system that provides two-dimensional vector graphics, imaging, and typography. GDI+ improves on Windows Graphics Device Interface (GDI, the graphics device interface included with earlier versions of Windows) by adding new features and optimizing existing features. GDI+ rendering in MapXtreme allows you to create translucent labels, themes, and layers, as well as apply anti-aliasing that will smooth the jagged edges of lines, curves, and region borders when representing a high-definition rendition at a lower resolution.
You can enable GDI+ rendering in the API through two new properties—EnableTranslucency and SmoothingMode—in the MapInfo.Mapping.DrawingAttributes and MapInfo.Mapping.LegendDrawingAttributes classes. You can also enable these properties using the Style tab in Workspace Manager’s Layer Control.

Note: Anti-aliasing can only be used when translucency is enabled. This is enforced both by the user interface via the Workspace Manager and programmatically. When you select Use Anti-Aliasing, Enable Translucency is also selected automatically. Whenever Enable Translucency is deselected, Use Anti-Aliasing is automatically deselected.

The following maps demonstrate label translucency and anti-aliasing effects on roads and highways. Notice the smoothness of the region borders and polylines when anti-aliasing is enabled.

Before GDI+ Enhanced Rendering: After GDI+ Enhanced Rendering:

atervliet Watervliet
A translucency trackbar (TrackBarValuePicker control) in all style dialogs (Line Style, Area Style, Symbol Style, Text Style, etc.) enables you to pick a percent value between 0-100 using the sliding trackbar or the numeric selection box. However, this trackbar only works when translucency is enabled.

To programmatically change a color's translucency, use the System.Drawing.Color.FromArgb() method. This allows you to specify an alpha value for the desired color (in GDI+, the alpha channel is the portion of pixel color data reserved for transparency information). For more information, see the .NET documentation at http://msdn2.microsoft.com/en-us/library/system.drawing.color.fromargb.aspx.

Translucency in all colors is supported and tools will work properly when translucency is enabled or disabled. The Enable Translucency option has no effect on the display or export of translucent raster images, although it must be enabled to print them.

**Note:** Rendering higher quality maps by enabling translucency and anti-aliasing, particularly in a map with three or more transparent layers, will often result in a slower rendering speed.

**Creating Translucent Effects**

Applying translucent effects to the colors of map features and labels or to the color of spread styles of a theme layer enables you to see other features of the map through the color.
How to Apply Translucent Effects to a Map

The following map contains a translucent city boundary region. The partial overlap of this region over the “Watervliet” label shows the difference between the uncovered part of the label (the first two letters of “Watervliet”) and the covered part of the label. The label itself is also translucent over the rest of the map.

To add translucent effects to a layer, use a style override to change the original style of the layer.

**Note:** When you programmatically superimpose an override style modifier (OverrideType = AddNew), the first style drawn is the feature's style. Since the superimposed style is translucent, the features show through it.

1. In Workspace Manager’s Layer Control, select the "root" node in the layer control (this corresponds to the map). In the Style tab select **Enable Translucency**.
2. Select the layer you want to add translucency to, move it to the top of the list, then add a style override.
3. On the Visibility tab for the style override, click the style button that is specific to the objects in that layer. For this example click the **Area style** button.
The Area Style dialog appears.

4. Use the translucency trackbar to select the level of translucency you want to apply. Each color has its own translucency level.

**How to Apply Translucent Effects to Labels**

The previous example also showed a translucent label similar to a watermark. If you turn off visibility on the translucent city boundary region, the label looks as follows:
If you want to add a translucent label over your map as shown by this example:

1. In Workspace Manager’s Layer Control, select the "root" node in the layer control (this corresponds to the map). In the Style tab select **Enable Translucency**.
2. Select the label source in the layer control and go to the Style tab.
3. Click the **Text style** button to display the Text Style dialog.
4. Use the translucency trackbar to select how translucent you want your label color to be. The above example shows text at 80% foreground translucency.

**How to Apply Translucent Effects to Themes**

You can apply translucent effects to thematic maps. The following example shows a ranged theme applied to a city boundary layer. The layer is positioned on top of the rest of the layers.

If translucency is enabled, you can select a translucent value for the start and end theme ranges (bins), and automatically spread the color. This will also automatically spread the translucent value between the start and end theme bins. For example, the theme in the above map has a translucency value of 75% for the start bin (gray) and 50% for the end bin (red). Since this theme has 3 bins, the middle bin is automatically given a translucency value of 63% (50 through 75 spread equally).

**Curved Labels**

Curved labels are labels that follow the curve of a line. They enhance the appearance of map features that are made up of lines, such as streets and rivers.

Curved labels are generated in the Workspace Manager by selecting the label layer, then selecting **Parallel to Multiple Segments** in the Position tab. For information on rendering curved labels via the API, see the ILayout Interface section of the MapXtreme Developer Reference.
MapXtreme attempts to create a curved label for every arc and polyline record in a map, just as it does for non-curved labels. For example, in street maps, the street can be made of several polylines or one long polyline. The length and number of the polylines, the rules that govern whether a curved label can be created, and the labeling options you choose, all affect which curved labels are created and where they display.

Some polyline and arc segments in your layer data may not contain label name entries. When this occurs, MapXtreme cannot display labels for that segment.

Several rules determine whether MapXtreme can display a curved label:

- MapXtreme can only draw curved labels using TrueType fonts. If you select a non-TrueType font, a comparable TrueType font is substituted and the label you chose may display differently than expected. Also, if you change a horizontal label using a non-TrueType to a curved label, the new label may display differently due to the font substitution.
- Part of the label string must fit along the arc or polyline that it is labeling. If it cannot fit, MapXtreme determines that the label is too long and discards it.
- MapXtreme cannot draw curved labels for polylines that are very jagged, however, it depends on the curvature of the line.
- Although a street segment appears to be straight, the label may be curved. This happens because the polyline data for the street segment contains a curve that is not visible at the current zoom level. The label is following the curve of the street even though the curve isn’t visible. If you zoom in on the map to a close enough distance, you will be able to see the curve in the street.
- Labels that curve onto themselves are discarded and do not display.
- Curved labels follow the same rules for overlap detection, duplicate text, and partial segment labeling as non-curved labels. Each of these rules affect how and when the labels are displayed.
- You can create curved labels with the Label tool at any point along an arc or polyline.
- You cannot drag curved labels as you can other labels; however, you can reposition them with the Label tool.
- The Label Lines controls are disabled for curved labels.
- Curved and non-curved labels persist for layers in the workspace.
- You cannot underline curved labels.
- Curved labels are always drawn smoothly whether or not translucency and anti-aliasing are enabled.

**Curved Labels Created in MapInfo Professional**

Curved label display settings available in MapInfo Professional can be saved to an .MWS workspace and loaded into MapXtreme. Changes to the position of a curved label made in MapInfo Professional can also be saved to an .MWS workspace and loaded into MapXtreme. The following describes the display settings and position modifications that MapXtreme can load:

- Although MapXtreme does not support creating label lines for curved labels in Workspace Manager, it can load label lines created in MapInfo Professional or if label lines are created using a customized label tool.
- MapXtreme can load callout lines for labels that contain a custom (user-defined) location for the line endpoint. For example, if you move the endpoint of a label in MapInfo Professional and save it to a workspace (.MWS), the endpoint is positioned as expected when the workspace is opened in MapXtreme. MapXtreme loads and saves custom line ends but does not support creating them.
Curved labels created in MapInfo Professional and saved to an MWS display in the same location in MapXtreme.

Labels that have been dragged to a new location in MapInfo Professional can be displayed as curved labels in MapXtreme. When the label changes are saved to an .MWS workspace and the workspace is opened in MapXtreme, the labels will display as curved labels when the curved label option is turned on. Note, however, if the label in MapInfo Professional was dragged and then rotated prior to being saved to the workspace, MapXtreme will not curve the label. This is because MapInfo Professional has modified the angle of the label, which MapXtreme honors over the curved label setting.

Creating Curved Labels

To position labels along a curve:

1. In Workspace Manager, open the map you want to change the labels for.
2. Highlight the layer source that contains the labels you want to change.
3. Verify that the labels are visible.
4. Click the Position tab. In the Rotation section, select **Parallel to multiple segments** to display the labels along the curve of the line.

5. If necessary, use the Orientation buttons on the Position tab to set the label’s position relative to its anchor point. When you select:
   - **Left**, the curved labels are left-justified starting at the beginning of the arc/polyline
   - **Center**, the curved labels are centered on the midpoint of the arc/polyline
   - **Right**, the curved labels are right-justified at the end of the arc/polyline

**Note** The length of the polyline(s) affect how the label is positioned. The longer the polyline(s), the more predictably the labels display.
**Repositioning Curved Labels**

You can use the Label tool on the Main toolbar to reposition curved labels. Make sure you have already selected the **Parallel to multiple segments** option on the Position tab for the label source you want to change.

1. Highlight the layer you want to move the labels for in the list.
2. In the Options tab, select the **Selectable** check box.
3. Highlight the label source where you want to move curved labels.
4. Click (the Label tool) on the Main toolbar.
5. Click the line on which you want to reposition the label.
6. Click the new location for the label until the label is positioned where you want it.

**Note**  If the segment you select does not have a label name associated with it in the data, no label is displayed.

**Graticule Layers**

Graticules are grids (lines of latitude and longitude) that overlay the map, spaced at a regular distance (for example, every five degrees, every fifteen degrees). They are used to establish a frame of reference.

In many ways, graticules behave just like other layers. However, a graticule layer differs from other layer types in the following ways:

- Graticule layers are not editable like other types of layers. That is, you cannot add new features to them.
- Although you can configure graticule label style and position, you cannot create a label layer on a graticule layer.
- You cannot create a thematic layer on a graticule layer.
Chapter 23: Workspace Manager

Adding Graticule Layers

You can add graticules directly from the Layer Control (right-click on the map node to access the context menu):

By default, the graticule spacing and extents are determined based on the zoom and size of the map window. So when you add a graticule layer to a map, graticule lines appear regardless of the zoom level.

Managing Graticule Layers

After you have added a graticule layer, you can manage and customize the layer from the Layer Control in a number of ways.

• Control graticule layer visibility and zoom/scale settings
• Show and configure spacing of major and minor graticule lines
• Control visibility and style of graticule labels and lines
• Specify the extents (in degrees, feet, or meters) of the graticule (east, west, north, south).
• Move the layers up and down in the Layer Control order
• Remove the graticule layer (the context menu is the only way to remove a graticule layer)
• Create multiple graticule layers and Group Layers

See Graticule Layer Settings on page 404 for more information.

Graticule layers are indicated by a graticule icon, as shown below:
Using the GeoDictionary Manager

The GeoDictionary Manager is a utility that comes with MapXtreme to aid in your Mapping applications. This chapter deals with how to use this tool.

In this chapter:

- Using the GeoDictionary Manager .................. 418
- Changes in the GeoDictionary Manager ............. 418
- The GeoDictionary Manager’s User Interface ........ 418
- The GeoDictionary File .............................. 421
Using the GeoDictionary Manager

The GeoDictionary Manager application is designed to support the manipulation of a GeoDictionary file. The GeoDictionary is an XML file containing registration information about the MapInfo tables that can be matched by your application during automatic databinding. Only MapInfo tables that can or will be matched against should be registered in the GeoDictionary.

There is no need to register every .tab file that an application uses in the GeoDictionary, and in fact there is some overhead in having unnecessary files registered. You only need to register those tables against which you would like to match.

Changes in the GeoDictionary Manager

The GeoDictionary Manager for MapXtreme is, in appearance, very similar to the utility that was included in MapX and previous versions of MapXtreme. The major difference is that now the underlying file (*.dct) is an XML file and completely editable using a text or XML editor.

The GeoDictionary Manager’s User Interface

This section describes the user interface for the GeoDictionary Manager.

Run GeoDictionary Manager

To run the GeoDictionary Manager when you want to manually register layers:

• Start -> All Programs -> MapInfo -> MapXtreme -> GeoDictionaryManager.

The GeoDictionary Manager is displayed. The file last opened is reloaded upon start-up.
Set the matching table properties for a given table using the Properties dialog box. To access the Properties Dialog box, click the Properties button, or double-click on any table name in the Registered Tables list.
## Table Properties

<table>
<thead>
<tr>
<th>Part</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapInfo Table</td>
<td>Read-only edit box containing the file name of the MapInfo table if it is located in the same directory of the GeoDictionary, or the relative pathname to the file if it is not. If the file is located on a different drive or volume, the full path is displayed.</td>
</tr>
<tr>
<td>Description</td>
<td>This field provides a mechanism for changing the friendly name for the table. This control is defaulted to the Description tag in the .TAB file, or the filename if the Description tag is not found, but can be changed by the user. Note that changes to the description in the GeoDictionary Manager are only be stored in the GeoDictionary and is not reflected in the table itself. This allows the GeoDictionary Manager to easily work with read-only data, e.g., data on CD-ROMs.</td>
</tr>
<tr>
<td>Match Percentage</td>
<td>This field is initially populated with the default value for the GeoDictionary and can be altered for specific tables by changing this value. Values must be between 1 and 100.</td>
</tr>
<tr>
<td>Field Information</td>
<td>This list box contains a list of all the indexed columns in the table. If the box for a given column is checked, that field is searched during the matching process.</td>
</tr>
</tbody>
</table>
Appendix 24: Using the GeoDictionary Manager

The GeoDictionary File

The GeoDictionary file (*.dct) can be manipulated manually if you choose to skip using the GeoDictionary Manager. If you understand and can write XML easily then this is a viable alternative to using the GeoDictionary Manager. The .dct file is straight XML.

Sample .dct file

The following is a sample GeoDictionary file, GeoDic_US.dct. This file is a very simple GeoDictionary provided merely as an example of the structure.

<?xml version="1.0" encoding="Windows-1252" standalone="yes"?>
<!--GeoDictionary file-->
<GeoDictionary>
  <DefaultMatchThreshold>80</DefaultMatchThreshold>
  <MatchTables>
    <MatchTable>
      <TablePath>US_CNTY.TAB</TablePath>
      <TableDescription>US county boundary</TableDescription>
      <RefineTableName>USA.TAB</RefineTableName>
      <MatchThreshold>90</MatchThreshold>
      <AssociatedMaps>
        <AssociatedMap>USA.GST</AssociatedMap>
      </AssociatedMaps>
    </MatchTable>
    <MatchFields>
      <MatchField>
        Table used to refine searches
        This field allows the user to set a refining table to determine exact match for data that is not unique in a particular index.
        Some tables, e.g., US Counties, contain indexed columns that are not unique. In that situation, a refining table is necessary to determine an exact match for data. If the table has non-unique indexed columns, use this field to specify a table to use to match against to find unique entries.
      </MatchField>
      <MatchField>
        Associated maps that this table belongs to
        This list box shows the particular Workspaces, GeoSets, or other files of which this particular table is a part.
      </MatchField>
      <MatchField>
        Add
        This displays a common file picker to allow you to choose GeoSets, Workspaces, or other files to associate with the particular Tab file. Selecting one or more files adds them to the list of associated maps. You can select multiple files by holding down the shift or control keys while clicking.
      </MatchField>
      <MatchField>
        Remove
        Deletes the selected geoset, workspace, or other filename from the list of associated maps.
      </MatchField>
    </MatchFields>
  </MatchTables>
</GeoDictionary>
The elements in the structure correspond to individual fields and controls in the GeoDictionary Manager as defined in the following table:

<table>
<thead>
<tr>
<th>XML Element</th>
<th>Table Properties Dialog Box Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeoDictionary</td>
<td>none</td>
<td>The root element of the file</td>
</tr>
<tr>
<td>DefaultMatchThreshold</td>
<td>none</td>
<td>The DefaultMatchThreshold is the threshold of matching when a specific MatchThreshold is not defined for a particular table. This value cannot be set using the GeoDictionary Manager.</td>
</tr>
<tr>
<td>MatchTables</td>
<td>none</td>
<td>The container elements for the registered MatchTables. There should be a single MatchTable element for each item in the Registered Table list.</td>
</tr>
<tr>
<td>MatchTable</td>
<td>MapInfo Table.</td>
<td>The filename for each table to be matched.</td>
</tr>
<tr>
<td>TablePath</td>
<td>MapInfo Table.</td>
<td>The relative path for each MatchTable.</td>
</tr>
<tr>
<td>TableDescription</td>
<td>Description field (also the name that appears in the Registered Tables list in the main dialog box.)</td>
<td>The friendly name for the MatchTable.</td>
</tr>
<tr>
<td>RefineTableName</td>
<td>Table used to refine searches</td>
<td>The related table with which to refine matching.</td>
</tr>
<tr>
<td>MatchThreshold</td>
<td>Match Percentage</td>
<td>The match threshold for this MatchTable. If none is specified, the DefaultMatchThreshold is used.</td>
</tr>
<tr>
<td>AssociatedMaps,</td>
<td>Associated maps to which this table belongs.</td>
<td>This list box shows the particular Workspaces and GeoSets of which this particular table is a part.</td>
</tr>
<tr>
<td>AssociatedMap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MatchFields</td>
<td>none</td>
<td>This element is a container for MatchField and FieldName elements.</td>
</tr>
<tr>
<td>MatchField, FieldName</td>
<td>Field Information list box.</td>
<td>Each item in the list box that is checked to be searched for automatch and binding.</td>
</tr>
</tbody>
</table>
Migrating to MapXtreme

This section is for existing users of MapX to become familiar with the .NET-based object model in MapXtreme and how it differs from the architecture of MapX.

In this appendix:

- Comparing MapXtreme’s Object Model to MapX .............. 424
Comparing MapXtreme’s Object Model to MapX

If you have used MapX and MapXtreme for Windows, you will find similarities and differences in the way they work from MapXtreme. This section describes some of the major differences between MapXtreme and previous versions of our APIs.

Specific Object Model Implementation Differences

The Map object from MapX is now represented by 3 or 4 classes. The Map contains the layers, area of interest (view), Adornments, Legends, regardless of how it is being viewed. The MapControl holds the Map object and acts as a control for the Map and tools which interact with the Map. There is a MapControl for both Windows Forms (desktop) applications and ASP.NET (web) applications. The MapExport class can be used to export a Map to a file or stream. The Session holds the collection of Maps, Selections, and Tables. Cartographic scale is now fully supported. Label layers can be used to better control the position of labels. Group Layers allow for treating many layers as one in the layer hierarchy.

The MapX Datasets concept has been replaced by a more flexible set of options in the MapInfo.Data namespace. MapXtreme uses ADO.NET for data access. ADO.NET allows a wide variety of data formats to be accessed with very similar code, facilitating easy transition between data sources. We have added a wider variety of cursors for forward and backward access through a database table.

MapXtreme uses a table-centric data model. In previous versions of our API, maps were made up of layers, but there was no specific table class. In the MapXtreme object model, a table is the central object used for accessing data. The Session.Catalog is used to open and enumerate tables. Layers can reference tables, but all table specific methods were moved from Layer to table. Also, Searching is done on tables in MapXtreme and not layers.

Thematics that modify styles, such as ranged themes and individual value themes, no longer use their own layer. In the previous models such a theme occupied a separate layer and worked independently of other layers of a map. Object themes still retain their own layer and can now be displayed in any way you choose, independent of the original map objects on which the theme was based.

Geometry has been changed to be a hierarchical model with much deeper access to properties of the objects. Styles are also represented in a hierarchy.

A larger quantity of map tools have been added for you to use. Their extensibility and functionality has been enhanced to make it much easier to develop customized tools. There are also tools for use in web applications that are also extensible.

For web applications, the MapXBroker and MapXServer are no longer needed. Instead MapXtreme uses standard COM+ to pool the Session object.

Application state is managed via serialization and the ASP.NET state server. See Chapter 6: Understanding State Management for more details about state management and serialization.

The following table lists the objects in the MapX 5.0 object model and the equivalent in the current object model for MapXtreme. Note that with the re-architecture of such an extensive product as MapXtreme, the equivalent may approximate.
<table>
<thead>
<tr>
<th>MapX 5.0</th>
<th>MapXtreme</th>
</tr>
</thead>
<tbody>
<tr>
<td>AffineTransform</td>
<td>AffineTransform</td>
</tr>
<tr>
<td>It is non-mutable (no set).</td>
<td>Use CoordSysFactory.CreateAffineTransform to</td>
</tr>
<tr>
<td></td>
<td>create one.</td>
</tr>
<tr>
<td>AllFeaturesConstraint</td>
<td></td>
</tr>
<tr>
<td>Annotation</td>
<td>Obsolete. New Adornments class. The MapControl</td>
</tr>
<tr>
<td></td>
<td>contains adornments derived from Control</td>
</tr>
<tr>
<td>Annotations</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>BindLayer</td>
<td>Obsolete. Replacement is in Table.AddColumns()</td>
</tr>
<tr>
<td></td>
<td>arguments.</td>
</tr>
<tr>
<td>BitmapSymbol</td>
<td>BitmapPointStyle</td>
</tr>
<tr>
<td>BitmapSymbols</td>
<td></td>
</tr>
<tr>
<td>BoundsConstraint</td>
<td></td>
</tr>
<tr>
<td>CoordSys</td>
<td>CoordSys</td>
</tr>
<tr>
<td>It is non-mutable (no set).</td>
<td>Use CoordSysFactory to create one.</td>
</tr>
<tr>
<td>Dataset</td>
<td>Obsolete.</td>
</tr>
<tr>
<td></td>
<td>Replacements:</td>
</tr>
<tr>
<td></td>
<td>• Expression columns: Table.AddColumns() allows</td>
</tr>
<tr>
<td></td>
<td>you to create expression columns.</td>
</tr>
<tr>
<td></td>
<td>• Manual Data binding (both dynamic or static):</td>
</tr>
<tr>
<td></td>
<td>Table.AddColumns() using a Table opened w/ an</td>
</tr>
<tr>
<td></td>
<td>TableInfoClient which points to a ADO.NET Data</td>
</tr>
<tr>
<td></td>
<td>Table.</td>
</tr>
<tr>
<td></td>
<td>• Auto Data Binding: Handled by Geodictionary</td>
</tr>
<tr>
<td></td>
<td>functionality.</td>
</tr>
<tr>
<td></td>
<td>• Non ADO.NET External Data</td>
</tr>
<tr>
<td></td>
<td>• XY Binding: SpatialSchemaXY applied to the</td>
</tr>
<tr>
<td></td>
<td>TableInfo.SpatialSchema property.</td>
</tr>
<tr>
<td></td>
<td>• PointRef Binding: SpatialSchemaPointRef</td>
</tr>
<tr>
<td></td>
<td>applied to the TableInfo.SpatialSchema property.</td>
</tr>
<tr>
<td></td>
<td>RowValues: MIDataReader, MIScrollableReader,</td>
</tr>
<tr>
<td></td>
<td>see MCommand.</td>
</tr>
<tr>
<td></td>
<td>Refresh: Table.RefreshColumns()</td>
</tr>
<tr>
<td>Datasets</td>
<td>Obsolete. See Dataset.</td>
</tr>
<tr>
<td>Datum</td>
<td>Datum</td>
</tr>
<tr>
<td>It is non-mutable (no set).</td>
<td>Use CoordSysFactory.CreateDatum() to create</td>
</tr>
<tr>
<td></td>
<td>one.</td>
</tr>
<tr>
<td>Feature</td>
<td>Geometry, style and key information is a</td>
</tr>
<tr>
<td></td>
<td>Feature.</td>
</tr>
<tr>
<td>Features</td>
<td>See Feature.</td>
</tr>
</tbody>
</table>
### Comparing MapXtreme’s Object Model to MapX

<table>
<thead>
<tr>
<th>MapX 5.0</th>
<th>MapXtreme</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeaturesConstraint</td>
<td>Table-level object processing</td>
</tr>
<tr>
<td></td>
<td>• Buffer: For one feature, it is at the Geometry level (FeatureGeometry.Buffer). For multiple features, use FeatureProcessor.Buffer.</td>
</tr>
<tr>
<td></td>
<td>• Combine: For one feature, it is at the Geometry level (FeatureGeometry.Combine). For multiple features, use FeatureProcessor.Buffer.</td>
</tr>
<tr>
<td></td>
<td>Intersect: For one feature, it is at the Geometry level (FeatureGeometry.Intersect). For multiple features, use FeatureProcessor.Intersect.</td>
</tr>
<tr>
<td>Field</td>
<td>Fields</td>
</tr>
<tr>
<td>Find</td>
<td>Find</td>
</tr>
<tr>
<td>FindFeature</td>
<td>FindResult</td>
</tr>
<tr>
<td>FindMatch</td>
<td>FindCloseMatch, FindAddressRange</td>
</tr>
<tr>
<td>FindMatches</td>
<td>FindCloseMatchEnumerator, FindAddressRangeEnumerator</td>
</tr>
<tr>
<td>FindResult</td>
<td>FindResult</td>
</tr>
<tr>
<td>Geoset</td>
<td>Obsolete. Similar functionality: MapLoader</td>
</tr>
<tr>
<td>Geosets</td>
<td>Obsolete. MapXtreme opens geosets.</td>
</tr>
<tr>
<td>Graphic</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>IndividualValueCategory</td>
<td>IndividualValueThemeBin</td>
</tr>
<tr>
<td>IndividualValueCategories</td>
<td>ModifierThemeBins</td>
</tr>
<tr>
<td>Label</td>
<td>LabelLayer, LabelSource</td>
</tr>
<tr>
<td>Labels</td>
<td>LabelLayer, LabelSource</td>
</tr>
<tr>
<td>LabelProperties</td>
<td>LabelProperties</td>
</tr>
</tbody>
</table>
### Appendix A: Migrating to MapXtreme

<table>
<thead>
<tr>
<th>MapX 5.0</th>
<th>MapXtreme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer</td>
<td>UserDrawLayer, LabelLayer, FeatureLayer, ObjectThemeLayer, GroupLayer...</td>
</tr>
<tr>
<td>• AddFeature: The current equivalent is to</td>
<td>bind a Geometry to the geometry column during an insert operation using</td>
</tr>
<tr>
<td>attach a Geometry to the geometry column</td>
<td>MICommand.</td>
</tr>
<tr>
<td>• AllFeatures</td>
<td></td>
</tr>
<tr>
<td>• AutoLabel: This no longer exists. The</td>
<td>equivalent is to have the</td>
</tr>
<tr>
<td>equivalent is to have the</td>
<td>LabelSource.DefaultLabelProperties.Enabled property set to true.</td>
</tr>
<tr>
<td>• Begin/EndAccess: Not addressed</td>
<td></td>
</tr>
<tr>
<td>• Bounds: FeatureLayer.Bounds</td>
<td></td>
</tr>
<tr>
<td>• ClearCustomLabels: Not addressed</td>
<td></td>
</tr>
<tr>
<td>• ClippedBounds: Obsolete. Use Map.SetView()</td>
<td></td>
</tr>
<tr>
<td>• CoordSys</td>
<td></td>
</tr>
<tr>
<td>• Datasets: Obsolete. A layer has a Table</td>
<td></td>
</tr>
<tr>
<td>(FeatureLayer.Table.)</td>
<td></td>
</tr>
<tr>
<td>• DeleteFeature: The equivalent is to to</td>
<td>delete row through MICommand.</td>
</tr>
<tr>
<td>delete row through MICommand.</td>
<td></td>
</tr>
<tr>
<td>• DrawLabelsAfter: This concept is</td>
<td>deprecated. Similar functionality is to use an LabelLayer instance</td>
</tr>
<tr>
<td>deprecated. Similar functionality is to</td>
<td>after each Layer. The only drawback is that each LabelLayer works off</td>
</tr>
<tr>
<td>use an LabelLayer instance after each</td>
<td>its own label cache so the labels will not interact between layers.</td>
</tr>
<tr>
<td>Layer.</td>
<td></td>
</tr>
<tr>
<td>• Editable: FeatureLayer.Editable</td>
<td></td>
</tr>
<tr>
<td>• FeatureIDFromFeatureName: Equivalent would</td>
<td>be an SQL command through MICommand.</td>
</tr>
</tbody>
</table>
### Comparing MapXtreme’s Object Model to MapX

<table>
<thead>
<tr>
<th>MapX 5.0</th>
<th>MapXtreme</th>
</tr>
</thead>
</table>
| Layer (continued) | • Find: Find  
| | • GetDrillDownFeaturesByID: Not addressed  
| | • GetFeatureByID: Equivalent functionality is a key cursor using MICommand.  
| | • Invalidate: IMapLayer.Invalidate  
| | • KeyField: Obsolete. Equivalent functionality is MICommand.  
| | • LabelAtPoint  
| | • Labels  
| | • Name: FeatureLayer.Name or Alias.  
| | • NoFeatures  
| | • OverrideStyle: Equivalent is the FeatureLayer.Modifiers collection.  
| | • Pack: Equivalent is FeatureLayer.Table.Pack  
| | • PredominantFeatureType: MISpatialColumnInfo.PredominantObjectType. You can get the column from the Table.TableInfo.ColumnInfos property.  
| | • Refresh: Flushes the cache. Equivalent: Table.Refresh()  
| | • Search: Equivalent is MICommand. The result, however, can contain more than just feature keys.  
| | • SearchAtPoint: No “easy” equivalent.  
| | • SearchWithinDistance: No “easy” equivalent.  
| | • SearchWithinRectangle: No “easy” equivalent.  
| | • SearchWithinFeature: No “easy” equivalent.  
| | • Selectable: FeatureLayer.Selectable.  
| | • Selection: Selection class in Session.Sessions collection.  
| | • ShowCentroids: FeatureLayer.ShowCentroids  
| | • ShowLineDirection: FeatureLayer.ShowLineDirection  
| | • ShowNodes: FeatureLayer.ShowNodes  
| | • Style: Equivalent is now the FeatureLayer.Modifiers collection.  
| | • SupportsPack: Table.SupportsPack  
| | • Type: IMapLayer.Type  
| | • UpdateFeature: Equivalent through MICommand  
| | • Visible: IMapLayer.Enabled. IMapLayer.Visible takes into account:  
| | • ZoomLayer: IMapLayer.ZoomRangeEnabled  
| | • ZoomMax: IMapLayer.ZoomRange.End  
| | • ZoomMin: IMapLayer.ZoomRange.Start |

<table>
<thead>
<tr>
<th>Layers</th>
<th>Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AnimationLayer equivalent: GroupLayer and BackingStoreState.Off</td>
</tr>
<tr>
<td></td>
<td>InsertionLayer equivalent: AddMapTool.InsertionLayer</td>
</tr>
<tr>
<td></td>
<td>Selections: Live in the Session object.</td>
</tr>
</tbody>
</table>

| LayerInfo | Obsolete. Equivalent is TableInfo |
## Appendix A: Migrating to MapXtreme

<table>
<thead>
<tr>
<th>MapX 5.0</th>
<th>MapXtreme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legend</td>
<td>Legend</td>
</tr>
<tr>
<td>LegendText</td>
<td>LegendRow</td>
</tr>
<tr>
<td>LegendTexts</td>
<td>LegendRows</td>
</tr>
</tbody>
</table>
Comparing MapXtreme’s Object Model to MapX

MapX 5.0 | MapXtreme
--- | ---
Map | Map has now been separated into two entities: Map and MapControl.

MapControl contains functionality applicable to the System.Windows.Forms.Control; an object that can be embedded in a WinForm.

Map contains the actual mapping functionality, regardless of how it is being viewed (whether it is a WinForm or an export.)

The following is a list of properties/methods in the former Map class and how they are being handled in the MapXtreme model:

- Annotations: Obsolete.
- CenterX/Y: Map.Center
- ConvertCoord: DisplayTransform.ToDisplay(), DisplayTransform.FromDisplay()
- CreateCustomTool: Inherit from MapTool or any of the Custom* tools available that provide marquee-line drawing capabilities using the mouse.
- Datasets: Obsolete. Tables are accessible through the MICatalog.
- Dataset: Obsolete.
- DatasetGeoField: Obsolete. See Map.Dataset.
- DatasetTheme: Obsolete. See Map.Dataset.
- DefaultStyle
- DisplayCoordSys: Map.GetDisplayCoordSys().
- Distance: CoordSys.Distance().
- DynamicSelectionSupport:
  - MapTools.DynamicSelectionModeSelectMapTool.DynamicSelectionEnabled
- EditableLabels
- ExportMap: MapExport class that allows exporting a map with a wide-range of options.
- ExportSelection: MapExport.ExportSelection
- FeatureEditMode enums:
  - miEditModeFeature -> MapTools.NodeMode = false
  - miEditModeNode -> MapTools.NodeMode = true
  - miMoveDuplicateNodes -> MapTools.MoveDuplicateNodes
  - miDeleteDuplicateNodes -> MapTools.DeleteDuplicateNodes
  - miEditModeAddNode -> MapTools.AddNodeMode
- FeatureFactory: FeatureProcessor
- GeoDictionary: Geodictionary
- Geoset: Obsolete.
## Appendix A: Migrating to MapXtreme

### Map (continued)

<table>
<thead>
<tr>
<th>MapX 5.0</th>
<th>MapXtreme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geosets: Obsolete.</td>
<td></td>
</tr>
<tr>
<td>GeosetWidth</td>
<td></td>
</tr>
<tr>
<td>hWnd</td>
<td></td>
</tr>
<tr>
<td>InfotipPopupDelay</td>
<td></td>
</tr>
<tr>
<td>InfotipSupport: Not addressed.</td>
<td></td>
</tr>
<tr>
<td>IsPointVisible</td>
<td></td>
</tr>
<tr>
<td>Layers: Layers</td>
<td></td>
</tr>
<tr>
<td>MapPaperHeight/Width: PaperSize (through MapExport.ExportSize)</td>
<td></td>
</tr>
<tr>
<td>MapScreenHeight/Width: MapControl.Size, Map.Size</td>
<td></td>
</tr>
<tr>
<td>MapUnit: CoordSys.Units of the display coordsys</td>
<td></td>
</tr>
<tr>
<td>MatchNumericFields: GeoDictionary functionality.</td>
<td></td>
</tr>
<tr>
<td>MatchThreshold: GeoDictionary functionality.</td>
<td></td>
</tr>
<tr>
<td>MaxSearchTime: GeoDictionary functionality.</td>
<td></td>
</tr>
<tr>
<td>MilitaryGridReferenceToPoint: CoordSys.MilitaryGridToPoint()</td>
<td></td>
</tr>
<tr>
<td>MilitaryGridReferenceFromPoint: CoordSys.PointToMilitaryGrid()</td>
<td></td>
</tr>
<tr>
<td>Mouseicon: MapControl.Cursor</td>
<td></td>
</tr>
<tr>
<td>MousePointer: MapControl.Cursor</td>
<td></td>
</tr>
<tr>
<td>MouseWheelSupport:</td>
<td></td>
</tr>
<tr>
<td>MapControl.MouseWheelSupport.MouseWheelBehavior</td>
<td></td>
</tr>
<tr>
<td>NumericCoordSys: Obsolete. Geometries have their own CoordSys now. Everything that has a coordinate has a coordsys.</td>
<td></td>
</tr>
<tr>
<td>SetSize</td>
<td></td>
</tr>
<tr>
<td>SnapToNodeSupport</td>
<td></td>
</tr>
<tr>
<td>SnapTolerance: MapTools.SnapTolerance</td>
<td></td>
</tr>
<tr>
<td>Title: Adornments</td>
<td></td>
</tr>
<tr>
<td>TitleText</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
</tr>
<tr>
<td>WaitCursorEnabled</td>
<td></td>
</tr>
<tr>
<td>Zoom</td>
<td></td>
</tr>
<tr>
<td>ZoomTo</td>
<td></td>
</tr>
</tbody>
</table>
### Comparing MapXtreme’s Object Model to MapX

<table>
<thead>
<tr>
<th>MapX 5.0</th>
<th>MapXtreme</th>
</tr>
</thead>
<tbody>
<tr>
<td>MultivarCategory</td>
<td>MultiVariableThemeCategory</td>
</tr>
<tr>
<td>MultivarCategories</td>
<td>MultiVariableThemeCategories</td>
</tr>
<tr>
<td>NotesQueryInfo</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>NotesViewInfo</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>OCIQueryInfo</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>ODBCQueryInfo</td>
<td>Obsolete.</td>
</tr>
<tr>
<td>Parts</td>
<td>Equivalent</td>
</tr>
<tr>
<td>Point</td>
<td>DPoint</td>
</tr>
<tr>
<td>Points</td>
<td>Obsolete</td>
</tr>
<tr>
<td>RangeCategory</td>
<td>RangedThemeBin</td>
</tr>
<tr>
<td>RangeCategories</td>
<td>ModifierThemeBins</td>
</tr>
<tr>
<td>Rectangle</td>
<td>DRect</td>
</tr>
<tr>
<td>ResolveObject</td>
<td>MatchResolver</td>
</tr>
<tr>
<td>ResolveObjects</td>
<td>MatchResolver Collection.</td>
</tr>
<tr>
<td>RowValue</td>
<td>MICommand</td>
</tr>
<tr>
<td>RowValues</td>
<td>MICommand</td>
</tr>
<tr>
<td>Selection</td>
<td>Selection</td>
</tr>
<tr>
<td>SourceRow</td>
<td>SourceRow</td>
</tr>
<tr>
<td>SourceRows</td>
<td>SourceRows collection</td>
</tr>
<tr>
<td>State</td>
<td></td>
</tr>
<tr>
<td>Style</td>
<td></td>
</tr>
<tr>
<td>Theme</td>
<td>ObjectTheme, FeatureStyleModifier</td>
</tr>
<tr>
<td>Themes</td>
<td>Obsolete. Themes are now contained by either an ObjectThemeLayer (for ObjectThemes) or in the Layer's Modifiers collection (for FeatureStyleModifier themes.)</td>
</tr>
<tr>
<td>ThemeProperties</td>
<td>Obsolete. Properties are within the theme classes themselves.</td>
</tr>
<tr>
<td>Title</td>
<td>Adornments</td>
</tr>
</tbody>
</table>
Variable | MIParameter. For binding in expressions, the MICommand.Parameters property allows you to define variables used within the command.
---|---
Variable | MIParameterCollection.
How to Create and Deploy a MapXtreme Application

This appendix is designed as a tutorial to demonstrate the simplicity of creating, packaging, and deploying a MapXtreme application. At the end of this tutorial you will have gone through all the steps to successfully develop, package, and deploy a well-implemented mapping application.

This tutorial assumes that you have already successfully installed Visual Studio and are familiar with the use of that product. If you are not familiar with Visual Studio, there are many resources available on the Microsoft MSDN web site. Look at http://msdn.microsoft.com/developerscenter/. You should also have installed MapXtreme. The code fragments contained within this document were implemented using Visual Basic. For many of the sample applications, we also provide C# .NET samples.

In this appendix:

- Customizing MapXtreme Samples .................................. 436
- Building a Desktop Application .................................... 436
- Building a Web Application ....................................... 447
Customizing MapXtreme Samples

This appendix contains two tutorials:

• Building a Desktop Application
• Building a Web Application

These tutorials guide you through customizing MapXtreme desktop applications and Web sites and preparing them for deployment.

Building a Desktop Application

We are going to base our desktop application on one of the thematic sample applications provided with MapXtreme. The desktop sample applications are already working mapping applications. With some simple modifications, you can customize a sample application for your own needs. You can use the modifications outlined in this tutorial as a model for customizing any desktop MapXtreme project. All the desktop sample applications provided with MapXtreme were created from the map application template provided in the product. See Creating Applications in Visual Studio on page 42 for more information.

Running a Sample Application

Let's start by running the ThemeLegend application to see what it looks like.

1. In Visual Studio, choose File>Open>Project/Solution, and open the project for the sample application called ThemeLegend. The default path to this project is:

   C:\Program Files\MapInfo\MapXtreme\6.x.x\Samples\VisualStudio<2005 or 2008>\Desktop\Features\ThemeLegend\vb\ThemeLegendVB.vbproj

   (where x is the release number of the version of MapXtreme you are using).

2. Choose Debug>Start Debugging, or press F5 to run the application.
The application displays a standard map of the United States with a theme based on the land area of each state. The larger states are shaded in blue and the smaller states are shaded in white. The legend shows the color assigned to each range of land area values.

The tools provided with the application enable you to perform some basic functions. You can open tables, zoom in or out on a particular region, select regions on the map, move the map, and access Layer Control to apply specific settings to each layer. You might want to practice using the tools to get a feel for how the application works and what your users will experience.

3. When you are finished, close the active Form1 window to stop the application and return to Design mode.

Modifying Your Application

Next, we’ll make some modifications to the ThemeLegend code to show how you can customize a sample application. We will change the following elements of the application:

- data source
- number of ranges (bins)
- range type
- range colors
- legend location
- toolbar
1. In Visual Studio, move your cursor to the Solution Explorer and right-click on MapForm1.vb. Choose View Code to display the code page.

   If you have MapForm1.vb displayed in Design mode, you can right-click anywhere on the form, and choose View Code.

**Note** All sample applications discussed in this tutorial were taken from the MapXtreme distribution CD. The line numbers referenced may not correspond exactly with the code in your installation.

2. A new tab displays to show the application code. Scroll down the code page to familiarize yourself with the code. Notice that the majority of the action in the application takes place in the Form1() class. This is where we will make our changes.

3. Let's change the data source on which the theme is based. To do this, we'll need to change the name of the table that the application uses and specify a different FeatureLayer. Make the following changes:
   - On line 185 change the table from usa.tab to mexico.tab. This loads the Mexico map instead of the USA map.
   - On line 193, change the FeatureLayer from usa to mexico.

4. The image below highlights the code changes with red boxes.

5. Choose Debug>Start Debugging (or press F5) to run the application and see your changes. Close the Form1 window when you are finished.

The application creates the same type of theme using Mexico data. You can pass in any data source that fits this theme by changing the name of the table and the FeatureLayer.

Next, let's make changes to the theme itself. By manipulating the parameters passed to the RangedTheme constructor we can change the way the theme is calculated and displayed.
6. Highlight the word RangedTheme in the code and press F1 (line 194). The help topic for RangedTheme displays on the screen. You can read this topic to see what each parameter does when it is passed to the constructor.

7. We’ll increase the number of ranges, or bins, in the theme. On line 196 change the number of bins from 5 to 8. (A bin is a range consisting of a maximum and minimum value and is used by themes to group together like values for the purpose of shading.)

8. Next, we’ll change the distribution method of the ranges. The distribution method indicates how the ranges are calculated. On line 196, change EqualCountPerRange to EqualRangeSize. In the EqualRangeSize distribution method, each range has an equal number of values.

9. The image below highlights the code changes with a red box.

![Image of code changes]

10. Choose Debug > Start Debugging (or press F5) to build and run the modified application. Notice that the number of theme bins has increased and the distribution method has changed. Close the Form1 window when you are finished.
Next we’ll change the color of the ranges. You have two options for how you would like to modify the colors. You can set a particular color for each bin, or you can set colors for the first and last bin. The shading of the bins in between will graduate from the first to the second color. Currently, the code specifies that the color ranges from white to blue. Let’s change the colors to range from blue to red.

11. To change the color of the ranges, we’ll need to edit the fill style colors. Make sure the code view is displayed and make the following changes:

   • To change the color of the first bin, on line 206, change `WhiteFillStyle` to `BlueFillStyle`
   • To change the color of the second bin, on line 216, change `BlueFillStyle` to `RedFillStyle`

   Every bin in between will have a shade between blue and red.

12. The image below highlights the code changes with red boxes.
13. Choose Debug > Start Debugging (or press F5) to build and run the modified application. The colors of the map have changed to reflect our new settings. Close the Form1 window when you are finished.
Because we have made a number of changes, the legend is now partially blocking the view of the map. We’ll move the legend so that we can see all of the map. We could use the pan tool to move the map, but panning would not be a permanent change to the application. The new map position would be in effect only for as long as the application is running. Instead, we’ll change the position of the legend programmatically.

14. Make sure the code page is displayed and make the following change to the legend location:

- On line 233, replace “mapControl1.Size.Width - legend.Size.Width” with “0”

This changes the X coordinate of the legend location to be at the left side of the frame. We are leaving the Y coordinate as it is.

15. The image below highlights the code changes with a red box.

16. Choose Debug>Start Debugging (or press F5) to build and run your application. Now the legend is in its new location. Close the Form1 window when you are finished.
The last modification we’ll make is to add a Label tool to the toolbar.

17. In Visual Studio, bring the Design view to the front by clicking on the MapForm1.vb [Design] tab.

18. To add a toolbar, we’ll need to use the toolbar properties. Right-click on the toolbar in the MapForm1.vb [Design] tab and choose Properties. In the Behavior properties, click Buttons. You should see an ellipsis (…) button next to (Collection) setting. Click on the ellipsis (…) button.

19. The ToolBarButton Collection Editor dialog box displays on the screen. Click the Add button to add a new button to the toolbar. A new toolbar member is added to the Members list.

20. To identify the tool as a Label tool, we’ll change the name of the button. In the (Name) property, change the name of the button to mapToolBarLabel. To give the tool labeling functionality, we
Building a Desktop Application

need to specify its type. Click the drop-down list next to the **ButtonType** property and choose **Label**.

21. Click **OK** to save your changes and close the ToolBarButton Collection Editor dialog box. You should be able to see the new Label tool in your toolbar.

As you can see, you can use the samples as a base for your own customization. You can substitute your own data, modify themes, and add tools easily. Other modifications we could make to this thematic map include:

- Using a different column from the data source (table) so that the theme is based on a different value, population, for example. Don’t forget to refresh the legend to match.
- Adding additional themes and legends.
- Adding additional tools and buttons.

**Building Under Release Mode**

When you have completed the modifications, you are ready to do a release build. The release build is the build of the application that you will release to customers or internal users. To make a release build:

1. Choose Build > Configuration Manager.
2. In the **Active Solution Configuration** drop-down list, change the build type to **Release**.
3. Click **Close** in the Configuration Manager dialog box.
4. In Solution Explorer, right-click on the ThemeLegendVB project and choose Build.

When you are finished, see **Packaging Your Desktop Application on page 445**.
Packaging Your Desktop Application

This section of the tutorial shows you how to package your desktop application. A package is a collection of files and directories required for a software product. A product must be built into one or more packages so that it can be transferred to a distribution medium, such as a CD-ROM or DVD-ROM. The package for a desktop application consists of a Setup.exe file, which contains all the files needed to install and run your desktop application.

MapXtreme was designed to make this process as simple as possible. Using features in Visual Studio as well as automation in MapXtreme, the correct merge modules will be included in your package. A merge module (MSM) is a single package that contains all files, resources, registry entries, and setup logic necessary to install a component. A list of MSMs can be found in Deploying Your Application on page 49 as well as a discussion about MapXtreme’s runtime installer and other options available to you.

Create a Setup Project

We will first need to create a setup project for our solution. The Setup project enables you to create the Setup.exe file.

1. In the Solution Explorer, click on the name of the solution to highlight it and choose File > Add New Project. The Add New Project dialog box displays on the screen.
2. In the Project types list, expand Other Project Types, and click on Setup and Deployment.
3. Under Visual Studio Installed Templates, click on Setup Project.

![Add New Project dialog box]

4. Enter a name and location for the Setup project, and click OK. Visual Studio displays the File System of the new setup project.
5. Next, we need to add the data files. In the Solution Explorer, right-click on ThemeLegendVB, and choose Add>Existing Item from the menu.
6. In the Add Existing Item dialog box, navigate to the location of the data files: C:\Program Files\MapInfo\MapXtreme\6.x.x\Samples\VisualStudio2003\Desktop\Features\ThemeLegendV
7. In the Files of type box, select All files from the drop-down list.
8. Select the following files:
   • MEXICO.DAT
   • MEXICO.ID
   • MEXICO.IND
   • MEXICO.MAP
   • mexico.TAB

9. Click Add. The files are added to the Solution Explorer, under the ThemeLegendVB project.
   
   An .mws for these files will be generated in the current application directory.

   **Note** If you are deploying this application to another machine, the absolute paths for the data on the development computer and the deployment computer must be identical.

### Add a License File

Next, you'll need to supply a desktop license file with your application (MapXtremeDesktop.lic) so that your users will be able to use it. See Runtime Licenses for Deployment on page 32.

To illustrate how to add a license file, we will include a copy of the Trial license in the package.  

When you package your own applications, you will need to supply the runtime license.

1. From the Visual Studio interface, click on the File System(Setup) tab.
2. Right-click on the Application Folder icon, and choose Add>File.
3. In the Add Files dialog box, navigate to C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x), and select MapXtremeTrial.lic. Click Open.

The file is now added to your setup project and will be installed with the other application files into your application folder.

### Optional Step: Add Workspace Manager and GeoDictionary Manager

If you wish, include the MapXtreme utilities Workspace Manager and GeoDictionary Manager in your package, so your customers can use them. Follow the steps under section Add a License File on page 446. Substitute <MapXtreme installation folder>\WorkspaceManager.exe and GeoDictionaryManager.exe. for MapXtreme.lic.

### Build the Setup Project

When you build the setup project, a Setup.exe file is created that contains all the data, compiled code, and necessary MSMS for the project. This Setup project is part of your MapXtreme solution.

When you are finished building the setup project, then build the solution. When you build the entire solution, Visual Studio will compile and build the mapping application and then compile and build the Setup.exe file for installation. Your package is now complete and ready for deployment.

---

1. MapXtremeTrial.lic must come from an SDK or runtime installation of MapXtreme in order to create a package for deployment. You cannot create a setup project using the Evaluation version of MapXtreme.
Deploying Your Desktop Application

Now that you have built your desktop application and packaged all the components for deployment, the final step is to get it onto the production workstation.

To deploy your application, you must deliver the Setup.exe file you created to your end user. You can copy the Setup.exe file to a CD-ROM or DVD-ROM to provide a means of delivering the file. When the user launches the Setup.exe executable file on their computer, the installation starts and walks them through the installation. That’s all there is to it!

Building a Web Application

In this tutorial, we are going to base our application on a Web Thematics sample that is provided with MapXtreme. The MapXtreme Web samples are Visual Studio Web site projects. They contain a fully functioning Web application, include pre-loaded data to support the application, and handle state management.

Running a Sample Web Application

Let’s start by running the Thematics Web sample to see what it looks like.

1. In Visual Studio, choose File>Open>Web Site, and open the folder for the sample ThematicsVB. The default path to this folder is: C:\Program Files\MapInfo\MapXtreme\6.x.x\Samples\Web\Features\Thematics\ThematicsVB (where x is the release number of the version of MapXtreme you are using).
2. Choose Debug>Start Debugging (or press F5 to run the sample in Debug mode).

The Thematics sample Web site project contains a background map of the world, a pull-down menu listing different themes to be applied and attribute columns from a MS Access table as well as an Apply button that applies the themes. The Web page contains the standard tool buttons that are
Building a Web Application

included in MapXtreme web templates: ZoomIn, ZoomOut, Center and Pan. The application also includes a Layer Control so that you can set which layers are visible and view the current map zoom level.

For more information, see Components of a MapXtreme Web Application on page 74.

When you are finished, close the Web page, but leave the project open.

Modifying Your Application

To modify this sample application, let’s look at the code behind page, WebForm1.aspx.vb. The code behind page is the code that represents the HTML for the Web form.

Note This Thematics sample Web site is designed very differently than its desktop sample application counterpart. The most important difference is that the Web sample is tightly coupled to the data table and data column names. These items cannot simply be changed as was the case in the desktop sample.

In the code behind page, we will make some simple theme property value and range color changes. These modifications will change the themes display when the user runs The method where we will make our changes is in the CreateThemeOrModifier method, on line 244. The first change is to the Dot Density theme and how it will be displayed.

1. In Visual Studio, move your cursor to the Solution Explorer and right click on WebForm1.aspx. Choose View Code to display the code behind page, WebForm1.aspx.vb.
   If you have WebForm1.aspx displayed in Design mode, you can right-click anywhere on the form, and choose View Code.

Note All sample applications discussed in this tutorial were taken from the MapXtreme distribution CD. The line numbers referenced may not correspond exactly with the code in your installation.

Scroll down the code page to familiarize yourself with the code.
2. Make the following changes:
   • Under the ThemeAndModifierTypes.DotDensityTheme case statement, go to line 285 and change Color.Purple to Color.DarkGreen
     When the application is run, this will change the display color of the dots when selecting the Dot Density Theme.
   • On line 286, change the ValuePerDot property from 2000000 to 5000000.
     This will change the number of dots displayed based on population. The result will be a fewer number of dots.

   In the image below, the code changes are highlighted with red boxes.

3. Choose Debug>Start Debugging (or press F5) to build and run the Web site. Once the page is displayed in the browser, select DotDensityTheme theme, and then check Pop_1994.

   The color of the dots on the map is now dark green.
Building a Web Application

Now let's make one more simple change to the Web sample. This change is to the Range Theme.

4. Make the following changes:
   • Under the ThemeAndModifierTypes.RangedTheme case statement, go to line 273 and change the bin count parameter from 5 to 8.
     This will change the number of data bins for the theme.
   • On the same line, line 273, change the bin distribution method from EqualCountPerRange to NaturalBreak.
     This will change the way that the data is divided in data bins.

5. Choose Debug>Start Debugging (or press F5) to build and run the Web application with these new changes. Once the page is displayed, select RangedTheme, and then check Pop_1994.
Notice how the theme has changed with more data bins (color codings) and how the countries have changed theme color based on the new data distribution method.

As you can see, making simple display modifications to this sample Web site is relatively easy. Other possible modifications to this sample include:

- Changing the web page layout, such as moving the controls around.
- Changing the web page styles such as colors and fonts.

**State Management Considerations**

Customizing a Web application sample not only involves changing elements of the application, but also building it with state management best practices in mind. Because of the intricacies of handling applications and user state in a web environment, you must understand how MapXtreme handles them and how you can apply these concepts in your own development. State Management for Web sites is discussed in Chapter 6: Understanding State Management.

This sample, in terms of best practices, uses a pre-loaded workspace of maps and settings, manages application and user state manually, and uses pooled session objects so that the application can service multiple requests efficiently.

It is designed to detect if the user is making a first time request, in which case the application is presented in its initial state, or if the user is revisiting the application, in which case the user’s personal settings are maintained. Because this sample is sharing session objects with multiple users (known as pooling), it detects if the session is in its initial state ("clean") or contains another user’s changes ("dirty").

To maintain all users’ settings separately, this sample sets Session.State to manual. In a Web application where the MapXtreme Session state is saved automatically, the Session.State would be set to HttpSessionState. This means that the entire session is saved to the HTTP Session. Although this option automatically saves state, performance degrades because it does not determine what to save. The entire session is saved.

**Configuring for Release Mode**

When you have completed the modifications, you are ready to do a release build.
An important concept to understand in ASP.NET 2.0 is that Visual Studio 2005 knows nothing about compiling a Web site project. Previously, ASP.NET 1.1 and Visual Studio .NET would build the code-behind source code and web forms and place the compiled code in an output directory. Visual Studio 2005 delegates all compilation responsibilities to the ASP.NET platform. The Visual Studio 2005 build command only validates the Web site configuration. This compilation model makes the Configuration Manager obsolete for Web site projects in Visual Studio 2005.

To configure your Web site project to be packaged without debugging turned on, you can change a configuration setting in the Web site project’s Web.config file. In the Web.config, find the XML element "compilation" (line 27) and change the value of the attribute "debug" to "false". The result should read as follows:

```xml
<compilation defaultLanguage="vb" debug="false"/>
```

This Web.config setting lets the ASP.NET compiler know to build all source code in release mode.

To build the application, right-click on the ThematicsVB project in Solution Explorer and choose Build Web Site.

Packaging Your Web Application

Creating a package for a Web application is similar to creating a package for a desktop application.

Create a Web Setup Project

To start we are going to add a Web setup project to our solution. To create a Web setup project:

1. In the Solution Explorer, highlight the Solution and choose File > Add New Project. The Add New Project dialog box displays on the screen.
2. In the Project types list, expand Other Project Types, and click on Setup and Deployment.
3. From Setup and Deployment Projects, select Web Setup Project. Click OK. A Web setup application project will be created.
4. The next step is to indicate which parts of the solution to include. In Solution Explorer, right-click on the newly-created Web Setup project and choose Add > Project Output. The Add Project Output Group dialog box displays on the screen.

5. Click on Content Files, and then click OK.

You must include the content files because the web pages in the application are generated as HTML files rather than executables as in the desktop application.

6. If you create your own deployment for a Web site, you will need to add the MSMs manually. For more information, see Deploying a Web Site on page 53.

7. Next, you must include your data on the production server. You have the option of copying the data to the production server manually, or installing it in a separate installer.

8. In the FileSystem.tab in Visual Studio, right-click the Web application folder and open the Properties window. Change the Virtual Directory name to ThematicsWeb_6_8, which defines the virtual directory on the target machine.

Add a License File

When you package a MapXtreme Web application, you need a valid copy of MapXtremeWeb.lic to enable your users to access any maps. For more information on deployment licenses, see Runtime Licenses for Deployment on page 32. For this example, we are using the trial license\(^1\) that comes with your version of MapXtreme.

1. First, we’ll create a location for the license file. We want to duplicate the path to the MapXtreme trial license in our MapXtreme installation. This path is:
   
   C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x.

   In the File System window, right-click on File System on Target Machine and choose Add Special Folder>Common Files Folder.

2. Right click the Common Files Folder, and choose Add>Folder. Rename the folder to "MapInfo".

\(^1\) MapXtremeTrial.lic must come from an SDK or runtime installation of MapXtreme in order to create a package for deployment. You cannot create a setup project using the Evaluation version of MapXtreme.
3. Add subsequent folders by right-clicking on the last folder in the path, choosing Add>Folder, and renaming the newly created folder according to the folder structure. Your file tree should look like the tree in the figure below.

4. Right-click the 6.x.x folder and choose Add>File. Navigate to the MapXtreme trial license file (MapXtremeTrial.lic), and it to the 6.x.x folder. The MapXtreme trial license file is located here: C:\Program Files\Common Files\MapInfo\MapXtreme\6.8).

**Build Web Setup Project**

Now you are ready to build your Web setup project.

1. Choose Build>Batch Build. The Batch Build dialog box displays on the screen.
2. Find the Release configuration of the Web site project and select its corresponding Build check box.
3. Click Build.

When the build is finished, go to the Release directory of the Web Setup project. You will see either a Setup.exe file or a setup.msi file. Use this file for deployment.

**Deploying Your Web Application**

In a Web deployment, the Setup.exe file must be run on a production server by the organization that is hosting the Web site. The Setup file must be transferred from the development server to the production server and then installed on that server. In small organizations the deployment may be done by the developer. In larger organizations, an Administrator might perform the deployment. Your users then access the site by entering its URL in their Internet browser. For specific information on different installers, accessing data, proxy servers, and permissions, see Deploying Your Application on page 49.

For the purposes of this tutorial, let’s assume that you, as the developer, will do the deployment. The production server must have the .NET Framework v. 2.0 or later, and MapXtreme installed.
Once you install your application on the production server and the license file is in place, modify the Web.config file so that the pre-loaded workspace is referenced on the local machine.

When your Web site is deployed, the setup application creates the necessary virtual directory in IIS and places all the other necessary components in the right places. In our application, the license file is placed in the Program Files\Common Files\MapInfo\MapXtreme\6.x.x directory.

To see the Web site run, open a browser, and enter the URL: http://<ProductionServerMachine>/ThematicsWeb_6_8/. The URL should be the location of the Web application on the production server, using the virtual directory that you specified when you packaged the application.

Due to a problem with the Microsoft Framework, you may need to end the web process, aspnet_wp.exe on your server to successfully run your application. This process automatically restarts as soon as you end it, so you may notice no change in the Task Manager.
Customizing MapXtreme

This appendix contains examples of classes, interfaces, and other elements of MapXtreme which can be customized.

In this appendix:

- Customizable Classes .................................................. 458
- Other Customizable Aspects of MapXtreme .................. 464
Customizable Classes

The following classes have been designed to facilitate sub-classing to create custom subclasses that more closely fit your development needs.

Engine.CustomProperties

Use the CustomProperties class to add custom information to an object. The different kinds of objects that CustomProperties can add to are: FeatureCollection, FeatureStyleModifier, GmlFeatureCollection, GroupLayer, IFeatureCollection, IMapLayer, ISession, LabelModifier, LabelSource, Legend, LegendFrame, Map, MapLayer, MapTool, MultiFeatureCollection, Session.PooledSession, and Table. Add information using the Add method. Retrieve information using the Item method. CustomProperties can be of any type.

**Note** Do not add MapXtreme objects directly into a CustomProperties collection. Doing so causes errors during serialization. Instead add an Alias. For example, do not add a Map to a CustomProperties collection. Instead add Map.Alias.

Here is an example of how CustomProperties is used to add and retrieve properties:

```csharp
public static void MapInfo_Engine_CustomProperties()
{
    CustomProperties bag = new CustomProperties();
    bag.Add("One", 1);
    bag.Add("DateNow", DateTime.Now);
    int i = (int)bag["One"];
    DateTime ts = (DateTime)bag["DateNow"];
}
```

Search

To customize your search functions, there are a few classes that you can work with to accomplish almost any kind of search that you desire. The QueryFilter class allows you to create custom “where” clause to be used in SQL queries; the QueryDefinition class allows you to define a custom SQL query to be executed; and a SearchResultProcessor sets up the post processing of the results of your query.

For an example of how these classes are used, refer to the Search sample application included in the Samples directory of your MapXtreme installation (the default installation location is: C:\Program Files\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features\Search).

**QueryFilter**

The IQueryFilter interface defines the interface that all query filters must support. A QueryFilter is used to define all or part of the “where” clause of a QueryDefinition.
QueryDefinition

The QueryDefinition class defines a query to be executed by a search. The QueryDefinition is made up of a filter (IQueryFilter), Columns, and OrderBy. If no Columns are specified, then "*" is used.

I SearchResultProcessor or SearchResultProcessor

The SearchResultProcessor implements the ISearchResultProcessor interface which is used to do post processing on the results of a search to narrow down the rows selected.

FeatureStyleModifier or FeatureOverrideStyleModifier

Create your own FeatureStyleModifiers to change the appearance of a layer by deriving from FeatureStyleModifier or FeatureOverrideStyleModifier. These classes are used to specify a specific FeatureStyle modification for the objects in a specific layer.

The following example creates a modifier that sets regions in USA.TAB to red if the population has decreased from 1990 to 2000.

```csharp
using MapInfo.Mapping;
using MapInfo.Style;

internal class UsaPopulationDecreasedModifier : MapInfo.Mapping.FeatureStyleModifier
{
    public UsaPopulationDecreasedModifier() : base(null, null)
    {
        // use 2 columns for expressions
        Expressions = new string[] { I18N.Wrap("Pop_1990", I18N.WrapType.Column),
                                      I18N.Wrap("Pop_2000", I18N.WrapType.Column) };
    }

    // called during draw events
    protected override System.Boolean Modify(MapInfo.Styles.FeatureStyleStack styles,
                                               object[] values)
    {
        // compares the values from each column
        if (double.Parse(values[0].ToString()) >
            double.Parse(values[1].ToString()) )
        {
            // if population decreased, color the region red
            CompositeStyle style = styles.Current;
            if (style.AreaStyle.Interior is SimpleInterior)
            {
                // modifies the region
                return true;
            }
        }
        // does not modify the region
    }
}```
Customizable Classes

return false;
}
}

UserDrawLayer

The UserDrawLayer class is used to draw a custom layer in your map. You can populate this layer with anything you like, including a customized logo, a transparent overlay of points, etc. To use this class derive a new class from it and overload the draw method. The sample code below illustrates this:

[Serializable]
class MyUserDrawLayer : UserDrawLayer {
    // Call the base class constructor with name and alias.
    public MyUserDrawLayer(string Name, string Alias) : base(Name, Alias)
    {
    }
    // Override the abstract Draw method to draw a rectangle.
    public override void Draw(System.Drawing.Rectangle ClientRect,
    System.Drawing.Rectangle UpdateRect, Graphics graphics) {
        // Create a pen.
        Pen blackPen = new Pen(Color.Black, 3);
        // Create location and size of rectangle.
        float x = 10.0F;
        float y = 10.0F;
        float width = 200.0F;
        float height = 200.0F;
        // Draw rectangle to screen.
        graphics.DrawRectangle(blackPen, x, y, width, height);
    }
}

A VB UserDrawLayer sample application is provided with MapXtreme in the ..\MapInfo\MapXtreme\6.x.x\Samples\Desktop\Features folder.

Windows.Controls

Many of the classes in the Windows.Controls namespace can be sub-classed to implement custom behavior. Below are two examples of customizing the LayerControl using this approach.

MapInfo.Windows.Controls.PropertiesUserControl

If you want to add your own custom tabs to the LayerControl, create a class that subclasses PropertiesUserControl. Then add your class to the collection of other tab classes by calling the LayerControl's GetLayerTypeControls() and SetLayerTypeControls() methods. For an example of how this works, refer to the sample application included in your Samples\Features directory of your MapXtreme installation. Look in the MapBackgroundControl.cs file in the LayerControl sample application for the class MapBackgroundControl which subclasses PropertiesUserControl.
Appendix C: Customizing MapXtreme

MapInfo.Windows.Controls.LayerNodeHelper and all of its subclasses

LayerNodeHelper classes control the appearance and behavior of nodes in the LayerControl's layer tree. For example, these helper classes dictate which layers the user is allowed to remove. If the properties of the ILayerNodeHelper interface do not give you enough control over the appearance or behavior of layer nodes—for example, if you want to apply per-table logic, where the user is allowed to remove some map layers, but not others—you can subclass any of these helper classes, override the appropriate method, and perform your per-layer logic in your overridden method. Then put your new helper class to use by calling the LayerControl.SetLayerTypeHelper method.

By default, when the user removes a GroupLayer, a confirmation dialog appears, except in cases when the GroupLayer is completely empty—in that case, the empty GroupLayer is simply removed with no confirmation prompt (LabelLayers work similarly). To force the Layer Control to always display a confirmation prompt, even when the GroupLayer is empty, create a class that derives from GroupLayerNodeHelper and override the IsRemovalConfirmed method. The code example below illustrates how this is done.

```csharp
using System;
using MapInfo.Windows.Controls;

namespace MapInfo.Samples.LayerControl
{
    public class CustomLayerNodeHelper : GroupLayerNodeHelper
    {
        public CustomLayerNodeHelper()
        {
        }
        public override bool IsRemovalConfirmed(object obj)
        {
            return true;
        }
    }
}
```

Then, to use this new helper class in your application, call the LayerControl's SetLayerTypeHelper method, using syntax such as this:

```csharp
layerControlDlg.LayerControl.SetLayerTypeHelper(
    typeof(GroupsLayer), new CustomLayerNodeHelper()) ;
```

The subclasses of LayerNodeHelper are:

- MapNodeHelper
- MapLayerNodeHelper
- GroupLayerNodeHelper
- LabelLayerNodeHelper
- LabelSourceNodeHelper
- LabelModifierNodeHelper
- FeatureStyleModifierNodeHelper
- RangedThemeNodeHelper
- DotDensityThemeNodeHelper
- IndividualValueThemeNodeHelper
Customizable Classes

- PieThemeNodeHelper
- BarThemeNodeHelper
- GradSymbolThemeNodeHelper

Tools

Tools can be customized in two ways: by subclassing an existing tool or by adding code to a custom base tool. See Desktop Applications, Controls, Dialogs, and Tools in Chapter 8 on page 123 for more information on desktop tools. See Web Applications, Controls, and Tools in Chapter 5 on page 73 for information on web tools.

Styles

The following classes can be used to add customization to your collections of styles to apply to objects.

**BitmapPointStyleRepository**

The BitmapPointStyleRepository class is used to iterate through all current bitmap point styles and allows you to add new bitmap images from a specified directory. Using the Refresh() method places any new bitmap images into the repository that holds all images. The following code sample demonstrates the reloading of the collection of images used for BitmapPointStyles from a directory named "C:\MyData\MyBitmapImages."

```csharp
using MapInfo.Styles;

StyleRepository styleRepository = Session.Current.StyleRepository;
BitmapPointStyleRepository bmpPointStyleRepository =
    styleRepository.BitmapPointStyleRepository;
bmpPointStyleRepository.Reload("C:\MyData\MyBitmapImages");
```

**InteriorStyleRepository**

The InteriorStyleRepository class is used to iterate through all current interior styles and allows you to add new bitmap images for new interior styles from a specified directory. To add onto the current set of interior patterns used to fill regions use the AppendBitmapPattern() method. The following code sample demonstrates adding a new BMP image to the InteriorStyleRepository.

```csharp
using MapInfo.Styles;

StyleRepository styleRepository = Session.Current.StyleRepository;
InteriorStyleRepository interiorStyleRepository =
    styleRepository.InteriorStyleRepository;
// AppendBitmapPattern returns the zero-based index of the pattern in
// the repository. The index returned may be used to
// retrieve a SimpleInterior style using the repository indexer.
int index = interiorStyleRepository.AppendBitmapPattern(
    "C:\MyImages\trees.bmp");
if (index >= 0)
{
```
SimpleInterior style =
    (SimpleInterior)interiorStyleRepository[index];

**LineStyleRepository**

The LineStyleRepository class is used to iterate through all current line styles and allows you to add new line styles (created with MILISTED.EXE, available for download from MapInfo.com). The new MAPINFOW.PEN file is written by the MILISTED.EXE application and then you can use the Reload() method to put the new file into the repository. The following sample code demonstrates reloading the collection of line style patterns used for SimpleLineStyles from a PEN file in the “C:\MyData” directory.

```csharp
using MapInfo.Styles;

StyleRepository styleRepository = Session.Current.StyleRepository;
LineStyleRepository lineStyleRepository =
    styleRepository.LineStyleRepository;
lineStyleRepository.Reload("C:\MyData\MyLineStyles.PEN");
```

**VectorPointStyleRepository**

The VectorPointStyleRepository class is used to iterate through all current vector symbols and allows you to add new vector symbols from a specified file. In order to create your own version of the symbol set, you need to use the Symbol Maker MapBasic application. Once your vector symbol set is changed you use the Reload() method to read the symbol set into the repository. The following sample code demonstrates reloading the collection of SimpleVectorPointStyles from an FNT file in the “C:\MyData” directory.

```csharp
using MapInfo.Styles;

StyleRepository styleRepository = Session.Current.StyleRepository;
VectorPointStyleRepository vecPointStyleRepository =
    styleRepository.VectorPointStyleRepository;
vecPointStyleRepository.Reload("C:\MyData\MyVectorSymbols.FNT");
```

**GmlFeatureCollection**

The GmlFeatureCollection class is used to import features from an XML file. You can add Features to a map by creating a FeatureCollection object and constructing FeatureObjects to add to it. Then you can insert the FeatureCollection into a table.

**WorkSpacePersistence and WorkSpaceLoader**

The WorkSpaceLoader can be subclassed so that the persistence file being used can hold extra information saved from the application. The developer can have the application write any information that they would like saved into the persistence XML file by adding it under the UserData node. When the WorkSpacePersistence class writes the persistence file and there is some content in this node, the content is automatically written to the file. The WorkSpaceLoader class or some derived version of the class reads in all the data including what is a child of the UserData node. The following code shows an example of this.
using MapInfo.Persistence;

class myWorkSpaceLoader : WorkSpaceLoader
{
   public myWorkSpaceLoader(string fileName): base(fileName)
   {
      // empty
   }
   override public void Load()
   {
      // called parent Load() method.
      base.Load();

      XmlNode userData = this.UserDataXmlNode;
      foreach (XmlNode childNode in userData.ChildNodes)
      {
         string text = childNode.Name;
         text = childNode.InnerText;
         // user can do their own load stuff here
      }
   }
}

Other Customizable Aspects of MapXtreme

ADO.NET

If you are accessing data stored in a DataTable that is not accessible through an ADO.NET data provider, you can use the TableInfoAdoNet class to access TableTypes of non-mappable data for which we have not provided a data source. See the TableInfoAdoNet class in the online help for more detailed information about using this class.

Location of Application Data Files

Any MapXtreme application uses data stored in the following files.

<table>
<thead>
<tr>
<th>File Type</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbreviation file</td>
<td>MAPINFOW.ABB</td>
</tr>
<tr>
<td>Pen file</td>
<td>MAPINFOW.PEN</td>
</tr>
<tr>
<td>Projection file</td>
<td>MapInfoCoordinateSystemSet.xml</td>
</tr>
<tr>
<td>Vector symbol file</td>
<td>MapInfow.fnt</td>
</tr>
<tr>
<td>Custom symbol directory</td>
<td>CustSymb</td>
</tr>
</tbody>
</table>
Appendix C: Customizing MapXtreme

By default, MapXtreme applications look in the following directories for data files:

- **C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x**—This is the directory the MapXtreme installer places these files.
- The directory where your application is located. For a Windows application, this is the directory where the .exe file is located. For web applications, this is the directory where the Web.config file is located.
- **MapInfo.CoreEngine assembly folder**—This directory is the last place any application looks.

The list of directories corresponding to the above locations is obtained via the ISessions.AppDataPaths property which returns the list as an array of strings. While you cannot add to this list programmatically, you can add to it by defining a custom section in the application's configuration file. This is illustrated in the sample code below. Note that the order in which additional paths are defined determines the order in which they are searched, and all custom locations are searched before the default locations listed above.

The following example shows the `<SpecialPath>` tag:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="MapInfo.CoreEngine"
      type="MapInfo.Engine.ConfigSectionHandler,
        MapInfo.CoreEngine, Version=4.0.0.350, Culture=neutral,
        PublicKeyToken=93e298a0f6b95eb1" />
  </configSections>
  <MapInfo.CoreEngine>
    <ApplicationDataPaths>
      <SpecialPath>
        <LocalApplicationData>YourCorporation\Your Application</LocalApplicationData>
      </SpecialPath>
    </ApplicationDataPaths>
  </MapInfo.CoreEngine>
</configuration>
```

In this example MapXtreme will search for custom symbols in a folder like C:\Documents and Settings\username\Application Data\Your Corporation\Your Application\CustSymb. Note that the `<SpecialPath>` element should not include the CustSymb folder name. If it does, the path would be interpreted as ...\Your Application\CustMapXtreme 2004\CustSymb\CustSymb. The duplicated "CustSymb" would be incorrect.

Use the `<SpecialPath>` syntax if the application data is stored in a location relative to a .NET Framework special system folder. For example, if your application data is stored in a directory named MyAppData located under the "My Documents" directory, then the entry in the configuration file could be:

<table>
<thead>
<tr>
<th>File Type</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nadcon files</td>
<td>*.las, *.los</td>
</tr>
<tr>
<td>jgd2000 files</td>
<td>jgd2000.*</td>
</tr>
</tbody>
</table>
<Personal>MyAppData</Personal>

where “Personal” is the value of the .NET Framework enumeration Environment.SpecialFolder that represents the “My Documents” directory.

Each element in the configuration element above is defined as follows:

<configSections>
This is where handlers for custom sections are declared. In this case the ConfigSectionHandler class defined in assembly MapInfo.CoreEngine is responsible for parsing the MapInfo.CoreEngine section of the configuration file.

<MapInfo.CoreEngine>
This section contains settings for the MapInfo.CoreEngine assembly. Specifically, application data search paths.

<ApplicationDataPaths>
This section defines additional paths used by MapXtreme when searching for application data files.

<Path>
An element to use if the full path to the application data is known.

<SpecialPath>
Use this syntax if the application data is stored in a location relative to a well-known system folder. For example, if your application data is stored in a directory named MyAppData, located under the “My Documents” directory, then the entry in the configuration file looks like

<Personal>MyAppData</Personal>, where “Personal” is the value of the Environment.SpecialFolder enumeration that represents the “My Documents” directory. Any of the enumeration values in the Environment.SpecialFolder can be used to define an application data path in the configuration file.

Find Abbreviation File

The Abbreviation file can be customized to match your data to make your find operations more efficient. See Fine Tuning the Find Process in Chapter 14 on page 254 for more information.
Understanding the MapInfo Workspace

MapXtreme provides a workspace format that is portable, interoperable, and uses the MapInfo codespace definition. This appendix covers its definition, capabilities and use. For more about the MapInfo codespace definition, see Appendix G: Defining the MapInfo Codespace.

In this appendix:

- What is the MapInfo Workspace? ............................................ 468
- Structure of a Workspace .................................................. 469
What is the MapInfo Workspace?

Using MapXtreme, you can persist the maps you create as XML-based workspaces (with an .MWS extension) that are portable and interoperable. You will then be able to share these maps with anyone else using MapInfo products regardless of their working environment.

If you are familiar with MapXtreme Java, MapX, or MapInfo Professional, you know that currently users persist maps using different file formats:

- MapXtreme Java maps are saved as Map Definition Files (.MDF files)
- MapX maps are saved as geosets (.GST files)
- MapInfo Professional maps are saved as workspaces (.WOR files)

These files have been the way for users of individual MapInfo products to share maps. MapInfo Professional workspaces can also contain settings for browser and layout windows, graphs, legends, and sometimes even printer settings.

Users requested a single persistence file format that could be shared by all MapInfo products. The MapInfo XML-based workspace used in MapXtreme will also be used by future releases of all MapInfo products. XML gives us the portability and interoperability we need to share workspaces across locales and networks.

In MapXtreme, you can create named connections to define preferred paths for sharing new workspaces. Named connections allow you to specify alternate drive and path information or specific database connections to data on shared equipment. Users working on different computers can set up these named connection paths based on their own environment to use workspaces created by others. The path may be absolute or relative, but the file names are never included. Named connections can establish paths to a URI, a WMS, a local connection, and an RDBMS (using ODBC, OCI, or JDBC). Later, when a user gets a workspace that contains a named connection path, the correct path can be defined to locate the data referenced in the workspace.

MXP_MapDataSource_1_0.xsd is a fully documented XML schema file that specifies the format of the XML file used to define named connections. The XML schemas (.XSDs) that define the workspace and named connections are provided with MapXtreme. There are comments in the schema files to assist you in using and understanding the workspace structure. You can view these files using any text viewing tool, such as Notepad, however, we recommend using a more robust XML viewing and editing tool.

Our XML documents use textual identifiers and, where possible, we have used identifiers defined by XML standards organizations. However, XML, being eXtensible, allows for identifiers to be added by any document author to clarify the meaning of the data used in the document. To ensure that these identifiers are clearly labeled as defined by PB MapInfo Corporation, we specify them in what we call the MapInfo Codespace. To review the MapInfo Codespace identifiers, see Appendix G: Defining the MapInfo Codespace.

For more information about creating workspaces and about using the Workspace Manager, see Chapter 23: Workspace Manager.

Note You cannot use workspaces created in MapInfo Professional with MapXtreme. MapXtreme can read MapX geosets.
Structure of a Workspace

To give you a sense of the structure of a workspace, let's look at a workspace and the XML code behind it. We will be looking at the World.MWS workspace, which is in the Sample directory of your MapXtreme installation. Here, we describe four of the five types of data in the workspace.

1. The Header contains the basic file information including the version type and the creation date.
2. The Connection Section defines the location of the individual files that are contained in the workspace. This is where the named connection information is stored.
3. The DataSourceDefinition Section lists the TAB file sources of the data in the workspace.
4. The MapDefinition Section contains definitions of layer, theme, and label features associated with the workspace, such as the label details, the zoom range, the colors used, etc.
5. The UserData Section allows applications to persist their own data. User data is a “wild card” element meaning you can enter any content you want because the schema does not validate it.

Header Section

The file begins with the required XML file header information that describes the XML version the file conforms to, the encoding description, and a field that indicates whether the file is a standalone. The top level element in the file is the Workspace Element, which contains attributes for the file version, date, and XML namespace definitions.

*Note* The supported encoding for workspace persistence is UTF-8 for MapXtreme.

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<WorkSpace version="MXP_WorkSpace_1_0" date="2004-02-03T14:00:00"
xmlns:gml="http://www.opengis.net/gml"
xmlns="http://www.mapinfo.com/mxp">
```

Connection Section

The ConnectionSet section defines the database connections, file connections, and WMS connections that are necessary to use the workspace. Currently supported database connections include MS Windows ODBC connections, Oracle C interface (OCI) connections, and JDBC driver connections. File connections identify the path to the file. What follows are some examples of these connections. These connections do not appear in your sample workspaces, but rather serve as an example of database and file connections.

```xml
<ConnectionSet>
  <FileConnection dbType="file">
    <ConnectionName>mts_data_japanese</ConnectionName>
    <FilePath>\servername\DIR\AP\DATA\TestScripts\Japanese</FilePath>
  </FileConnection>
  <DBConnection dbType="xy">
    <ConnectionName>Project_xy_ids_informix_9.3</ConnectionName>
    <ODBCConnectionString>Driver={INFORMIX 3.81 32 BIT};UID=privateuser;PWD=privateuser;DATABASE=xy;HOST=HOST;SRVR=Project_ds;SERV=1800;PRO=onsoctcp;CLOC=en_US.CP1252;DLOC=en_US.CP1252;VMB=0;CURB=0;</ODBCConnectionString>
  </DBConnection>
</ConnectionSet>
```
Structure of a Workspace

DataSource Definition Section

The DataSource Definition section defines the data files (for example, TAB files) and database tables that are retrieved at each connection location. Optional meta data may also be included to describe the data that is retrieved.

<DataSourceDefinitionSet>
  <DBDataSourceDefinition id="id1" readOnly="false">
    <DataSourceName>alias_Line_ontario_oracle_9i_release_2</DataSourceName>
    <ConnectionMember>
      <ConnectionName>ontario_oracle_9i_release_1</ConnectionName>
      <DBQuery>
        <Query>select * from us_hiway_extra</Query>
      </DBQuery>
    </ConnectionMember>
    <DBDataSourceMetadata>
      
    </DBDataSourceMetadata>
  </DBDataSourceDefinition>
</DataSourceDefinitionSet>
Map Definition Section

The Map Definition section defines one or more maps and their constituent layers. Each map has display conditions that include the size of the image, the zoom and center settings, the coordinate system of the rendered map.

```xml
<MapDefinitionSet>
  <MapDefinition id="id7" name="tab_map__632140027761992151"
      alias="alias_tab_map__632140027761992151">
    <DisplayConditions>
      <MapSize uom="mapinfo:imagesize pixel">
        <ImageWidth>600</ImageWidth>
        <ImageHeight>400</ImageHeight>
      </MapSize>
      <ZoomAndCenter>
        <MapZoom uom="mapinfo:length mi">3000</MapZoom>
        <gml:Point srsName="mapinfo:coordsys 1,62">
          <gml:coordinates>-79.771366,38.003251000000006</gml:coordinates>
        </gml:Point>
      </ZoomAndCenter>
      <DisplayCoordSys>
        <SRSName>mapinfo:coordsys 1,62</SRSName>
      </DisplayCoordSys>
      <ResizeMethod>preserveZoom</ResizeMethod>
      <RasterConditions dither="halfTone" trueColor="true"
        optimizeRasterLayers="screen"
        optimizeVectorLayers="screen" />
    </DisplayConditions>
    <LayerList>
      <FeatureLayer id="id8" name="Tab_Point_632140027762460886"
        alias="alias_Tab_Point_632140027762460886"
        volatile="unknown">
        <DataSourceRef ref="id4" />
      </FeatureLayer>
      <FeatureLayer id="id9" name="Tab_Line_632140027764492071"
        alias="alias_Tab_Line_632140027764492071"
        volatile="unknown">
        <DataSourceRef ref="id2" />
      </FeatureLayer>
      <FeatureLayer id="id10" name="Tab_Region_632140027764960806"
        alias="alias_Tab_Region_632140027764960806"
        volatile="unknown">
        <DataSourceRef ref="id6" />
      </FeatureLayer>
    </LayerList>
  </MapDefinition>
  ...
</MapDefinitionSet>
```
The layer list displays with the label properties that are set for this map. These properties include the layer the labels are on, the file source for the labels, the visibility rules for the labels including the range at which the labels are visible, if applicable, the font and size properties of the labels, the label character limits, callouts, if applicable, label position settings, alignment, justification, and orientation of the label with regard to the point it is labeling.
Creating a Workspace Programmatically

In MapX, you used the Map.SaveMapAsGeoset method to create a geoset from the existing map. You give users the ability to save their workspaces programmatically as well and view these workspaces in the Workspace Manager later on using the code fragment shown below.

**Note** Users need to have a map open with one or more layers to save it as a workspace.

```csharp
// Reads a MapX geoset, writes a MapXtreme workspace.
using MapInfo.Persistence;
using MapInfo.Mapping;

MapLoader MapLoader = MapLoader.CreateFromFile("my.gst");
MapExport MapExport = new MapExport();
MapExport.Map.Load(MapLoader);
MapInfo.Persistence wsp = new WorkSpacePersistence();
wsp.Save("c:\temp\newwork.mws");
```
This appendix will guide you in printing the best possible map images from your MapXtreme development project. We begin by giving you an overview of printing functionality and some helpful tips and tricks, then we help you troubleshoot issues you may be experiencing printing with your MapXtreme application.

In this appendix:

- Overview ................................................................. 476
- Understanding the Print Options in MapXtreme ................ 476
- Implementing Printing in Your Application ..................... 480
- General Printing Tips and Tricks ................................. 482
- Resolutions to Known Printing Issues ......................... 485
Overview

Printing from MapXtreme-developed applications can usually be straightforward. However as the variety of printing devices continue to expand, device-specific problems do occur. We provide a variety of features to you, the developer, to customize your user’s printing experience. These options are designed to optimize printing depending on the map being printed and the device being used. These different settings are designed to meet the needs and nuances of a variety of printers and plotters.

MapXtreme supports:

- Printing maps – Use the MapPrintDocument class.
- Printing legends – Use the LegendPrintDocument class.
- Printing either directly to a device or by using an Enhanced Metafile (EMF).
- Printing maps in different sizes
- Printing maps that have translucent style colors and layers
  See GDI+ Translucency and Anti-Aliasing on page 478 for information.
- Printing translucent raster images
  EnableTranslucency must be enabled to print translucent raster images. See GDI+
  Translucency and Anti-Aliasing on page 478 for information.

MapXtreme does not support:

- The use of a printing options dialog box.
- Printing of layouts.
  Layouts are important for the printing of legends in a map. See Printing a Legend in Your
  Map on page 482 for an example of how to get around this limitation.
- Printing multiple page maps.
  If a map does span across multiple pages, only the first page is printed.
- Printing to a file programmatically.
  You would need to use the Microsoft.NET 2.0 Framework System.Printing.PrinterSettings
  class to implement this functionality.


Understanding the Print Options in MapXtreme

When printing a map or a legend from a MapXtreme application you can provide the following options for printing to your user. In order to give your users control over these settings, you need to build this functionality into the printing dialog boxes of your application. If you do not specifically allow them to change these options the default settings are used. You can also adjust these settings programmatically.
Appendix E: Printing From MapXtreme Applications

Printing Sizes

MapXtreme provides the options to print your map in different sizes. To alter the size in which maps are printed set the MapPrintDocument.MapPrintSize property to one of the values in the MapPrintSize enumeration. The values are as follows:

**Fit to Page**
This option is the default and the resulting map is printed with its aspect ratio maintained, but scaled to fit on the page.

MapPrintDocument.MapPrintSize=MapPrintSize.FitPage

**Fill Page**
This option prints the map so that it fills the page. This method does not maintain the aspect ratio and the resultant map may be skewed.

MapPrintDocument.MapPrintSize=MapPrintSize.FillPage

**Current Map Size**
This option prints the map in its original size. This may be fine, however if the map is larger than the page on which it is printed, the overflow is lost.

MapPrintDocument.MapPrintSize=MapPrintSize.MapSize

Note The following properties are common to both the MapInfo.Printing.MapPrintDocument and MapInfo.Printing.LegendPrintDocument classes.

Special Transparent Raster Handling

This sets your application to internally manage the transparent pixel display and printing for raster images. On screen, the transparent image is rendered using a raster operation (ROP) to handle the transparent pixels. This method may or may not work when printing since MapXtreme uses a different method to decide the transparent method for printing. You must determine if your particular print driver handles ROP correctly, and set it to true or false accordingly. Under most printing conditions, this value should be set to true. When drawing to the screen, this value is normally set to false. Use DrawingAttributes.SpecialTransparentRasterHandling to set this value. Setting this value to true (the default) allows the application to internally manage the printing.

Note This setting has no effect when printing vector layers.

Special Transparent Vector Handling

This sets your application to internally manage the transparent vector fill patterns when you print. Use DrawingAttributes.SpecialTransparentVectorHandling to enable or disable this functionality. Set this to true to allow MapXtreme to perform special handling when printing transparent fill patterns or transparent bitmap symbols. This setting is normally used for printing. Set this to false to let the printing device handle how it prints transparent fill patterns and transparent bitmap symbols. See Need for Speed When Using Fill Patterns? on page 483 for details about which fill patterns are bitmaps and which are vectors.
Display Raster in True Color When Possible

Some printers do not support 24-bit (true color) images. MapXtreme has an internal functionality that accommodates this. Use `DrawingAttributes.TrueColorRaster` to enable or disable this functionality.

Set this to `true` (default) to use the 24-bit (true color) to print raster and grid images. This is possible when the image is 24-bit and the printer supports more than 256 colors (8 bit). If your printer does not support 24-bit images, set this attribute as `false` to render the raster image using only 256 colors with dither specified in `DitherMethod`. When printing a 24-bit image with this value set as `false`, the resulting printed image will be quite deteriorated. When this attribute is set as `false`, you will need to set the dither method as well (see below).

GDI+ Translucency and Anti-Aliasing

GDI+ rendering in MapXtreme allows you to create translucent lines, labels, and layers, as well apply anti-aliasing that will smooth the jagged edges of lines, curves, and the edges of filled areas when representing a high-definition rendition at a lower resolution.

When printing a map or a legend using `MapPrintDocument` or `LegendPrintDocument`, the `DrawingAttributes.EnableTranslucency` and `DrawingAttributes.SmoothingMode` properties are automatically set to the values from `Map.DrawingAttributes` or `Legend.DrawingAttributes` respectively. After initialization, the user can override the properties by setting them with their own value. This allows these two display options to carry over from the map or legend without having to explicitly set them.

**EnableTranslucency**: Obtains or sets whether to honor translucent values in style colors and/or layers when drawing the map onto the screen, printer or file export. When this property is false, translucency values in styles (e.g., `Color.A`) and layers (e.g., `MapLayer.Alpha`) are ignored. A value of 255 is used instead. This property has no effect on raster translucency (e.g., `RasterStyle.Alpha`); however, `EnableTranslucency` must be enabled to print translucent raster images. This is not required for screen display or exporting. If you set this property to false, the `DrawingAttributes.SmoothingMode` property is automatically set to `SmoothingMode.None`. In other words, if translucency is not enabled, anti-aliasing cannot be used.

**SmoothingMode**: Obtains or sets the rendering quality of the map. This property specifies a member of the `MapInfo.Mapping.SmoothingMode` enumeration. The default value is `SmoothingMode.None`. The smoothing mode specifies whether lines, curves, and the edges of filled areas use smoothing (also called anti-aliasing). If you set this property to `SmoothingMode.AntiAlias`, the `MapInfo.Mapping.DrawingAttributes.EnableTranslucency` property is automatically set to true. In other words, translucency is always honored when anti-aliasing is used.

If you set `EnableTranslucency` to false, `SmoothingMode` will automatically be set to `None` if it is not already. If you set `SmoothingMode` to `AntiAlias`, `EnableTranslucency` will automatically be set to true if not already. While `EnableTranslucency` is true, you can switch `SmoothingMode` between `None` and `AntiAlias` without losing translucency.
MapStyleControl Class

The MapStyleControl (MapInfo.Windows.Controls namespace) can be displayed on a tab in the desktop LayerControl to allow the user to set map style and rendering options, such as anti-aliasing and translucency.

Dither Method

Dithering is a technique that blends pixels electronically to maintain the look of an image when decreasing the color depth. Select a dither method when you are converting a 24-bit image to 256 colors.

- **Halftone dithering** calculates a series of half tone differences in color between high-contrast elements in your image to create a smooth transition of color. This option is selected by default for display, print, and export options.
- **Error diffusion dithering** calculates an interim color between contrasting colors and shades the surrounding pixels to blend evenly toward that interim color.

Set this value by using the DrawingAttributes.RasterDitherMethod property. You can set this to a value from the DitherMethod enumeration.

Example:

```csharp
```

Special Polygon Hole Handling

MapXtreme applications can draw complicated Polygon objects by using several separate pieces and then merging them together. When a polygon is printed with holes or islands, the pieces may not match well, and can overlap or leave gaps. There is some internal programming that accommodates this errant behavior to make sure that the polygons are displayed correctly and print properly. To take advantage of this functionality set the value of DrawingAttributes.SpecialPolygonHoleHandling property to true. When drawing to the screen, this value should be set to false.

Scale Patterns

This setting either prints non-transparent bitmap fill patterns that look like what you see on your screen or allows the printer driver to have exclusive control over rendering the fill patterns. Set this value with DrawingAttributes.ScaleBitmapPatterns. If this value is true, the bitmap fill patterns are scaled up to compensate for the difference in printer and screen resolutions. False does not scale the pattern and relies on the printer driver for scaling the fill pattern. If the printer driver can scale fill patterns, you should set this value to false. If the printer driver cannot scale fill patterns, you should set this value to true in order to prevent the printed output from displaying too small and appearing like a solid fill. See Need for Speed When Using Fill Patterns? on page 483 for details about which fill patterns are bitmaps and which are vectors.
If this property is set to true, the print preview and the resulting printed document may not look the same, as the printer driver scales the pattern, but the display does not scale the pattern. It will print correctly, even if the display doesn’t show it properly.

Note If a fill pattern has a background color it is considered non-transparent, while one without a background color is considered a transparent fill and is always scaled, regardless of the value of this property.

Print Directly to Device

This option allows you to print your image directly to your printer. Set this value with PrintMethod.Direct.

Print Using Enhanced Metafile (EMF)

Use this option to generate an enhanced metafile of your image before sending it to the printer. This option takes advantage of current printer technology to shrink the spool size and print your file quicker without sacrificing quality. Set this value with PrintMethod.Emf.

Implementing Printing in Your Application

The printing namespace in MapXtreme contains a set of classes to help you print maps. Use these classes to handle basic printing of maps to any connected printing device.

Use the MapPrinting class to access the dialogs and printer control dialogs. It is important to allocate a single MapPrinting object for your application and use this same instance for print/print-preview/page setup.

Setup a MapPrinting and assign a map:
this.mapPrinting = new MapPrinting();
this.mapPrinting.Map = this.mapControl1.Map;

Following is sample code to print a map:
this.mapPrinting.ShowDialog = true;
this.mapPrinting.Print();

Following is sample code to display a print preview of a map:
this.mapPrinting.PrintPreview();

Following is sample code to show the Page Setup dialog:
this.mapPrinting.PageSettingsDialog();

You have access to the printer settings through the print document, which is part of the main printing class. Because we add setting and options to our derived class that support our mapping needs, you must get the class off the map printing object and re-cast it to our derived class.
mapPrinting = new MapPrinting();
mapPrinting.Map = mapControl1.Map;
MapPrintDocument mapPrintDocument = mapPrinting.PrintDocument
as MapPrintDocument;
if (mapPrintDocument != null) {
    // here are some examples of how to set print options
    // Set these based on your needs
    mapPrintDocument.DrawingAttributes.SpecialTransparentRasterHandling = true;
    mapPrintDocument.PrintMethod = PrintMethod.Direct;
    // and set other properties of mapPrintDocument
}

We do not provide any UI for changing our map print options. The dialogs allow the user to change
system printer settings only. If you want to provide the user with additional options, you'll must
provide your own UI to set the properties on MapPrintDocument (such as DrawingAttributes and
PrintMethod and PrintSize).

The print document is also where you gain access to the print process and where you add callbacks
when printing events occur. For example, you may want to add a logo or other graphics to each
printed page.

mapPrinting = new MapPrinting();
mapPrinting.Map = mapControl1.Map;
mapPrinting.PrintDocument.PrintPage += new
PrintPageEventHandler(mapPrintDocument1_PrintPage);

private void mapPrintDocument1_PrintPage(object sender,
PrintPageEventArgs e)
{
    // add customization for each page (ie; title, page #, etc.)
    Graphics g = e.Graphics;
}

mapPrinting = New MapPrinting()
mapPrinting.Map = mapControl1.Map
mapPrinting.PrintDocument.PrintPage += New
PrintPageEventHandler(mapPrintDocument1_PrintPage)

Private Property sender,() As mapPrintDocument1_PrintPage(object
End Property
Private Function e}() As PrintPageEventArgs
    ' add customization for each page (ie; title, page #, etc.)
    Dim g As Graphics = e.Graphics
End Function

Using this method, you have access to the graphics object and can use any of the available graphics
routines to print extra graphics or text.
General Printing Tips and Tricks

Note MapInfo cannot, and does not, recommend one printer/plotter over another, nor verify that a particular printer works 100 percent of the time. There are too many variables that affect the output to be able to make a recommendation.

The following tips and suggestions come from our knowledgebase of specific common problems and solutions our users have found while printing.

Good first steps in troubleshooting a printing problem are to make sure you have downloaded and installed the patch for your release of MapXtreme, if any, and are using the latest printer driver for your printer/operating system. Exceptions are noted in this document.

How to Overload a Print Page Event

Our .NET printing API provides a mechanism for overloading the print page event. This allows you to add customizations to each printed page. For example, you could add a title, page number, logo, etc.

To do this, you need to implement a PrintPageEventHandler. Here's an example:

```csharp
this.mapPrinting = new MapPrinting();
this.mapPrinting.Map = this.mapControl1.Map;

private void mapPrintDocument1_PrintPage(object sender,
{
    // TODO - add your code here to alter the printed page
}
```

Printing a Legend in Your Map

As mentioned at the beginning of this section, MapXtreme does not support the printing of layouts. If you want to print a legend in your map, you need to add the legend as an adornment to your map. Here is some sample code to demonstrate how this works:

```csharp
//create cartographic legend
legend.Border = true;
MapInfo.Mapping.LayerType[] normalLyr = new MapInfo.Mapping.LayerType[1];
normalLyr[0] = MapInfo.Mapping.LayerType.Normal;
MapInfo.Mapping.IMapLayerFilter filter = MapInfo.Mapping.MapLayerFilterFactory.FilterByLayerType(normalLyr);
MapInfo.Mapping.Legends.LegendFrame frame;
{
```

frame =
    ftrLayer);
    legend.Frames.Append(frame);
}

//set legend location on the map
System.Drawing.Point pt = new System.Drawing.Point(220, 200);
legend.Location = pt;

//append legend as map adornment
mapControl1.Map.Adornments.Append(legend);

mapPrinting.print ();

Old Driver Works, New Driver Doesn’t
When in doubt, if an older driver worked and the new one does not, go back to the older driver.

HP 755 Driver Suggestion
If you are having difficulties printing using this plotter model, try the plotter driver for the HP 650C (C2859B) instead. In many cases, if you are having difficulty with a printer or plotter model and a similar model exists, you can substitute the similar model’s driver for the current one and get good results. For example, you can use the printer driver for the HP 8500 DN Color LaserJet with the HP8550 color LaserJet driver.

Recommendations for Effective Pattern Scaling
When you are printing, find out what type of printer driver you are using. Many PCL6 and some HPGL drivers handle fill pattern scaling and give you control over this feature. Turning off their scaling may be the difference between what you see in print and what you see on your monitor. We recommend that you try turning off your driver’s scaling options and try ours first, because we have enhanced our method to better meet your Map and Legend printing requirements. To turn our pattern scaling options on, enable Scale Patterns in your application (see Scale Patterns on page 479. Try disabling our scaling to see which you like better. Tests show that our scaling produces color output that more closely matches a screen’s display.

If you are printing to PostScript drivers using LanguageLevel 2 or 3, we find that some of the Microsoft drivers do not support pattern scaling. As a result, our scaling method may not help you. Microsoft recommended that you reset the language level of the PostScript driver to LanguageLevel 1 to remove this restriction. We did find some exceptions to this condition. On Windows 2000 and Windows NT, some HP Laser Jet and Color Laser Jet PostScript drivers, using our scaling option, printed correctly.

Need for Speed When Using Fill Patterns?
Note that the first six fill patterns (after the solid fill) in of the AreaStyle dialog box are Windows standard and tend to print faster. These fill patterns are vector-based. The rest of the patterns are bitmaps that ship with MapXtreme. You might want to consider this when you are selecting fill patterns.
1 Use these fill patterns for fast printing results.

**Speed Still an Issue?**
If you want to improve printer speed, and your printer has Fast, Normal, Best print quality options, we suggest you select Fast. This will also lower your output resolution.

**Disk Space an Issue?**
Make sure you have plenty of temporary disk space, particularly if you are using the Print using the Enhanced Metafile option. The system is trying to create a layered bitmap locally on disk.

**Print Globally? Spool Locally!**
Try spooling\(^1\) print jobs locally rather than at the plotter. This allows the computer to rasterize your output rather than the printer, which can be more efficient.

---

1. Spooling is the process of storing data constituting a document to be printed in memory or in a file until the printer is ready to process it.
1 Click Spool Print Documents and Start printing after last page is spooled button.

1. To set up local print job spooling, choose Start > Settings > Control Panel > Printers.
2. Right-click the printer and choose Properties in the menu to display the printer’s properties.
3. Click the Advanced tab to display the advanced property options.

   Note If you do not have administrative rights to your computer, you may not be able to use the
     spooling option. Contact your IT department if you want to make this change to get their
     support.

4. Click OK to save your changes.

Resolutions to Known Printing Issues

There are many variables that affect printing and plotting with any application, and MapXtreme
applications are no exception. MapXtreme does not provide printer drivers; it uses the existing ones
installed under the current Windows operating system. This section addresses specific printer/plotter
issues that have been uncovered by users and partners. We are providing these in an effort to
support you to support your users.

Platform Independent Issues

These issues occur regardless of operating system or hardware/software, except where specifically
noted. Look for the issue you are working with below and refer to the resolution.

Hatch Patterns Printing as Black Polygons

Printers/Plotters: HP Designjet Series Plotters
Resolutions to Known Printing Issues

**Issue**: When you print maps with hatch patterns, some regions display as solid black.

**Resolution**: According to Hewlett Packard support, the new HP printer driver (4.63) handles non-Windows standard hatch patterns properly. We recommend that you download and install the new HP 4.63 driver to resolve this problem.

**Hatch Patterns are Compressed when I Print**

**Operating System**: WindowsNT 4.0/2000

**Printers/Plotters**: HP LaserJet Series, tested on HP LaserJet 4050 Series PCL 6

**Issue**: When you use a MapInfo hatch pattern (a pattern that is not one of the first 6 patterns AreaStyle dialog box), you may notice that the pattern may become very compressed. This is because the patterns are a bitmap that get drawn at a higher printer resolution. The standard Windows patterns do not exhibit this problem. The printer resolution affects the amount of compression.

**Resolution**: Some printer drivers have a new output setting called Scale Patterns (WYSIWYG) that enables the patterns to scale correctly. In the HP printer we tested, this option is available when you select the **Printing Preferences > Print Quality > Details** options.

Users should turn off either the driver’s, or our scaling, as doing both scales fill patterns twice. Try both types of scaling to find the one you like better.

**Platform-Specific Issues**

These issues pertain to particular operating systems and/or hardware except where specifically noted. These issues are grouped by operating system.

**Print output repeats itself every 2 to 4 cm using the HP 500, 800, 5000 DesignJet plotter when printing Rasters**

We have two suggestions to working around this problem. First, try spooling the printer locally either from the driver level or the printer level.

**Note**: You must have read/write rights to your printer and printer driver to resolve this issue.

To work around this problem, there is an advanced hidden setting that you can change within the properties of the driver.

To prevent the printer from repeating itself every 2-4 cm. when printing rasters with vector overlays:

1. Locate the printer driver’s properties in the Control Panel. To get to this setting, right-click the printer icon and choose Properties.
2. Click the Advanced tab. Select the following options:
   - Select the **Avoid out of memory** option.
   - If you are using the Windows 2000 or XP operating system, select a scaling factor of 100%.
3. From this dialog box, click the **Printing Defaults** button.
4. In this dialog box, select the **About** button. The About *<Driver Name>* dialog box displays.
5. Holding down the **F8** key on your keyboard, click the **OK** button. The Special Options dialog box displays:
6. Clear the Enable SpoolSmart check box and click OK:

**Note** When you modify settings in this driver option box, make sure to note the original settings in case you need to change them back. These were designed for HP Support Engineers and not for general public use.

You can save these settings as a Quick Set to ensure that they run every time you print raster files.

1. Navigate to the HP printer driver and access the driver preferences. Use one of these methods based on your operating system:
   - **Using Windows NT 4.0** — From the Windows desktop, select **Start > Settings > Printers**. Then, right-click the appropriate plotter driver icon and choose Document Defaults.
   - **Using Windows 2000/XP (Classic View)** — From the Windows desktop, select **Start > Settings > Printers**. Then, right-click the appropriate plotter driver icon and choose Printing Preferences.

2. Set the preferences the way you want them in the hidden menu.

3. In the Quick Sets box, type a name for the selected settings (for example, "Raster Print Settings") and click **Save**. All current driver settings (paper type, hidden menu settings, etc.) are saved under the Quick Set name. The printer driver remembers these settings and they can be used for future print jobs.

To print a raster map later with the same settings:

1. Choose the Print command to display the Print dialog box.

2. Choose the **Properties** button. The Properties dialog box displays.

3. Make sure the name you entered (as in "Raster Print Settings" in our example) displays in the Quick Sets drop-down list and click **OK**. The Print dialog box redisplays.

4. Click **OK** to print.

**Note** If you do not need this special setting in Quick Sets, remember to change this setting back to the default machine setting.
Cannot Print with HP Designjet Printers (Driver 5.31 or 5.32)

When you attempt to use these drivers one of these things happens:

- If you select the **Maximum Performance** option, you get memory error messages and the printer prints only part of the image
- If you select the **Avoid out of memory** option, the computer crashes and displays both an Out of Hard Disk Space message and an Out of Virtual Memory message

The resolution of this issue is similar to the preceding issue. The Special Options dialog box is the same.

1. **Programmatically set the following printing options:**
   ```csharp
   MapPrintDocument mapPrintDocument = this.mapPrinting.PrintDocument as MapPrintDocument;
   mapPrintDocument.PrintMethod = PrintMethod.Direct;
   mapPrintDocument.DrawingAttributes.SpecialTransparentVectorHandling = true;
   mapPrintDocument.DrawingAttributes.SpecialTransparentRasterHandling = false;
   mapPrintDocument.DrawingAttributes.TrueColorRaster = true;
   ```
2. **From the Control Panel, select Printers or Printers and Faxes and find the printer you want to print to.**
3. **Right-click on the printer and select the Properties option. The Properties dialog box displays.**
4. **Click the **About** button to display the About <Printer Driver> dialog box.**
5. **Holding down the F8 key on your keyboard, click the **OK** button. The Special Options dialog box displays.**
6. **Clear the Enable SpoolSmart check box.**
This appendix contains lookup tables for supported styles, including fill patterns, line styles, vector symbols and custom bitmap symbols. For more information on style descriptions and how to use them, see Chapter 16: Stylizing Your Maps.

In this appendix:

- Fill Patterns ......................................................... 490
- Line Styles ............................................................ 504
- Vector Symbols ...................................................... 504
- Custom Symbols .................................................... 508
Fill Patterns

The following table summarizes the MapXtreme fill patterns (also referred to as interior styles within the MapXtreme programming API). Each fill pattern has an associated Index Number, which is used for programmatic access into an InteriorStyleRepository object, and a Pattern Number, which is an internal descriptive name of the fill pattern.

More specifically:

**Fill Pattern**
The graphical fill pattern itself.

**Index Number**
The 0-based index used to retrieve the interior style that represents the fill pattern from the InteriorStyleRepository.

**Pattern Number**
The numeric identifier for the fill pattern, which may be used to construct an interior style object. This is an internal number only and can *not* be used to access the InteriorStyleRepository programmatically. However, these pattern numbers are used to indicate fill patterns within MapXtreme workspace files (.mws).

Understanding the Index Numbering Schemes

As seen in the Fill Patterns table, the fill pattern indexes (listed in the Index Number column) range from 1 to 172. This 1-based indexing scheme is used throughout other MapInfo products (for example, MapInfo Professional and MapBasic). However in the MapXtreme API, fill patterns (also called interior style objects) are acquired through an InteriorStyleRepository object that uses 0-based indexing.

Specifically, an InteriorStyleRepository object allows users to iterate through all of the available interior style objects via a 0-based indexing scheme that ranges from 0 to 172. So there are actually 173 interior style objects to choose from, however, the interior style objects at indexes 0 and 1 are identical. The equivalence of the objects at elements 0 and 1 is necessary for the following reasons:

First, we must accommodate the inherent 0-based indexing functionality of the InteriorStyleRepository class itself. That is, the InteriorStyleRepository class permits only 0-based index access for retrieving elements from its IList. (IList is a standard .NET interface that the InteriorStyleRepository class implements.)

Second, the 0-based indexing scheme must remain synchronized with the 1-based indexing scheme shown in the Pattern Number column. So there are only 172 distinct fill patterns whose indexes range from 1 to 172. The programmatic fill pattern at index 0 was added to give a definition for that index's element.

**Note** You will notice that the first eight fill patterns have the same Index Number and Pattern Number. Subsequently, the Pattern Number is always three greater than the corresponding Index Number (and hence the Pattern Numbering sequence goes up to 175 while the Index Numbering stops at 172). This is not an error. There are still 172 unique fill patterns. Nothing is missing from the table.
### Fill Patterns and Associated Index and Patterns Numbers

<table>
<thead>
<tr>
<th>Fill Pattern</th>
<th>Index Number</th>
<th>Pattern Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>(None)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>N</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>Fill Pattern</td>
<td>Index Number</td>
<td>Pattern Number</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>29</td>
</tr>
</tbody>
</table>
Fill Patterns and Associated Index and Patterns Numbers *(continued)*

<table>
<thead>
<tr>
<th>Fill Pattern</th>
<th>Index Number</th>
<th>Pattern Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Pattern Image]</td>
<td>27</td>
<td>30</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>28</td>
<td>31</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>29</td>
<td>32</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>30</td>
<td>33</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>31</td>
<td>34</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>32</td>
<td>35</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>33</td>
<td>36</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>34</td>
<td>37</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>35</td>
<td>38</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>36</td>
<td>39</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>38</td>
<td>41</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>39</td>
<td>42</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>40</td>
<td>43</td>
</tr>
</tbody>
</table>
### Fill Patterns and Associated Index and Patterns Numbers (continued)

<table>
<thead>
<tr>
<th>Fill Pattern</th>
<th>Index Number</th>
<th>Pattern Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Pattern" /></td>
<td>41</td>
<td>44</td>
</tr>
<tr>
<td><img src="image2" alt="Pattern" /></td>
<td>42</td>
<td>45</td>
</tr>
<tr>
<td><img src="image3" alt="Pattern" /></td>
<td>43</td>
<td>46</td>
</tr>
<tr>
<td><img src="image4" alt="Pattern" /></td>
<td>44</td>
<td>47</td>
</tr>
<tr>
<td><img src="image5" alt="Pattern" /></td>
<td>45</td>
<td>48</td>
</tr>
<tr>
<td><img src="image6" alt="Pattern" /></td>
<td>46</td>
<td>49</td>
</tr>
<tr>
<td><img src="image7" alt="Pattern" /></td>
<td>47</td>
<td>50</td>
</tr>
<tr>
<td><img src="image8" alt="Pattern" /></td>
<td>48</td>
<td>51</td>
</tr>
<tr>
<td><img src="image9" alt="Pattern" /></td>
<td>49</td>
<td>52</td>
</tr>
<tr>
<td><img src="image10" alt="Pattern" /></td>
<td>50</td>
<td>53</td>
</tr>
<tr>
<td><img src="image11" alt="Pattern" /></td>
<td>51</td>
<td>54</td>
</tr>
<tr>
<td><img src="image12" alt="Pattern" /></td>
<td>52</td>
<td>55</td>
</tr>
<tr>
<td><img src="image13" alt="Pattern" /></td>
<td>53</td>
<td>56</td>
</tr>
</tbody>
</table>
## Fill Patterns and Associated Index and Patterns Numbers (continued)

<table>
<thead>
<tr>
<th>Fill Pattern</th>
<th>Index Number</th>
<th>Pattern Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>54</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>55</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>56</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>57</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>58</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>59</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>61</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>62</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>65</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>66</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>67</td>
<td>70</td>
</tr>
</tbody>
</table>
## Fill Patterns and Associated Index and Patterns Numbers (continued)

<table>
<thead>
<tr>
<th>Fill Pattern</th>
<th>Index Number</th>
<th>Pattern Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Pattern Image" /></td>
<td>68</td>
<td>71</td>
</tr>
<tr>
<td><img src="image2" alt="Pattern Image" /></td>
<td>69</td>
<td>72</td>
</tr>
<tr>
<td><img src="image3" alt="Pattern Image" /></td>
<td>70</td>
<td>73</td>
</tr>
<tr>
<td><img src="image4" alt="Pattern Image" /></td>
<td>71</td>
<td>74</td>
</tr>
<tr>
<td><img src="image5" alt="Pattern Image" /></td>
<td>72</td>
<td>75</td>
</tr>
<tr>
<td><img src="image6" alt="Pattern Image" /></td>
<td>73</td>
<td>76</td>
</tr>
<tr>
<td><img src="image7" alt="Pattern Image" /></td>
<td>74</td>
<td>77</td>
</tr>
<tr>
<td><img src="image8" alt="Pattern Image" /></td>
<td>75</td>
<td>78</td>
</tr>
<tr>
<td><img src="image9" alt="Pattern Image" /></td>
<td>76</td>
<td>79</td>
</tr>
<tr>
<td><img src="image10" alt="Pattern Image" /></td>
<td>77</td>
<td>80</td>
</tr>
<tr>
<td><img src="image11" alt="Pattern Image" /></td>
<td>78</td>
<td>81</td>
</tr>
<tr>
<td><img src="image12" alt="Pattern Image" /></td>
<td>79</td>
<td>82</td>
</tr>
<tr>
<td><img src="image13" alt="Pattern Image" /></td>
<td>80</td>
<td>83</td>
</tr>
<tr>
<td><img src="image14" alt="Pattern Image" /></td>
<td>81</td>
<td>84</td>
</tr>
</tbody>
</table>
## Fill Patterns and Associated Index and Patterns Numbers (continued)

<table>
<thead>
<tr>
<th>Fill Pattern</th>
<th>Index Number</th>
<th>Pattern Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>82</td>
<td>85</td>
</tr>
<tr>
<td></td>
<td>83</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td>84</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td>85</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>86</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>87</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>88</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>91</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>92</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>93</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>94</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>95</td>
<td>98</td>
</tr>
<tr>
<td>Fill Pattern</td>
<td>Index Number</td>
<td>Pattern Number</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>96</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>97</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>98</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>102</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>102</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>103</td>
<td>106</td>
</tr>
<tr>
<td></td>
<td>104</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>105</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>106</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>107</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>108</td>
<td>111</td>
</tr>
</tbody>
</table>
### Fill Patterns and Associated Index and Patterns Numbers (continued)

<table>
<thead>
<tr>
<th>Fill Pattern</th>
<th>Index Number</th>
<th>Pattern Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Pattern 1" /></td>
<td>109</td>
<td>112</td>
</tr>
<tr>
<td><img src="image2" alt="Pattern 2" /></td>
<td>110</td>
<td>113</td>
</tr>
<tr>
<td><img src="image3" alt="Pattern 3" /></td>
<td>111</td>
<td>114</td>
</tr>
<tr>
<td><img src="image4" alt="Pattern 4" /></td>
<td>112</td>
<td>115</td>
</tr>
<tr>
<td><img src="image5" alt="Pattern 5" /></td>
<td>113</td>
<td>116</td>
</tr>
<tr>
<td><img src="image6" alt="Pattern 6" /></td>
<td>114</td>
<td>117</td>
</tr>
<tr>
<td><img src="image7" alt="Pattern 7" /></td>
<td>115</td>
<td>118</td>
</tr>
<tr>
<td><img src="image8" alt="Pattern 8" /></td>
<td>116</td>
<td>119</td>
</tr>
<tr>
<td><img src="image9" alt="Pattern 9" /></td>
<td>117</td>
<td>120</td>
</tr>
<tr>
<td><img src="image10" alt="Pattern 10" /></td>
<td>118</td>
<td>121</td>
</tr>
<tr>
<td><img src="image11" alt="Pattern 11" /></td>
<td>119</td>
<td>122</td>
</tr>
<tr>
<td><img src="image12" alt="Pattern 12" /></td>
<td>120</td>
<td>123</td>
</tr>
<tr>
<td><img src="image13" alt="Pattern 13" /></td>
<td>121</td>
<td>124</td>
</tr>
<tr>
<td>Fill Pattern</td>
<td>Index Number</td>
<td>Pattern Number</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td></td>
<td>122</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>123</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>124</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>126</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>127</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>131</td>
</tr>
<tr>
<td></td>
<td>129</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>130</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>131</td>
<td>134</td>
</tr>
<tr>
<td></td>
<td>132</td>
<td>135</td>
</tr>
<tr>
<td></td>
<td>133</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>134</td>
<td>137</td>
</tr>
</tbody>
</table>
## Fill Patterns and Associated Index and Patterns Numbers (continued)

<table>
<thead>
<tr>
<th>Fill Pattern</th>
<th>Index Number</th>
<th>Pattern Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>= = =</td>
<td>135</td>
<td>138</td>
</tr>
<tr>
<td></td>
<td>136</td>
<td>139</td>
</tr>
<tr>
<td></td>
<td>137</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>138</td>
<td>141</td>
</tr>
<tr>
<td></td>
<td>139</td>
<td>142</td>
</tr>
<tr>
<td></td>
<td>140</td>
<td>143</td>
</tr>
<tr>
<td></td>
<td>141</td>
<td>144</td>
</tr>
<tr>
<td></td>
<td>142</td>
<td>145</td>
</tr>
<tr>
<td></td>
<td>143</td>
<td>146</td>
</tr>
<tr>
<td></td>
<td>144</td>
<td>147</td>
</tr>
<tr>
<td></td>
<td>145</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td>146</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td>147</td>
<td>150</td>
</tr>
<tr>
<td>Fill Pattern</td>
<td>Index Number</td>
<td>Pattern Number</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td></td>
<td>148</td>
<td>151</td>
</tr>
<tr>
<td></td>
<td>149</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>153</td>
</tr>
<tr>
<td></td>
<td>151</td>
<td>154</td>
</tr>
<tr>
<td></td>
<td>152</td>
<td>155</td>
</tr>
<tr>
<td></td>
<td>153</td>
<td>156</td>
</tr>
<tr>
<td></td>
<td>154</td>
<td>157</td>
</tr>
<tr>
<td></td>
<td>155</td>
<td>158</td>
</tr>
<tr>
<td></td>
<td>156</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>157</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>158</td>
<td>161</td>
</tr>
<tr>
<td></td>
<td>159</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>160</td>
<td>163</td>
</tr>
<tr>
<td>Fill Pattern</td>
<td>Index Number</td>
<td>Pattern Number</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>161</td>
<td>164</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>162</td>
<td>165</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>163</td>
<td>166</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>164</td>
<td>167</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>165</td>
<td>168</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>166</td>
<td>169</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>167</td>
<td>170</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>168</td>
<td>171</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>169</td>
<td>172</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>170</td>
<td>173</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>171</td>
<td>174</td>
</tr>
<tr>
<td>![Pattern Image]</td>
<td>172</td>
<td>175</td>
</tr>
</tbody>
</table>
Vector Symbols

MapXtreme automatically installs 10 MapInfo-specific TrueType fonts during its installation process. These fonts offer the user glyph symbol choices that range from Weather, Real Estate, and Transportation, and others. The glyph numbers are Unicode character values, which, since they fall within the first Unicode character code block range, are also assignment-wise compatible with the ASCII character set.
MapInfo Arrows

MapInfo Cartographic
Vector Symbols

MapInfo Miscellaneous

MapInfo Oil & Gas

MapInfo Shields
Appendix F: Style Lookups

MapInfo Real Estate

Map Symbols

MapInfo Symbols

MapInfo 3.0 Compatible symbols
Custom Symbols

MapInfo Transportation

MapInfo Weather

Custom Symbols

The following symbols are located in C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x\CustSymb. The file extension for each image is .BMP.

These symbols can be accessed programmatically via the BitmapPointStyleRepository collection class in the MapInfo.Styles namespace.

You can create your own bitmap images and add them to the CustSymb directory. There are virtually no size limitations on an image that you create; however, the ability of MapXtreme to display it will depend on available memory. The image does not have to be square and can also have up to 24-bit color depth. To ensure that your image is displayed using its natural width and height, you must set the Boolean “NativeSize” property to true within the image’s respective BitmapPointStyle object.
<table>
<thead>
<tr>
<th>Appendix F: Style Lookups</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>BANK-64</th>
<th>BOOK1-32</th>
<th>BUILDINGS</th>
<th>CAMP1-32</th>
<th>CAR1-32</th>
<th>CAUT1-32</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHUR1-32</td>
<td>COMP1-32</td>
<td>FARM1-32</td>
<td>FAST1-32</td>
<td>FIRE1-32</td>
<td>FIRE-64</td>
</tr>
<tr>
<td>FOOD-64</td>
<td>GLOB1-32</td>
<td>GOLF1-32</td>
<td>HOSP1-32</td>
<td>HOUS1-32</td>
<td>HOUS2-32</td>
</tr>
<tr>
<td>HOUS3-32</td>
<td>HOUS-64</td>
<td>HYDR1-32</td>
<td>INTE1-32</td>
<td>LITE1-32</td>
<td>LITE2-32</td>
</tr>
<tr>
<td>MAIL1-32</td>
<td>MBOX1-32</td>
<td>MBOX2-32</td>
<td>MOSQ1-32</td>
<td>ONEW1-32</td>
<td>ONEW2-32</td>
</tr>
<tr>
<td>PENC1-32</td>
<td>PIN1-32</td>
<td>PIN2-32</td>
<td>PIN3-32</td>
<td>PIN4-32</td>
<td>PIN5-32</td>
</tr>
<tr>
<td>PIN6-32</td>
<td>PINB-64</td>
<td>PING-64</td>
<td>PINGY-64</td>
<td>PINR-64</td>
<td>POLI1-32</td>
</tr>
<tr>
<td>(PURPLE)</td>
<td>(BLUE)</td>
<td>(GREEN)</td>
<td>(GRAY)</td>
<td>(RED)</td>
<td></td>
</tr>
<tr>
<td>(CYAN)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Custom Symbols</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RAIL1-32</td>
<td>RAIL2-32</td>
<td>RAIL3-32</td>
<td>RAIL-64</td>
<td>RED-CAR</td>
<td>REST1-32</td>
</tr>
<tr>
<td><img src="image" alt="rail1-32" /></td>
<td><img src="image" alt="rail2-32" /></td>
<td><img src="image" alt="rail3-32" /></td>
<td><img src="image" alt="rail-64" /></td>
<td><img src="image" alt="red-car" /></td>
<td><img src="image" alt="rest1-32" /></td>
</tr>
<tr>
<td>STAT1-32</td>
<td>STOP1-32</td>
<td>SYNA1-32</td>
<td>TARG1-32</td>
<td>TAXI1-32</td>
<td>TEMP1-32</td>
</tr>
<tr>
<td><img src="image" alt="stat1-32" /></td>
<td><img src="image" alt="stop1-32" /></td>
<td><img src="image" alt="syna1-32" /></td>
<td><img src="image" alt="targ1-32" /></td>
<td><img src="image" alt="taxi1-32" /></td>
<td><img src="image" alt="temp1-32" /></td>
</tr>
<tr>
<td>TOWE1-32</td>
<td>TOWE2-32</td>
<td>TRAF1-32</td>
<td>TRUC1-32</td>
<td>TRUC2-32</td>
<td>TRUC-64</td>
</tr>
<tr>
<td><img src="image" alt="towe1-32" /></td>
<td><img src="image" alt="towe2-32" /></td>
<td><img src="image" alt="traf1-32" /></td>
<td><img src="image" alt="truc1-32" /></td>
<td><img src="image" alt="truc2-32" /></td>
<td><img src="image" alt="truc-64" /></td>
</tr>
<tr>
<td>YIEL1-32</td>
<td>YIEL2-32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="yield1-32" /></td>
<td><img src="image" alt="yield2-32" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Defining the MapInfo Codespace

The MapInfo Codespace is a list of definitions and standards that are commonly used in creating MapInfo maps and workspaces. You can refer to these definitions to assist you in using MapXtreme. You may want to compare our codespace definitions with another commonly used codespace, the European Petroleum Survey Group (EPSG), which is available on their website: http://www.epsg.org/.

In this appendix:

- Defining the MapInfo Codespace .............................. 512
In the following table, we show the current MapInfo Codespace. These definitions will allow your XML-based map documents to use our codespace and refer to common values, such as srsName. An example of a point geometry definition, for example, could look like this:

\[<\text{gml:Point srsName="mapinfo:coordsys 1,74"}> ...</\text{gml:Point}> \text{ or } <\text{gml:Point srsName="epsg:4269"}> ...</\text{gml:Point}>\]

### MapInfo Codespace Definition

<table>
<thead>
<tr>
<th>Category</th>
<th>Position 1</th>
<th>Position 2</th>
<th>Position 3</th>
<th>Position 4</th>
<th>Explanation and Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>coordsys</td>
<td>CoordSys file*</td>
<td></td>
<td></td>
<td></td>
<td>A coordinate reference system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:coordsys 1,74</td>
</tr>
<tr>
<td>pen</td>
<td>[1-255]</td>
<td></td>
<td></td>
<td></td>
<td>Simple Mapinfo pen pattern; numbering [1-255] based on default</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MapInfo MapInfow.pen definitions. Follows all established rules.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:pen 46</td>
</tr>
<tr>
<td>brush</td>
<td>[1-8, 12-175]</td>
<td></td>
<td></td>
<td></td>
<td>Simple Mapinfo brush pattern number [1-8, 12-175]. (Note: 9-11 are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>reserved.) Based on default MapInfo bitmap definitions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:brush 17</td>
</tr>
<tr>
<td>length</td>
<td>in</td>
<td>ft</td>
<td>yd</td>
<td>mi</td>
<td>mm</td>
</tr>
<tr>
<td></td>
<td>sft</td>
<td>nmi</td>
<td>li</td>
<td>ch</td>
<td>rd</td>
</tr>
<tr>
<td></td>
<td>pt= 1/72 in</td>
<td></td>
<td></td>
<td></td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td>pica = 12 pt</td>
<td></td>
<td></td>
<td></td>
<td>mapinfo:length m</td>
</tr>
<tr>
<td></td>
<td>twip = 1/20 pt</td>
<td></td>
<td></td>
<td></td>
<td>mapinfo:length pt</td>
</tr>
<tr>
<td></td>
<td>1 deg = (\pi/180) *</td>
<td></td>
<td></td>
<td></td>
<td>mapinfo:length deg</td>
</tr>
<tr>
<td></td>
<td>6370997 meter = 69.09329 mi</td>
<td></td>
<td></td>
<td></td>
<td>(based on great circle)</td>
</tr>
</tbody>
</table>

Based on great circle)}
### MapInfo Codespace Definition (continued)

<table>
<thead>
<tr>
<th>Category</th>
<th>Position 1</th>
<th>Position 2</th>
<th>Position 3</th>
<th>Position 4</th>
<th>Explanation and Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>imagesize</td>
<td>pixel</td>
<td></td>
<td></td>
<td></td>
<td>A non-linear unit of image dimension. Example: mapinfo:imagesize pixel</td>
</tr>
<tr>
<td>type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A data type.</td>
</tr>
<tr>
<td>boolean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type boolean</td>
</tr>
<tr>
<td>byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type byte</td>
</tr>
<tr>
<td>date</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type date</td>
</tr>
<tr>
<td>datetime</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type datetime</td>
</tr>
<tr>
<td>time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type time</td>
</tr>
<tr>
<td>decimal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>optional min \n optional max \n optional precision \n Examples: \n mapinfo:type decimal (100 -- all d&gt;100) \n mapinfo:type decimal 100 -- all d&lt;100 \n mapinfo:type decimal [200 300] -- all 200&lt;=d&lt;=300 \n mapinfo:type decimal 10.7 -- all decimals with no more than 10 digits with 7 digits reserved for right of decimal point, m&lt;=n \n mapinfo:type decimal [200 300] 10.7 -- combination of previous two examples.</td>
</tr>
<tr>
<td>double</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type double (0 150)</td>
</tr>
<tr>
<td>float</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type float 30.25]</td>
</tr>
<tr>
<td>int</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type int [0 180]</td>
</tr>
<tr>
<td>short</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type short</td>
</tr>
<tr>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Optional max length n. Example: mapinfo:type string 256</td>
</tr>
<tr>
<td>char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:type char</td>
</tr>
</tbody>
</table>
### MapInfo Codespace Definition (continued)

<table>
<thead>
<tr>
<th>Category</th>
<th>Position 1</th>
<th>Position 2</th>
<th>Position 3</th>
<th>Position 4</th>
<th>Explanation and Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>operators</td>
<td></td>
<td></td>
<td></td>
<td>A scalar attribute comparison operator or a geometry operator or a boolean operator.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>eq</td>
<td>neq</td>
<td>lt</td>
<td>gt</td>
<td>lteq</td>
</tr>
<tr>
<td></td>
<td>in</td>
<td>not_in</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>between</td>
<td>not_between</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>like</td>
<td>not_like</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>intersects</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The feature geometry fg intersects the given Polygon.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:op intersects</td>
</tr>
</tbody>
</table>
## Appendix G: Defining the MapInfo Codespace

### MapInfo Codespace Definition (continued)

<table>
<thead>
<tr>
<th>Category</th>
<th>Position 1</th>
<th>Position 2</th>
<th>Position 3</th>
<th>Position 4</th>
<th>Explanation and Example</th>
</tr>
</thead>
</table>
| operators (continued)  | mbr_intersects |            |            |            | The mbr (minimum bound rectangle) of the feature geometry (fg) intersects the mbr of the given Polygon.  
Example: mapinfo:op mbr_intersects |
| contains               |            |            |            |            | The feature geometry (fg) contains the given Point. The given Polygon contains the feature geometry fg.  
Example: mapinfo:op contains |
| contains_centroid      |            |            |            |            | The feature geometry (fg) contains the centroid of the given Polygon. The given polygon contains the centroid of the feature geometry (fg)  
Example: mapinfo:op contains_centroid |
| and                    |            |            |            |            | Boolean and.  
Example: mapinfo:op and   |
| or                     |            |            |            |            | Boolean or.  
Example: mapinfo:op or   |
<table>
<thead>
<tr>
<th>Category</th>
<th>Position 1</th>
<th>Position 2</th>
<th>Position 3</th>
<th>Position 4</th>
<th>Explanation and Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>area</td>
<td>sq</td>
<td>in</td>
<td>ft</td>
<td>yd</td>
<td>mi</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>acre</td>
<td>a</td>
<td>ha</td>
<td>perch</td>
<td>rood</td>
</tr>
<tr>
<td></td>
<td>a = are (an area of 10m x 10m)</td>
<td>ha = hectare (an area of 100m x 100m)</td>
<td>perch = 272.25 sq ft (one sq rod 16.5 ft)</td>
<td>rood = 40 perch (1/4 acre)</td>
<td>Example: mapinfo:area acre</td>
</tr>
<tr>
<td>time</td>
<td>msec</td>
<td>sec</td>
<td>min</td>
<td>hr</td>
<td>day</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>angle</td>
<td>deg</td>
<td>rad</td>
<td></td>
<td>A unit of angular measure. degree</td>
<td>radian</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:angle rad</td>
<td></td>
</tr>
<tr>
<td>temp</td>
<td>K</td>
<td>F</td>
<td>C</td>
<td></td>
<td>A unit of temperature. Kelvin</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Example: mapinfo:temp C</td>
<td></td>
</tr>
</tbody>
</table>

* Coordinate System information for MapXtreme is located in the MapInfoCoordinateSystemSet.xml, located in C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x.
Elements of a Coordinate System

In this appendix:

- Projections and Their Parameters ........................................... 518
- Projection Datums ................................................................. 522
- Datum Conversion ................................................................. 533
- Custom Datums ................................................................. 534
- Information on Coordinate Systems and Projections .......... 539
The following table indicates the parameters applicable to each projection, which are listed in the order they appear in the relevant coordinate system lines in the MapInfoCoordinateSystemSet.xml. The document is located in C:\Program Files\Common Files\MapInfo\MapXtreme\6.x.x.

<table>
<thead>
<tr>
<th>Datum</th>
<th>Units</th>
<th>Origin, Longitude</th>
<th>Origin, Latitude</th>
<th>Standard Parallel 1</th>
<th>Standard Parallel 2</th>
<th>Azimuth</th>
<th>Scale Factor</th>
<th>False Easting</th>
<th>False Northing</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albers Equal-Area Conic</td>
<td>X  X  X  X  X  X  X  X</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Azimuthal Equidistant</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cassini-Soldner</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cylindrical Equal Area</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Double Stereographic</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eckert IV</td>
<td>X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eckert VI</td>
<td>X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equidistant Conic</td>
<td>X  X  X  X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gall</td>
<td>X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hotine Oblique Mercator</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lambert Azimuthal Equal-Area</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lambert Conformal Conic</td>
<td>X  X  X  X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Longitude-Latitude</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mercator</td>
<td>X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Miller</td>
<td>X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mollweide</td>
<td>X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Zealand Map Grid</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polyconic</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regional Mercator</td>
<td>X  X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robinson</td>
<td>X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix H: Elements of a Coordinate System

Projection

The projection is the equation or equations used by a coordinate system. The following list names the projections MapInfo uses and gives the number used to identify the projection in the MapInfoCoordinateSystemSet.xml file:

<table>
<thead>
<tr>
<th>Number</th>
<th>Projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Albers Equal-Area Conic</td>
</tr>
<tr>
<td>28</td>
<td>Azimuthal Equidistant (all origin latitudes)</td>
</tr>
<tr>
<td>5</td>
<td>Azimuthal Equidistant (polar aspect only)</td>
</tr>
<tr>
<td>30</td>
<td>Cassini-Soldner</td>
</tr>
<tr>
<td>2</td>
<td>Cylindrical Equal-Area</td>
</tr>
<tr>
<td>31</td>
<td>Double Stereographic</td>
</tr>
<tr>
<td>14</td>
<td>Eckert IV</td>
</tr>
<tr>
<td>15</td>
<td>Eckert VI</td>
</tr>
<tr>
<td>6</td>
<td>Equidistant Conic, also known as Simple Conic</td>
</tr>
<tr>
<td>17</td>
<td>Gall</td>
</tr>
<tr>
<td>7</td>
<td>Hotine Oblique Mercator</td>
</tr>
<tr>
<td>4</td>
<td>Lambert Azimuthal Equal-Area (polar aspect only)</td>
</tr>
<tr>
<td>29</td>
<td>Lambert Azimuthal Equal-Area</td>
</tr>
<tr>
<td>3</td>
<td>Lambert Conformal Conic</td>
</tr>
</tbody>
</table>
### Projections and Their Parameters

For example, a Longitude/Latitude projection is listed in the MapInfoCoordinateSystemSet.xml as:

```xml
<GeographicCRS>
  <srsName>Longitude / Latitude (Porto Santo 1936)</srsName>
  <srsID>
    <code>coordsys 1,94</code>
    <codeSpace>mapinfo</codeSpace>
    <remarks>Longitude / Latitude (v 6.0 and later projections)</remarks>
  </srsID>
  <usesEllipsoidalCS/>
  <usesGeodeticDatum/>
</GeographicCRS>
```

Projection numbers in the MapInfoCoordinateSystemSet.xml may be modified by the addition of a constant value to the base number listed in the Projection table, above. Valid values and their meanings are tabulated below:

<table>
<thead>
<tr>
<th>Number</th>
<th>Projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>Lambert Conformal Conic (modified for Belgium 1972)</td>
</tr>
<tr>
<td>1</td>
<td>Longitude/Latitude</td>
</tr>
<tr>
<td>10</td>
<td>Mercator</td>
</tr>
<tr>
<td>11</td>
<td>Miller Cylindrical</td>
</tr>
<tr>
<td>13</td>
<td>Mollweide</td>
</tr>
<tr>
<td>18</td>
<td>New Zealand Map Grid</td>
</tr>
<tr>
<td>27</td>
<td>Polyconic</td>
</tr>
<tr>
<td>26</td>
<td>Regional Mercator</td>
</tr>
<tr>
<td>12</td>
<td>Robinson</td>
</tr>
<tr>
<td>16</td>
<td>Sinusoidal</td>
</tr>
<tr>
<td>20</td>
<td>Stereographic</td>
</tr>
<tr>
<td>25</td>
<td>Swiss Oblique Mercator</td>
</tr>
<tr>
<td>8</td>
<td>Transverse Mercator, (also known as Gauss-Kruger)</td>
</tr>
<tr>
<td>21</td>
<td>Transverse Mercator, (modified for Danish System 34 Jylland-Fyn)</td>
</tr>
<tr>
<td>22</td>
<td>Transverse Mercator, (modified for Danish System 34 Sjaelland)</td>
</tr>
<tr>
<td>23</td>
<td>Transverse Mercator, (modified for Danish System 34/45 Bornholm)</td>
</tr>
<tr>
<td>24</td>
<td>Transverse Mercator, (modified for Finnish KKJ)</td>
</tr>
</tbody>
</table>

For example, a Longitude/Latitude projection is listed in the MapInfoCoordinateSystemSet.xml as:
Appendix H: Elements of a Coordinate System

Example:
Assume you want to work with a simple system based on the Transverse Mercator projection and using the NAD 1983 datum. You might have a line such as the following in your MapInfoCoordinateSystemSet.xml file:

```xml
<ProjectedCRS>
  <srsName>UTM Zone 1 (NAD 83)</srsName>
  <srsID>
    <code>coordsys 8,33,7,-177,0,0.9996,500000,0</code>
    <codeSpace>mapinfo</codeSpace>
    <remarks>Universal Transverse Mercator</remarks>
  </srsID>
  <baseCRS/>
  <definedByConversion/>
  <usesCartesianCS/>
</ProjectedCRS>
```

Now let’s say that you want a system based on this, but with an affine transformation specified by the following parameters:

Units=meters; \( A=0.5; B=-0.866; C=0; D=0.866; E=0.5; \) and \( F=0. \)

The required line in the MapInfoCoordinateSystemSet.xml file is:

```xml
<ProjectedCRS>
  <srsName>UTM Zone 1 (NAD 83)- rotated 60 degrees</srsName>
  <srsID>
    <code>coordsys 8,33,7,-177,0,0.9996,500000,0, 7, 0.5, -0.866, 0, 0.866, 0.5, 0</code>
    <codeSpace>mapinfo</codeSpace>
    <remarks>Universal Transverse Mercator</remarks>
  </srsID>
  <baseCRS/>
  <definedByConversion/>
  <usesCartesianCS/>
</ProjectedCRS>
```

Alternatively, if you want to bound the system to \( (x_1, y_1, x_2, y_2)=(-500000, 0, 500000, 1000000), \) the required line is:

```xml
<ProjectedCRS>
  <srsName>UTM Zone 1 (NAD 83)- bounded</srsName>
  <srsID>
    <code>coordsys 8,33,7,-177,0,0.9996,500000,0, 7, 0.5, -0.866, 0, 0.866, 0.5, 0
    <codeSpace>mapinfo</codeSpace>
    <remarks>Universal Transverse Mercator</remarks>
  </srsID>
  <baseCRS/>
  <definedByConversion/>
  <usesCartesianCS/>
</ProjectedCRS>
```

<table>
<thead>
<tr>
<th>Constant</th>
<th>Meaning</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>System has affine transformations</td>
<td>Affine units specifier and coefficients appear after the regular parameters for the system.</td>
</tr>
<tr>
<td>2000</td>
<td>System has explicit bounds</td>
<td>Bounds appear after the regular parameters for the system.</td>
</tr>
<tr>
<td>3000</td>
<td>System with both affine and bounds</td>
<td>Affine parameters follow system’s parameters; bounds follow affine parameters.</td>
</tr>
</tbody>
</table>
To customize the system using both of these modifications, the line is:

```xml
<ProjectedCRS>
  <srsName>UTM Zone 1 (NAD 83)</srsName>
  <srsID>
    <code>coordsys 8,33,7,-177,0,0.9996,500000,0, 7, 0.5, -
     0.866, 0, 0.866, 0.5, 0, -500000, 0, 500000, 1000000</code>
  <codeSpace>mapinfo</codeSpace>
  <remarks>Universal Transverse Mercator</remarks>
  </srsID>
  <baseCRS/>
  <definedByConversion/>
  <usesCartesianCS/>
</ProjectedCRS>
```

### Projection Datums

The datum is established by tying a reference ellipsoid to a particular point on the earth. The following table lists these details for each datum:

- The number used to identify the datum in the MapInfoCoordinateSystemSet.xml file.
- The datum’s name
- The maps for which the datum is typically used
- The datum’s reference ellipsoid

<table>
<thead>
<tr>
<th>Number</th>
<th>Datum</th>
<th>Area Maps</th>
<th>Ellipsoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adindan</td>
<td>Ethiopia, Mali, Senegal, Sudan</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>2</td>
<td>Afgooye</td>
<td>Somalia</td>
<td>Krassovsky</td>
</tr>
<tr>
<td>1007</td>
<td>AGD 66, 7 parameter</td>
<td>Australia, A.C.T.</td>
<td>Australian National</td>
</tr>
<tr>
<td>1008</td>
<td>AGD 66, 7 parameter</td>
<td>Australia, Tasmania</td>
<td>Australian National</td>
</tr>
<tr>
<td>1009</td>
<td>AGD 66, 7 parameter</td>
<td>Australia, Victoria/NSW</td>
<td>Australian National</td>
</tr>
</tbody>
</table>
## Appendix H: Elements of a Coordinate System

<table>
<thead>
<tr>
<th>Number</th>
<th>Datum</th>
<th>Area Maps</th>
<th>Ellipsoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1006</td>
<td>AGD 84, 7 parameter</td>
<td>Australia</td>
<td>Australian National</td>
</tr>
<tr>
<td>3</td>
<td>Ain el Abd 1970</td>
<td>Bahrain Island</td>
<td>International</td>
</tr>
<tr>
<td>118</td>
<td>American Samoa</td>
<td>American Samoa Islands</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>4</td>
<td>Anna 1 Astro 1965</td>
<td>Cocos Islands</td>
<td>Australian National</td>
</tr>
<tr>
<td>119</td>
<td>Antigua Island Astro 1943</td>
<td>Antigua, Leeward Islands</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>5</td>
<td>Arc 1950</td>
<td>Botswana, Lesotho, Malawi, Swaziland, Zaire, Zambia, Zimbabwe</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>6</td>
<td>Arc 1960</td>
<td>Kenya, Tanzania</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>7</td>
<td>Ascension Island 1958</td>
<td>Ascension Island</td>
<td>International</td>
</tr>
<tr>
<td>9</td>
<td>Astro B4 Sorol Atoll</td>
<td>Tem Island</td>
<td>International</td>
</tr>
<tr>
<td>8</td>
<td>Astro Beacon “E”</td>
<td>Iwo Jima Island</td>
<td>International</td>
</tr>
<tr>
<td>10</td>
<td>Astro DOS 71/4</td>
<td>St. Helena Island</td>
<td>International</td>
</tr>
<tr>
<td>11</td>
<td>Astronomic Station 1952</td>
<td>Marcus Island</td>
<td>International</td>
</tr>
<tr>
<td>151</td>
<td>ATS77 (Average Terrestrial System 1977)</td>
<td>Canada</td>
<td>ATS77</td>
</tr>
<tr>
<td>12</td>
<td>Australian Geodetic 1966 (AGD 66 - 3 param)</td>
<td>Australia and Tasmania Island</td>
<td>Australian National</td>
</tr>
<tr>
<td>13</td>
<td>Australian Geodetic 1984 (AGD 84 - 3 param)</td>
<td>Australia and Tasmania Island</td>
<td>Australian National</td>
</tr>
<tr>
<td>1006</td>
<td>Australia-National-AGD84 7 param</td>
<td>Australia</td>
<td>Australian National</td>
</tr>
<tr>
<td>1007</td>
<td>Australia (A.C.T. AGD66 7-param)</td>
<td>Australia</td>
<td>Australian National</td>
</tr>
<tr>
<td>1008</td>
<td>Australia (Tasmania AGD66 7-param)</td>
<td>Tasmania</td>
<td>Australian National</td>
</tr>
<tr>
<td>1009</td>
<td>Australia (Victoria/NSW AGD66 7-param)</td>
<td>Victoria, NSW</td>
<td>Australian National</td>
</tr>
<tr>
<td>151</td>
<td>Average Terrestrial System 1977 (ATS77)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>Ayabelle Lighthouse</td>
<td>Djibouti</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>Number</td>
<td>Datum</td>
<td>Area Maps</td>
<td>Ellipsoid</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------</td>
<td>------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>110</td>
<td>Belgium</td>
<td>Belgium</td>
<td>International</td>
</tr>
<tr>
<td>14</td>
<td>Bellevue (IGN)</td>
<td>Efate and Erromango Islands</td>
<td>International</td>
</tr>
<tr>
<td>15</td>
<td>Bermuda 1957</td>
<td>Bermuda Islands</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>16</td>
<td>Bogota Observatory</td>
<td>Colombia</td>
<td>International</td>
</tr>
<tr>
<td>121</td>
<td>Bukit Rimpah</td>
<td>Bangka and Belitung Islands</td>
<td>Bessel 1841</td>
</tr>
<tr>
<td>17</td>
<td>Campo Inchauspe</td>
<td>Argentina</td>
<td>International</td>
</tr>
<tr>
<td>18</td>
<td>Canton Astro 1966</td>
<td>Phoenix Islands</td>
<td>International</td>
</tr>
<tr>
<td>19</td>
<td>Cape</td>
<td>South Africa</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>20</td>
<td>Cape Canaveral</td>
<td>Florida and Bahama Islands</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>1005</td>
<td>Cape, 7 parameter</td>
<td>South Africa</td>
<td>WGS 84</td>
</tr>
<tr>
<td>21</td>
<td>Carthage</td>
<td>Tunisia</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>22</td>
<td>Chatham 1971</td>
<td>Chatham Island (New Zealand)</td>
<td>International</td>
</tr>
<tr>
<td>23</td>
<td>Chua Astro</td>
<td>Paraguay</td>
<td>International</td>
</tr>
<tr>
<td>122</td>
<td>Co-Ordinate System 1937 of Estonia</td>
<td>Estonia</td>
<td>Bessel 1841</td>
</tr>
<tr>
<td>24</td>
<td>Corrego Alegre</td>
<td>Brazil</td>
<td>International</td>
</tr>
<tr>
<td>123</td>
<td>Dabola</td>
<td>Guinea</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>124</td>
<td>Deception Island</td>
<td>Deception Island, Antarctica</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>1000</td>
<td>Deutsches Hauptdreikantsnetz (DHDN)</td>
<td>Germany</td>
<td>Bessel</td>
</tr>
<tr>
<td>25</td>
<td>Djakarta (Batavia)</td>
<td>Sumatra Island (Indonesia)</td>
<td>Bessel 1841</td>
</tr>
<tr>
<td>26</td>
<td>DOS 1968</td>
<td>Gizo Island (New Georgia Islands)</td>
<td>International</td>
</tr>
<tr>
<td>27</td>
<td>Easter Island 1967</td>
<td>Easter Island</td>
<td>International</td>
</tr>
<tr>
<td>115</td>
<td>EUREF 89</td>
<td>Europe</td>
<td>GRS 80</td>
</tr>
<tr>
<td>Number</td>
<td>Datum</td>
<td>Area Maps</td>
<td>Ellipsoid</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>28</td>
<td>European 1950 (ED 50)</td>
<td>Austria, Belgium, Denmark, Finland, France, Germany, Gibraltar, Greece, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland</td>
<td>International</td>
</tr>
<tr>
<td>29</td>
<td>European 1979 (ED 79)</td>
<td>Austria, Finland, Netherlands, Norway, Spain, Sweden, Switzerland</td>
<td>International</td>
</tr>
<tr>
<td>108</td>
<td>European 1987 (ED 87)</td>
<td>Europe</td>
<td>International</td>
</tr>
<tr>
<td>125</td>
<td>Fort Thomas 1955</td>
<td>Nevis, St. Kitts, Leeward Islands</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>30</td>
<td>Gandajika Base</td>
<td>Republic of Maldives</td>
<td>International</td>
</tr>
<tr>
<td>116</td>
<td>GDA 94</td>
<td>Australia</td>
<td>GRS 80</td>
</tr>
<tr>
<td>33</td>
<td>Geodetic Reference System 1980 (GRS 80)</td>
<td>Worldwide</td>
<td>GRS 80</td>
</tr>
<tr>
<td>126</td>
<td>Graciosa Base SW 1948</td>
<td>Faial, Graciosa, Pico, Sao Jorge, and Terceira Islands (Azores)</td>
<td>International 1924</td>
</tr>
<tr>
<td>34</td>
<td>Guam 1963</td>
<td>Guam Island</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>35</td>
<td>GUX 1 Astro</td>
<td>Guadalcanal Island</td>
<td>International</td>
</tr>
<tr>
<td>150</td>
<td>Hartbeesthoek 94</td>
<td>South Africa</td>
<td>WGS 84</td>
</tr>
<tr>
<td>127</td>
<td>Herat North</td>
<td>Afghanistan</td>
<td>International 1924</td>
</tr>
<tr>
<td>128</td>
<td>Hermannskogel</td>
<td>Yugoslavia (Prior to 1990), Slovenia, Croatia, Bosnia and Herzegovina, Serbia</td>
<td>Bessel 1841</td>
</tr>
<tr>
<td>36</td>
<td>Hito XVIII 1963</td>
<td>South Chile (near 53×S)</td>
<td>International</td>
</tr>
<tr>
<td>37</td>
<td>Hjorsey 1955</td>
<td>Iceland</td>
<td>International</td>
</tr>
<tr>
<td>38</td>
<td>Hong Kong 1963</td>
<td>Hong Kong</td>
<td>International</td>
</tr>
<tr>
<td>1004</td>
<td>Hungarian Datum (HD 72)</td>
<td>Hungary</td>
<td>GRS 67</td>
</tr>
<tr>
<td>Number</td>
<td>Datum</td>
<td>Area Maps</td>
<td>Ellipsoid</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------</td>
<td>------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>39</td>
<td>Hu-Tzu-Shan</td>
<td>Taiwan</td>
<td>International</td>
</tr>
<tr>
<td>40</td>
<td>Indian</td>
<td>Thailand and Vietnam</td>
<td>Everest (India 1830)</td>
</tr>
<tr>
<td>41</td>
<td>Indian</td>
<td>Bangladesh, India, Nepal</td>
<td>Everest (India 1830)</td>
</tr>
<tr>
<td>129</td>
<td>Indian</td>
<td>Pakistan</td>
<td>Everest (Pakistan)</td>
</tr>
<tr>
<td>130</td>
<td>Indian 1954</td>
<td>Thailand</td>
<td>Everest (India 1830)</td>
</tr>
<tr>
<td>131</td>
<td>Indian 1960</td>
<td>Vietnam</td>
<td>Everest (India 1830)</td>
</tr>
<tr>
<td>132</td>
<td>Indian 1975</td>
<td>Thailand</td>
<td>Everest (India 1830)</td>
</tr>
<tr>
<td>133</td>
<td>Indonesian 1974</td>
<td>Indonesia</td>
<td>Indonesian 1974</td>
</tr>
<tr>
<td>42</td>
<td>Ireland 1965</td>
<td>Ireland</td>
<td>Modified Airy</td>
</tr>
<tr>
<td>134</td>
<td>ISTS 061 Astro 1968</td>
<td>South Georgia Island</td>
<td>International 1924</td>
</tr>
<tr>
<td>43</td>
<td>ISTS 073 Astro 1969</td>
<td>Diego Garcia</td>
<td>International</td>
</tr>
<tr>
<td>152</td>
<td>Japanese Geodetic Datum 2000 (JGD2000)</td>
<td>Japan</td>
<td>GRS80</td>
</tr>
<tr>
<td>44</td>
<td>Johnston Island 1961</td>
<td>Johnston Island</td>
<td>International</td>
</tr>
<tr>
<td>45</td>
<td>Kandawala</td>
<td>Sri Lanka</td>
<td>Everest (India 1830)</td>
</tr>
<tr>
<td>46</td>
<td>Kerguelen Island</td>
<td>Kerguelen Island</td>
<td>International</td>
</tr>
<tr>
<td>47</td>
<td>Kertau 1948</td>
<td>West Malaysia and Singapore</td>
<td>Everest (W. Malaysia and Singapore 1948)</td>
</tr>
<tr>
<td>135</td>
<td>Kusaie Astro 1951</td>
<td>Caroline Islands, Federated States of Micronesia</td>
<td>International 1924</td>
</tr>
<tr>
<td>48</td>
<td>L.C. 5 Astro</td>
<td>Cayman Brac Island</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>136</td>
<td>Leigon</td>
<td>Ghana</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>49</td>
<td>Liberia 1964</td>
<td>Liberia</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>113</td>
<td>Lisboa (DLx)</td>
<td>Portugal</td>
<td>International</td>
</tr>
<tr>
<td>Number</td>
<td>Datum</td>
<td>Area Maps</td>
<td>Ellipsoid</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------</td>
<td>----------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>50</td>
<td>Luzon</td>
<td>Philippines (excluding Mindanao Island)</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>51</td>
<td>Luzon</td>
<td>Mindanao Island</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>52</td>
<td>Mahe 1971</td>
<td>Mahe Island</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>53</td>
<td>Marco Astro</td>
<td>Salvage Islands</td>
<td>International</td>
</tr>
<tr>
<td>54</td>
<td>Massawa</td>
<td>Eritrea (Ethiopia)</td>
<td>Bessel 1841</td>
</tr>
<tr>
<td>114</td>
<td>Melrica 1973 (D73)</td>
<td>Portugal</td>
<td>International</td>
</tr>
<tr>
<td>55</td>
<td>Merchich</td>
<td>Morocco</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>56</td>
<td>Midway Astro 1961</td>
<td>Midway Island</td>
<td>International</td>
</tr>
<tr>
<td>57</td>
<td>Minna</td>
<td>Nigeria</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>137</td>
<td>Montserrat Island Astro 1958</td>
<td>Montserrat, Leeward Islands</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>138</td>
<td>M’Poraloko</td>
<td>Gabon</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>58</td>
<td>Nahrwan</td>
<td>Masirah Island (Oman)</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>59</td>
<td>Nahrwan</td>
<td>United Arab Emirates</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>60</td>
<td>Nahrwan</td>
<td>Saudi Arabia</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>61</td>
<td>Naparima, BWI</td>
<td>Trinidad and Tobago</td>
<td>International</td>
</tr>
<tr>
<td>109</td>
<td>Netherlands</td>
<td>Netherlands</td>
<td>Bessel</td>
</tr>
<tr>
<td>31</td>
<td>New Zealand Geodetic Datum 1949 (NZGD 49)</td>
<td>New Zealand</td>
<td>International</td>
</tr>
<tr>
<td>1010</td>
<td>(New Zealand (NZGD49 7-parameter)</td>
<td>New Zealand</td>
<td>International</td>
</tr>
<tr>
<td>63</td>
<td>North American 1927 (NAD 27)</td>
<td>Alaska</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>64</td>
<td>North American 1927 (NAD 27)</td>
<td>Bahamas (excluding San Salvador Island)</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>65</td>
<td>North American 1927 (NAD 27)</td>
<td>San Salvador Island</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>Number</td>
<td>Datum</td>
<td>Area Maps</td>
<td>Ellipsoid</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>66</td>
<td>North American 1927 (NAD 27)</td>
<td>Canada (including Newfoundland Island)</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>67</td>
<td>North American 1927 (NAD 27)</td>
<td>Canal Zone</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>68</td>
<td>North American 1927 (NAD 27)</td>
<td>Caribbean (Turks and Caicos Islands)</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>69</td>
<td>North American 1927 (NAD 27)</td>
<td>Central America (Belize, Costa Rica, El Salvador, Guatemala, Honduras, Nicaragua)</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>70</td>
<td>North American 1927 (NAD 27)</td>
<td>Cuba</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>71</td>
<td>North American 1927 (NAD 27)</td>
<td>Greenland (Hayes Peninsula)</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>72</td>
<td>North American 1927 (NAD 27)</td>
<td>Mexico</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>73</td>
<td>North American 1927 (NAD 27)</td>
<td>Michigan (used only for State Plane Coordinate System 1927)</td>
<td>Modified Clarke 1866</td>
</tr>
<tr>
<td>74</td>
<td>North American 1983 (NAD 83)</td>
<td>Alaska, Canada, Central America, Continental US, Mexico</td>
<td>GRS 80</td>
</tr>
<tr>
<td>139</td>
<td>North Sahara 1959</td>
<td>Algeria</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>107</td>
<td>Nouvelle Triangulation Francaise (NTF) Greenwich Prime Meridian</td>
<td>France</td>
<td>Modified Clarke 1880</td>
</tr>
<tr>
<td>1002</td>
<td>Nouvelle Triangulation Francaise (NTF) Paris Prime Meridian</td>
<td>France</td>
<td>Modified Clarke 1880</td>
</tr>
<tr>
<td>111</td>
<td>NWGL 10</td>
<td>Worldwide</td>
<td>WGS 72</td>
</tr>
<tr>
<td>117</td>
<td>NZGD 2000</td>
<td>New Zealand</td>
<td>GRS 80</td>
</tr>
<tr>
<td>1010</td>
<td>NZGD 49, 7 parameter</td>
<td>New Zealand</td>
<td>International</td>
</tr>
<tr>
<td>75</td>
<td>Observatorio 1966</td>
<td>Corvo and Flores Islands (Azores)</td>
<td>International</td>
</tr>
<tr>
<td>Number</td>
<td>Datum</td>
<td>Area Maps</td>
<td>Ellipsoid</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>140</td>
<td>Observatorio Meteorologico 1939</td>
<td>Corvo and Flores Islands (Azores)</td>
<td>International 1924</td>
</tr>
<tr>
<td>76</td>
<td>Old Egyptian</td>
<td>Egypt</td>
<td>Helmert 1906</td>
</tr>
<tr>
<td>77</td>
<td>Old Hawaiian</td>
<td>Hawaii</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>97</td>
<td>Old Tokyo</td>
<td>Japan, Korea, Okinawa</td>
<td>Bessel 1841</td>
</tr>
<tr>
<td>78</td>
<td>Oman</td>
<td>Oman</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>79</td>
<td>Ordnance Survey of Great Britain 1936</td>
<td>England, Isle of Man, Scotland, Shetland Islands, Wales</td>
<td>Airy</td>
</tr>
<tr>
<td>80</td>
<td>Pico de las Nieves</td>
<td>Canary Islands</td>
<td>International</td>
</tr>
<tr>
<td>81</td>
<td>Pitcairn Astro 1967</td>
<td>Pitcairn Island</td>
<td>International</td>
</tr>
<tr>
<td>141</td>
<td>Point 58</td>
<td>Burkina Faso and Niger</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>142</td>
<td>Pointe Noire 1948</td>
<td>Congo</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>143</td>
<td>Porto Santo 1936</td>
<td>Porto Santo and Madeiras Islands</td>
<td>International 1924</td>
</tr>
<tr>
<td>1000</td>
<td>Potsdam</td>
<td>Germany</td>
<td>Bessel</td>
</tr>
<tr>
<td>82</td>
<td>Provisional South American 1956</td>
<td>Bolivia, Chile, Colombia, Ecuador, Guyana, Peru, Venezuela</td>
<td>International</td>
</tr>
<tr>
<td>36</td>
<td>Provisional South Chilean 1963</td>
<td>South Chile (near 53×S)</td>
<td>International</td>
</tr>
<tr>
<td>83</td>
<td>Puerto Rico</td>
<td>Puerto Rico and Virgin Islands</td>
<td>Clarke 1866</td>
</tr>
<tr>
<td>1001</td>
<td>Pulkovo 1942</td>
<td>Germany</td>
<td>Krassovsky</td>
</tr>
<tr>
<td>1012</td>
<td>PZ90</td>
<td>Russia</td>
<td>PZ90</td>
</tr>
<tr>
<td>84</td>
<td>Qatar National</td>
<td>Qatar</td>
<td>International</td>
</tr>
<tr>
<td>85</td>
<td>Qornoq</td>
<td>South Greenland</td>
<td>International</td>
</tr>
<tr>
<td>1000</td>
<td>Rauenberg</td>
<td>Germany</td>
<td>Bessel</td>
</tr>
<tr>
<td>86</td>
<td>Reunion</td>
<td>Mascarene Island</td>
<td>International</td>
</tr>
<tr>
<td>112</td>
<td>Rikets Triangulering 1990 (RT 90)</td>
<td>Sweden</td>
<td>Bessel</td>
</tr>
<tr>
<td>Number</td>
<td>Datum</td>
<td>Area Maps</td>
<td>Ellipsoid</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>1011</td>
<td>Rikets Triangulering 1990 (RT 90), 7 parameter</td>
<td>Sweden</td>
<td>Bessel</td>
</tr>
<tr>
<td>87</td>
<td>Rome 1940</td>
<td>Sardinia Island</td>
<td>International</td>
</tr>
<tr>
<td>88</td>
<td>Santo (DOS)</td>
<td>Espírito Santo Island</td>
<td>International</td>
</tr>
<tr>
<td>89</td>
<td>São Braz</td>
<td>São Miguel, Santa Maria Islands (Azores)</td>
<td>International</td>
</tr>
<tr>
<td>90</td>
<td>Sapper Hill 1943</td>
<td>East Falkland Island</td>
<td>International</td>
</tr>
<tr>
<td>91</td>
<td>Schwarzeck</td>
<td>Namibia</td>
<td>Modified Bessel 1841</td>
</tr>
<tr>
<td>144</td>
<td>Selvagem Grande 1938</td>
<td>Salvage Islands</td>
<td>International 1924</td>
</tr>
<tr>
<td>145</td>
<td>Sierra Leone 1960</td>
<td>Sierra Leone</td>
<td>Clarke 1880</td>
</tr>
<tr>
<td>146</td>
<td>S-JTSK</td>
<td>Czech Republic</td>
<td>Bessel 1841</td>
</tr>
<tr>
<td>1013</td>
<td>SK42</td>
<td>Russia</td>
<td>PZ90</td>
</tr>
<tr>
<td>1024</td>
<td>SK95</td>
<td>Russia</td>
<td>PZ90</td>
</tr>
<tr>
<td>92</td>
<td>South American 1969</td>
<td>Argentina, Bolivia, Brazil, Chile, Colombia, Ecuador, Guyana, Paraguay, Peru, Venezuela, Trinidad, and Tobago</td>
<td>South American 1969</td>
</tr>
<tr>
<td>93</td>
<td>South Asia</td>
<td>Singapore</td>
<td>Modified Fischer 1960</td>
</tr>
<tr>
<td>94</td>
<td>Southeast Base</td>
<td>Porto Santo and Madeira Islands</td>
<td>International</td>
</tr>
<tr>
<td>95</td>
<td>Southwest Base</td>
<td>Faial, Graciosa, Pico, São Jorge, Terceira Islands (Azores)</td>
<td>International</td>
</tr>
<tr>
<td>1003</td>
<td>Switzerland (CH 1903)</td>
<td>Switzerland</td>
<td>Bessel</td>
</tr>
<tr>
<td>147</td>
<td>Tananarive Observatory 1925</td>
<td>Madagascar</td>
<td>International 1924</td>
</tr>
<tr>
<td>96</td>
<td>Timbalai 1948</td>
<td>Brunei and East Malaysia (Sarawak and Sabah)</td>
<td>Everest (India 1830)</td>
</tr>
<tr>
<td>1015</td>
<td>Tokyo</td>
<td>Japan</td>
<td>Bessel 1841</td>
</tr>
</tbody>
</table>
### Units

The following table lists the available coordinate units and the number used to identify the unit in the `MapInfoCoordinateSystemSet.xml` file:

<table>
<thead>
<tr>
<th>Number</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Centimeters</td>
</tr>
<tr>
<td>31</td>
<td>Chains</td>
</tr>
<tr>
<td>3</td>
<td>Feet (also called International Feet)*</td>
</tr>
<tr>
<td>2</td>
<td>Inches</td>
</tr>
<tr>
<td>1</td>
<td>Kilometers</td>
</tr>
<tr>
<td>30</td>
<td>Links</td>
</tr>
<tr>
<td>7</td>
<td>Meters</td>
</tr>
<tr>
<td>0</td>
<td>Miles</td>
</tr>
</tbody>
</table>
Coordinate System Origin

The origin is the point specified in longitude and latitude from which all coordinates are referenced. It is chosen to optimize the accuracy of a particular coordinate system. As we move north from the origin, Y increases. X increases as we move east. These coordinate values are generally called northings and eastings.

For the Transverse Mercator projection the origin’s longitude defines the central meridian. In constructing the Transverse Mercator projection a cylinder is positioned tangent to the earth. The central meridian is the line of tangency. The scale of the projected map is true along the central meridian.

In creating a Hotine Oblique Mercator projection it is necessary to specify a great circle that is not the equator nor a meridian. MapInfo does this by specifying one point on the ellipsoid and an azimuth from that point. That point is the origin of the coordinate system.

Standard Parallels (Conic Projections)

In conic projections a cone is passed through the earth intersecting it along two parallels of latitude. These are the standard parallels. One is to the north and one is to the south of the projection zone. To use a single standard parallel specify that latitude twice. Both are expressed in degrees of latitude.

Oblique Azimuth (Hotine Oblique Mercator)

When specifying a great circle (Hotine Oblique Mercator) using a point and an azimuth (arc), the azimuth is called the Oblique Azimuth and is expressed in degrees.

Scale Factor (Transverse Mercator)

A scale factor is applied to cylindrical coordinates to average scale error over the central area of the map while reducing the error along the east and west boundaries. The scale factor has the effect of recessing the cylinder into the earth so that it has two lines of intersection. Scale is true along these lines of intersection.
You may see the scale factor expressed as a ratio, such as 1:25000. In this case it is generally called the scale reduction. The relationship between scale factor and scale reduction is:

\[
\text{scale factor} = 1 - \text{scale reduction}
\]

In this case the scale factor would be 1-(1/25000) or 0.99996.

**False Northings and False Eastings**

Calculating coordinates is easier if negative numbers aren’t involved. To eliminate this problem in calculating State Plane and Universal Transverse Mercator coordinates, it is common to add measurement offsets to the northings and eastings. These offsets are called False Northings and False Eastings. They are expressed in coordinate units, not degrees. (The coordinate units are specified by the Units parameter.)

**Range (Azimuthal Projections)**

The range specifies, in degrees, how much of the earth you are seeing. The range can be between 1 and 180. When you specify 90, you see a hemisphere. When you specify 180 you see the whole earth, though much of it is very distorted.

**Polyconic Projection**

The following description is copied from *Map Projections – A Working Manual*, USGS Professional Paper 1395, by John P. Snyder.

The Polyconic projection, usually called the American Polyconic in Europe, achieved its name because the curvature of the circular arc for each parallel on the map is the same as it would be following the unrolling of a cone which had been wrapped around the globe tangent to the particular parallel of latitude, with the parallel traced onto the cone. Thus, there are many (“poly-”) cones involved, rather than the single cone of each regular conic projection.

The Polyconic projection is neither equal-area nor conformal. Along the central meridian, however, it is both distortion free and true to scale. Each parallel is true to scale, but the meridians are lengthened by various amounts to cross each parallel at the correct position along the parallel, so that no parallel is standard in the sense of having conformality (or correct angles), except at the central meridian. Near the central meridian, distortion is extremely small.

This projection is not intended for mapping large areas. The conversion algorithms used break down when mapping wide longitude ranges. For example, the sample table WORLD.TAB may exhibit anomalies if reprojected using Polyconic.

**Datum Conversion**

When converting coordinates from one datum to another, MapInfo has used the Molodensky (3-parameter) and Bursa-Wolf (7-parameter) methods. These are general-purpose methods that can convert coordinates from any datum to any other datum.

After the NAD 83 datum was introduced, NOAA developed a program called NADCON, which stands for North American Datum CONversion. This is a very specialized program that converts coordinates only from NAD 27 to NAD 83 and vice versa. For this specialized task, it’s much more
accurate than the Molodensky general-purpose method; NADCON is accurate to about 0.1 meter, and Molodensky is accurate to only 10–30 meters. Most U.S. government agencies, including the Census Bureau, have standardized on NADCON for converting between NAD 27 and NAD 83.

Beginning with MapInfo 4.1.2, the NADCON algorithm is used to convert coordinates between NAD 27 and NAD 83 if those coordinates lie within the areas covered by NADCON (United States, Puerto Rico, and the Virgin Islands). If the coordinates lie outside those areas, or if they use datums other than NAD 27 or NAD 83, MapInfo uses the Molodensky or Bursa-Wolfe conversion methods.

Due to the file access required, the NADCON conversion method can be slightly slower than the Molodensky method. If you want to turn off the NADCON conversion, remove the *.las and *.los files from the MapXtreme program or the MapXtreme Common directory.

Custom Datums

A datum is a mathematical description of the earth’s shape and orientation. Because the earth’s shape is not uniform, there are many different local datums used in different parts of the world. These local datums provide a close approximation to the earth’s surface in a particular area.

Each Earth coordinate system uses a specific datum to approximate the earth’s surface. If two coordinate systems use different datums, then Pitney Bowes MapInfo’s mapping products must perform a datum transformation when it converts coordinates from one coordinate system to the other. Pitney Bowes MapInfo uses the Bursa-Wolfe datum transformation method, which is generally accurate to within 10 meters. (When the conversion is between two coordinate systems that use the same datum, no datum transformation is performed, and the results are generally accurate to within 0.1 meter.)

Defining Custom Datums

Most coordinate systems use one of Pitney Bowes MapInfo’s predefined datums, listed in this appendix. If you need to use a datum that isn’t in the list, and you know what the datum’s mathematical parameters are, then you can define the coordinate system using a custom datum. MapInfo uses the following information to define a datum:

- An ellipsoid, also called a spheroid. This is an ellipse rotated around its minor axis to form a three-dimensional surface. The ellipsoid is described by two mathematical parameters: the length, in meters, of its semi-major axis (denoted by the letter a) and its degree of flattening (denoted by the letter f). MapInfo supports over 40 predefined ellipsoids, which are listed in the next table.
- Three shift parameters specifying the distance, in meters, to shift the ellipsoid along each of its axes. These parameters are usually denoted by dX, dY, and dZ. You may also see them denoted by DX, DY, and DZ, or by u, v, and w.
- Three rotation parameters specifying the angle, in arc-seconds, to rotate the ellipsoid around each of its axes. These parameters are usually denoted by EX, EY, and EZ. You may also see them denoted by eX, eY, and eZ, or by e, y, and w.
- A scale correction factor specifying the amount, in parts per million, to adjust the size of the ellipsoid. This parameter is denoted by the letter m, or sometimes k.
• The longitude of the prime meridian, in degrees east of Greenwich. The prime meridian specifies which location on earth is assigned longitude 0°. Most datums use Greenwich as the prime meridian, so this parameter is usually zero. However, some datums use a different location as the prime meridian. For example, the NTF datum uses Paris as its prime meridian, which is 2.33722917 degrees east of Greenwich. If you use the NTF datum in a coordinate system, all longitudes in that coordinate system are relative to Paris instead of Greenwich.

You can define a custom datum in any coordinate system definition. Use datum number 9999 followed by the datum parameters, in this order:

9999, EllipsoidNumber, dX, dY, dZ, EX, EY, EZ, m, PrimeMeridian

Some datums specify only an ellipsoid and shift parameters (dX, dY, dZ), with no rotation parameters, scale correction, or prime meridian. In those cases, you can use datum number 999 instead of 9999, to simplify the definition:

999, EllipsoidNumber, dX, dY, dZ

The ellipsoid number must be chosen from the following list. Currently, there is no way to define a custom ellipsoid. If you need to use an ellipsoid that does not appear on this list, please notify MapInfo Technical Support so that we can add your ellipsoid to a future MapInfo release.

<table>
<thead>
<tr>
<th>Number</th>
<th>Ellipsoid</th>
<th>a</th>
<th>1/f</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Airy 1930</td>
<td>6377563.396</td>
<td>299.3249646</td>
</tr>
<tr>
<td>13</td>
<td>Airy 1930 (modified for Ireland 1965)</td>
<td>6377340.189</td>
<td>299.3249646</td>
</tr>
<tr>
<td>51</td>
<td>ATS7 77</td>
<td>6378135.0</td>
<td>298.257</td>
</tr>
<tr>
<td>2</td>
<td>Australian</td>
<td>6378160.0</td>
<td>298.25</td>
</tr>
<tr>
<td>10</td>
<td>Bessel 1841</td>
<td>6377397.155</td>
<td>299.1528128</td>
</tr>
<tr>
<td>35</td>
<td>Bessel 1841 (modified for NGO 1948)</td>
<td>6377492.0176</td>
<td>299.15281</td>
</tr>
<tr>
<td>14</td>
<td>Bessel 1841 (modified for Schwarzeck)</td>
<td>6377483.865</td>
<td>299.1528128</td>
</tr>
<tr>
<td>36</td>
<td>Clarke 1858</td>
<td>6378293.639</td>
<td>294.26068</td>
</tr>
<tr>
<td>7</td>
<td>Clarke 1866</td>
<td>6378206.4</td>
<td>294.9786982</td>
</tr>
<tr>
<td>8</td>
<td>Clarke 1866 (modified for Michigan)</td>
<td>6378450.047484481</td>
<td>294.9786982</td>
</tr>
<tr>
<td>6</td>
<td>Clarke 1880</td>
<td>6378249.145</td>
<td>293.465</td>
</tr>
<tr>
<td>15</td>
<td>Clarke 1880 (modified for Arc 1950)</td>
<td>6378249.145326</td>
<td>293.4663076</td>
</tr>
<tr>
<td>30</td>
<td>Clarke 1880 (modified for IGN)</td>
<td>6378249.2</td>
<td>293.4660213</td>
</tr>
<tr>
<td>37</td>
<td>Clarke 1880 (modified for Jamaica)</td>
<td>6378249.136</td>
<td>293.46631</td>
</tr>
<tr>
<td>16</td>
<td>Clarke 1880 (modified for Merchich)</td>
<td>6378249.2</td>
<td>293.46598</td>
</tr>
<tr>
<td>Number</td>
<td>Ellipsoid</td>
<td>a</td>
<td>1/f</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>38</td>
<td>Clarke 1880 (modified for Palestine)</td>
<td>6378300.79</td>
<td>293.46623</td>
</tr>
<tr>
<td>39</td>
<td>Everest (Brunei and East Malaysia (Sabah and Sarawak))</td>
<td>6377298.556</td>
<td>300.8017</td>
</tr>
<tr>
<td>11</td>
<td>Everest (India 1830)</td>
<td>6377276.345</td>
<td>300.8017</td>
</tr>
<tr>
<td>40</td>
<td>Everest (India 1956)</td>
<td>6377301.243</td>
<td>300.80174</td>
</tr>
<tr>
<td>50</td>
<td>Everest (Pakistan)</td>
<td>6377309.613</td>
<td>300.8017</td>
</tr>
<tr>
<td>17</td>
<td>Everest (W. Malaysia and Singapore 1948)</td>
<td>6377304.063</td>
<td>300.8017</td>
</tr>
<tr>
<td>48</td>
<td>Everest (West Malaysia 1969)</td>
<td>6377295.664</td>
<td>300.8017</td>
</tr>
<tr>
<td>18</td>
<td>Fischer 1960</td>
<td>6378166.0</td>
<td>298.3</td>
</tr>
<tr>
<td>19</td>
<td>Fischer 1960 (modified for South Asia)</td>
<td>6378155.0</td>
<td>298.3</td>
</tr>
<tr>
<td>20</td>
<td>Fischer 1968</td>
<td>6378150.0</td>
<td>298.3</td>
</tr>
<tr>
<td>21</td>
<td>GRS 67</td>
<td>6378160.0</td>
<td>298.247167427</td>
</tr>
<tr>
<td>0</td>
<td>GRS 80</td>
<td>6378137.0</td>
<td>298.257222101</td>
</tr>
<tr>
<td>5</td>
<td>Hayford</td>
<td>6378388.0</td>
<td>297.0</td>
</tr>
<tr>
<td>22</td>
<td>Helmert 1906</td>
<td>6378200.0</td>
<td>298.3</td>
</tr>
<tr>
<td>23</td>
<td>Hough</td>
<td>6378270.0</td>
<td>297.0</td>
</tr>
<tr>
<td>31</td>
<td>IAG 75</td>
<td>6378140.0</td>
<td>298.257222</td>
</tr>
<tr>
<td>41</td>
<td>Indonesian</td>
<td>6378160.0</td>
<td>298.247</td>
</tr>
<tr>
<td>4</td>
<td>International 1924</td>
<td>6378388.0</td>
<td>297.0</td>
</tr>
<tr>
<td>49</td>
<td>Irish (WOFO)</td>
<td>6377542.178</td>
<td>299.325</td>
</tr>
<tr>
<td>3</td>
<td>Krassovksy</td>
<td>6378245.0</td>
<td>298.3</td>
</tr>
<tr>
<td>32</td>
<td>MERIT 83</td>
<td>6378137.0</td>
<td>298.257</td>
</tr>
<tr>
<td>33</td>
<td>New International 1967</td>
<td>6378157.5</td>
<td>298.25</td>
</tr>
<tr>
<td>43</td>
<td>NWL 10D</td>
<td>6378135.0</td>
<td>298.26</td>
</tr>
<tr>
<td>42</td>
<td>NWL 9D</td>
<td>6378145.0</td>
<td>298.25</td>
</tr>
<tr>
<td>44</td>
<td>OSU86F</td>
<td>6378136.2</td>
<td>298.25722</td>
</tr>
<tr>
<td>45</td>
<td>OSU91A</td>
<td>6378136.3</td>
<td>298.25722</td>
</tr>
</tbody>
</table>
The shift and rotation parameters describe the ellipsoid’s orientation in space, as compared to the WGS 84 datum. It’s important to make sure that these parameters have the correct signs (positive or negative). Usually, a document describing a local datum will list the parameters required to convert coordinates from the local datum to WGS 84. (This is the same as saying that the parameters were derived by subtracting the local datum from WGS 84.) In that case, you can use the parameters exactly as they appear in the document. However, if you have a document that lists parameters for converting coordinates in the opposite direction — from WGS 84 to the local datum — then you must reverse the signs of the shift, rotation, and scale correction parameters.

It’s also very important to list the parameters in the correct order. Some documents list the rotation parameters with EZ first, like this: EZ, EY, EX. In those cases, you must reverse the order of the rotation parameters when defining the custom datum. This is especially easy to overlook when your document uses Greek letters to denote the parameters. If the document lists the parameters in order as \( \omega, \gamma, e \), then you must reverse their order in the custom datum definition.

Here’s an example of a local datum description (we’ll call it LD-2) as it might appear in a technical article:

LD-2 ellipsoid: Krassovsky

<table>
<thead>
<tr>
<th>Number</th>
<th>Ellipsoid</th>
<th>( a )</th>
<th>( 1/f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>46</td>
<td>Plessis 1817</td>
<td>6376523.0</td>
<td>308.64</td>
</tr>
<tr>
<td>52</td>
<td>PZ90</td>
<td>6378136.0</td>
<td>298.257839303</td>
</tr>
<tr>
<td>24</td>
<td>South American</td>
<td>6378160.0</td>
<td>298.25</td>
</tr>
<tr>
<td>12</td>
<td>Sphere</td>
<td>6370997.0</td>
<td>0.0</td>
</tr>
<tr>
<td>47</td>
<td>Struve 1860</td>
<td>6378297.0</td>
<td>294.73</td>
</tr>
<tr>
<td>34</td>
<td>Walbeck</td>
<td>6376896.0</td>
<td>302.78</td>
</tr>
<tr>
<td>25</td>
<td>War Office</td>
<td>6378300.583</td>
<td>296.0</td>
</tr>
<tr>
<td>26</td>
<td>WGS 60</td>
<td>6378165.0</td>
<td>298.3</td>
</tr>
<tr>
<td>27</td>
<td>WGS 66</td>
<td>6378145.0</td>
<td>298.25</td>
</tr>
<tr>
<td>1</td>
<td>WGS 72</td>
<td>6378135.0</td>
<td>298.26</td>
</tr>
<tr>
<td>28</td>
<td>WGS 84</td>
<td>6378137.0</td>
<td>298.257223563</td>
</tr>
</tbody>
</table>

The shift and rotation parameters describe the ellipsoid’s orientation in space, as compared to the WGS 84 datum. It’s important to make sure that these parameters have the correct signs (positive or negative). Usually, a document describing a local datum will list the parameters required to convert coordinates from the local datum to WGS 84. (This is the same as saying that the parameters were derived by subtracting the local datum from WGS 84.) In that case, you can use the parameters exactly as they appear in the document. However, if you have a document that lists parameters for converting coordinates in the opposite direction — from WGS 84 to the local datum — then you must reverse the signs of the shift, rotation, and scale correction parameters.

It’s also very important to list the parameters in the correct order. Some documents list the rotation parameters with EZ first, like this: EZ, EY, EX. In those cases, you must reverse the order of the rotation parameters when defining the custom datum. This is especially easy to overlook when your document uses Greek letters to denote the parameters. If the document lists the parameters in order as \( \omega, \gamma, e \), then you must reverse their order in the custom datum definition.

Here’s an example of a local datum description (we’ll call it LD-2) as it might appear in a technical article:

LD-2 ellipsoid: Krassovsky

\[
a = 6378245.0 \text{ m}
\]

\[
f = 1 / 298.3
\]

Converting from LD-2 to WGS 84
This datum uses the Krassovsky ellipsoid, which is number 3 in the ellipsoid table above. We do not need to reverse the signs of the parameters, since they describe a conversion from the local datum to WGS 84. However, the rotation parameters are listed with \( w \) first, so we must reverse their order in the custom datum definition:

\[
9999, 3, 24, -123, -94, -0.02, 0.25, 0.13, 1.1, 0
\]

Here’s a final example, LD-3, that provides only the ellipsoid and shift parameters:

LD-3 ellipsoid: Clarke 1880

\[
\begin{align*}
a & = 6378249.145 \text{ m} \\
f & = 1/293.465
\end{align*}
\]

WGS 84 minus local datum LD-3

\[
\begin{align*}
dX & = -7 \text{ m} \\
dY & = 36 \text{ m} \\
dZ & = 225 \text{ m}
\end{align*}
\]

This datum uses the Clarke 1880 ellipsoid, which is number 6 in the ellipsoid table above. We do not need to reverse the signs of the parameters or worry about the order of the rotation parameters (since they aren’t present). In this case, you can use datum number 999 instead of 9999 in the custom datum definition. These two definitions are equivalent, and you can use either one:

\[
999, 6, -7, 36, 225
\]

\[
9999, 6, -7, 36, 225, 0, 0, 0, 0, 0
\]

As with the other custom datum definitions, you would insert one of these definitions in place of the datum number in a MapInfoCoordinateSystemSet.xml line, as follows:

"Longitude / Latitude (LD-3)", 1, 999, 6, -7, 36, 225

<rsName>UTM Zone 30 (LD-3)</rsName>
Information on Coordinate Systems and Projections

The first three publications listed are relatively short pamphlets. The last two are substantial books. We've also given addresses and phone numbers for the American Congress of Surveying and Mapping (the pamphlets) and the U.S. Geological Survey (the books).


Contact Information
- American Congress on Surveying and Mapping, 5410 Grosvenor Lane, Suite 100, Bethesda, MD 20814 2212; (301) 493-0200
- Earth Science Information Center, U.S. Geological Survey, 507 National Center, Reston, VA 22092; (703) 860-6045 or (800) USA-MAPS
- Peter H. Dana of the Department of Geography, University of Texas at Austin has also put up a website for explanations of Map projections, Geodetic Datums, and Coordinate systems. It is a valuable as many of these explanations were also presented using MapInfo Professional. The materials may be used for study, research, and education, but please credit the author:
  - Peter H. Dana, The Geographer’s Craft Project, Department of Geography, The University of Texas at Austin.

For Geodetic Datum information and explanations, go to:

http://www.colorado.edu/geography/gcraft/notes/datum/datum.html

For Information on Coordinate systems and other principles, go to:

http://www.colorado.edu/geography/gcraft/notes/coordsys/coordsys.html

For Information on Map Projections go to:

http://www.colorado.edu/geography/gcraft/notes/mapproj/mapproj.html
User-Defined Metadata

This appendix contains information on user-defined metadata support for TableInfoServer queries.

In this appendix

- Metadata and the MapCatalog .......................... 542
- User-Defined Metadata Support for TableInfoServer Queries . 542
Metadata and the MapCatalog

When working with spatial database queries, the MapCatalog has long been used as the sole source of metadata when trying to infer accurate definitions for FeatureGeometry and Style columns in the query result set. In some cases, however, the MapCatalog may not be a convenient, appropriate, or reliable source. For example:

- Views—When defining view tables in the remote database server, the onus is upon the database administrator to add a corresponding entry into the MapCatalog. This may be inconvenient, and there is a risk of inaccuracy in the content.
- Stored Procedures—As a potentially dynamic generator of spatial result sets, it may not be feasible to place a single, static entry into the MapCatalog that accurately represents all of the result sets capable of being generated as output to invocations of the stored procedure.
  **Note** Stored Procedure queries are currently supported for SQL Server only.
- Tables containing multiple spatial columns—Although spatial databases may allow for tables to contain more than one spatial column, the MapCatalog schema currently only permits the metadata to be defined for a single spatial column of a table.
- Complex queries—Even when the MapCatalog contains accurate information for base tables referenced in complex queries, the inferencing logic may be unreliable for several reasons:
  - It may be difficult for the metadata inferencing logic to properly locate the correct metadata (e.g., complex join queries).
  - The MapCatalog metadata may be inaccurate due to functions applied on the geometries (e.g., coordinate system transformations, buffering points, convex hull aggregations, etc.).
  - The inferencing logic may not know how to properly reconcile metadata identified for two or more base tables (e.g., UNION queries).

User-Defined Metadata Support for TableInfoServer Queries

The MapInfo.Data.TableInfoServer class contains properties that enable you to define metadata for FeatureGeometry and Style columns in a table. This user-defined metadata, or column hints, identify these columns in the absence of a MapCatalog.

**ColumnHints Property**

The ColumnHints property added to TableInfoServer gives users an explicit means of providing user-defined metadata to influence the proper identification of FeatureGeometry and Style columns in a query result set. These "hints" are principally intended for providing complete metadata in cases where MapCatalog metadata could not be located; however, these hints may also be used as potentially sparse definitions to selectively override components of the MapCatalog metadata. In either case, user-defined metadata is considered definitive, so users are cautioned to use column hints only when necessary, and when the details of the query results are well understood.

The ColumnHints property is defined as a Columns collection. After creating an instance of the collection, and inserting Column instances for each hint within it, this collection is assigned to the ColumnHints property; for example, for a TableInfoServer variable named tblInfoSrv…
Appendix I: User-Defined Metadata

[C#]
Columns hints = new Columns();
// insert Column instances into the collection to serve as hints (see below)
tblInfoSrv.ColumnHints = hints;

Column instances for types other than FeatureGeometry and Style may be added into the Columns collection for the hints, but these other column definitions will currently be ignored.

Applying a FeatureGeometry Column Hint

A FeatureGeometry column hint may be used to indicate that a named column within the query result contains spatial values to be interpreted as feature geometries. For example, if a query returns a column named geo that contains spatial objects (e.g., a SpatialWare ST_Spatial binary, or an Oracle Spatial MDSYS.SDO_GEOMETRY structure), we can identify this column as such by constructing a hint and adding it into the Columns collection (from above) as follows:

[C#]
GeometryColumn geoCol = new GeometryColumn("geo");
...
hints.Add(geoCol);

The ellipsis in the code fragment above indicates that additional properties of the GeometryColumn instance will also need to be set before completing the hint and adding it into the Columns collection. FeatureGeometry hints are principally intended to be used for explicitly providing the same information that the MapCatalog provides, specifically:

- Coordinate System
- FeatureGeometry Type
- Default View
- Default Style

Taking all of these properties into account, a more complete example of a FeatureGeometry hint might look something like this:

[C#]
CoordSysFactory cfs = new CoordSysFactory();
CoordSys cs = cfs.CreateFromMapBasicString("Earth Projection 1, 0");
GeometryColumn geo = new GeometryColumn("geo", cs);
geo.PredominantGeometryType = GeometryType.Polygon;
geo.DefaultView = new DRect(-74, 40, -70, 44);
geo.DefaultStyle = new MapInfo.Styles.AreaStyle(
    new MapInfo.Styles.SimpleLineStyle(new LineWidth(2.0,
        MapInfo.Styles.LineWidthUnit.Pixel)),
    new MapInfo.Styles.SimpleInterior(10));
hints.Add(geo);

If a MapCatalog entry is known to exist which provides default metadata for the spatial column, a "sparse" hint can be used as a means of overriding specific properties. Usually these would be preference variety properties such as default view and default style, although any set of the properties can be overridden depending upon the unique requirements of the complex query.
ColumnHints are not intended to be used for (x,y) data tables. For these, SpatialSchemaXY should continue to be used. When a SpatialSchemaXY is applied to a table, any associated ColumnHints are disregarded.

Since a MapXtreme table currently supports only a single FeatureGeometry column, the Columns collection is expected to contain no more than a single FeatureGeometry column definition; however, there's nothing in the Columns collection that precludes users from adding in as many as they'd like. In this case, the last non-sparse value encountered for each property is the one that will be honored in compiling the final metadata definition.

**Applying a Style Column Hint**

A Style column hint may be applied as a way of indicating that a column within the query result contains values (notably, MapBasic style strings) used to create Style instances for each feature, which correspond to the associated FeatureGeometry within the feature. For example, if a query returns a column named mb_style that contains MapBasic style strings, we could construct the hint and add it into the hints collection (from above) as follows:

[C#]
```csharp
Column styleCol = new Column("mb_style", MIDbType.Style);
hints.Add(styleCol);
```

Since a table currently supports only a single Style column, the hints collection is expected to contain no more than a single Style column definition; however, there's nothing in the Columns collection that precludes users from adding in as many as they'd like. In this case, the last one inserted is the one that will be honoed.

Style column hints are not required within the hints collection. If the query does not contain a column containing style information, this hint can be excluded.

**KeyType and KeyColumns Properties**

Each table must have a "key" that provides a unique value for each feature (row) within the table. While not supplied by the MapCatalog, this is another important piece of metadata that is inferences for remote database tables by looking at things like primary key definitions, auto-assigned columns (e.g., IDENTITY columns on SQL Server), pseudo-columns (e.g., ROWID on Oracle), etc.

Many of the content sources listed above, as well as tables containing only composite primary keys, also present challenges for inferring a unique key to use. In these situations, the KeyType and KeyColumns properties provided through the base TableInfo class can be used to provide explicit instructions. For example, consider a database view of customer information that contains a column named CUSTID that is known to contain unique and non-null values. We can direct this column to be used as the key for the resulting MapXtreme table as follows:

[C#]
```csharp
tblInfoSrv.KeyType = KeyType.Explicit;
StringCollection keyColumns = new StringCollection();
keyColumns.Add("CUSTID");
tblInfoSrv.KeyColumns = keyColumns;
```

In this example, the StringCollection class resides in the System.Collections.Specialized namespace.
Composite keys are allowable and would be defined by adding two or more column names into the string collection.

Explicit key definitions may also be used as overriding definitions in cases where the database table contains two or more candidates, and the developer has a specific preference for which gets used.

**Workspace Persistence**

User-defined metadata constructs for column hints and key definitions are persisted as part of the data source definition tag within a workspace file (.mws).
Sample Data Copyrights

The following sources provided the data released with MapXtreme. For more information on MapInfo products, see the “MapInfo Data Products Catalog” or contact Pitney Bowes MapInfo.
<table>
<thead>
<tr>
<th>Map Name</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>Africa Country Boundaries</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Africa Water Bodies</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Capitals</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Country Boundaries</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Asia</td>
<td></td>
</tr>
<tr>
<td>Asia Capitals</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Asia Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>European Country Boundaries</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>World Ocean (for Asia)</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Australia Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Australia Highways</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Australia Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Australia State Boundaries</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Australia State Capital Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Canada Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Canada Highways</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Canada Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Canadian Province Boundaries</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Canadian Province Capital Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>US State Boundaries</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Map Name</td>
<td>Copyright</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
</tbody>
</table>

**China MWS/Geoset**

<table>
<thead>
<tr>
<th>Map Name</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Asia Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>China Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>China Country Bdy</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>China Highways</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>China Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Ocean (for Asia)</td>
<td>MapInfo</td>
</tr>
</tbody>
</table>

**DC MWS/Geoset**

<table>
<thead>
<tr>
<th>Map Name</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dc Area Landmarks</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Dc City Boundaries</td>
<td>MapInfo</td>
</tr>
<tr>
<td>DC Highways</td>
<td>MapInfo from the Bureau of Transportation Statistics</td>
</tr>
<tr>
<td>DC Interstate Roads</td>
<td>MapInfo from the Bureau of Transportation Statistics</td>
</tr>
<tr>
<td>DC Landmarks</td>
<td>MapInfo</td>
</tr>
<tr>
<td>DC Roads</td>
<td>MapInfo from the Bureau of Transportation Statistics</td>
</tr>
<tr>
<td>DC State Roads</td>
<td>MapInfo from the Bureau of Transportation Statistics</td>
</tr>
<tr>
<td>DC Water Layer</td>
<td>MapInfo</td>
</tr>
</tbody>
</table>

**Europe MWS/Geoset**

<table>
<thead>
<tr>
<th>Map Name</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>European Capitals</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>European Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>European Country Boundaries</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>European Highways</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>European Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>European Water Bodies</td>
<td>GisNET data licensed to MapInfo by GISdata Limited.</td>
</tr>
<tr>
<td></td>
<td>© GDC Ltd 1993</td>
</tr>
<tr>
<td>Map Name</td>
<td>Copyright</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Norway Country Boundary</td>
<td>MapInfo Corporation from National Imagery and Mapping Agency (NIMA)</td>
</tr>
<tr>
<td>Switzerland Country Boundary</td>
<td>MapInfo Corporation from National Imagery and Mapping Agency (NIMA)</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
<tr>
<td><strong>France MWS/Geoset</strong></td>
<td></td>
</tr>
<tr>
<td>European Country Boundaries</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>France Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>France Highway Map</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>France Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
<tr>
<td><strong>Germany MWS/Geoset</strong></td>
<td></td>
</tr>
<tr>
<td>European Country Boundaries</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>Germany Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Germany Highways</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>Germany Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
<tr>
<td><strong>India MWS/Geoset</strong></td>
<td></td>
</tr>
<tr>
<td>Asia</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>India Capital Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>India District Bdys</td>
<td>Risk Management Solutions, Inc.</td>
</tr>
<tr>
<td>India Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>India Minor Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>India State Bdys</td>
<td>Risk Management Solutions, Inc.</td>
</tr>
<tr>
<td>World Ocean (for Asia)</td>
<td>MapInfo</td>
</tr>
<tr>
<td><strong>Italy MWS/Geoset</strong></td>
<td></td>
</tr>
<tr>
<td>European Country Boundaries</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>Map Name</td>
<td>Copyright</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Italy Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Italy Highways</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>Italy Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
<tr>
<td><strong>Japan MWS/Geoset</strong></td>
<td></td>
</tr>
<tr>
<td>Japan Cased Roads</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>Japan Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Japan Country background</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>Japan Highways</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>Japan Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Japan Rivers and Lakes</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
<tr>
<td><strong>Mexico MWS/Geoset</strong></td>
<td></td>
</tr>
<tr>
<td>Mexico Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Mexico Highways</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Mexico Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Mexico State Boundaries</td>
<td>MapInfo from the Bureau of Transportation Statistics</td>
</tr>
<tr>
<td>Mexico State Capital Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>US State Boundaries</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
<tr>
<td><strong>North American MWS/Geoset</strong></td>
<td></td>
</tr>
<tr>
<td>Canada Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Canada Highways</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Map Name</td>
<td>Copyright</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------</td>
</tr>
<tr>
<td>Canada Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Canadian Province Boundaries</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Canadian Province Capital Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Mexico Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Mexico Highways</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Mexico Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Mexico State Boundaries</td>
<td>MapInfo from the Bureau of Transportation Statistics</td>
</tr>
<tr>
<td>Mexico State Capital Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>US Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>US County Boundaries</td>
<td>MapInfo from Census Bureau</td>
</tr>
<tr>
<td>US Highways</td>
<td>MapInfo from the Bureau of Transportation Statistics</td>
</tr>
<tr>
<td>US Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>US State Boundaries</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>US State Capitals</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
</tbody>
</table>

**Portugal MWS/Geoset**

<table>
<thead>
<tr>
<th>Map Name</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>European Country Boundaries</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>Portugal Highways</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>Portugal Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
</tbody>
</table>

**Spain MWS/Geoset**

<table>
<thead>
<tr>
<th>Map Name</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>European Country Boundaries</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>Spain Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Spain Highways</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>Spain Major Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Map Name</td>
<td>Copyright</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
</tbody>
</table>

**StreetPro Samples**

<table>
<thead>
<tr>
<th>Map Name</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>StreetPro Austria</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Benelux</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Canada</td>
<td>Tele Atlas/GDT Canada (c) 2005, 2006</td>
</tr>
<tr>
<td>StreetPro Denmark</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Deutschland</td>
<td>(c) 2005 GDTTeleAtlas.</td>
</tr>
<tr>
<td>StreetPro Espana</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro France</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Great Britain</td>
<td>(c) 2005 GDTTeleAtlas.</td>
</tr>
<tr>
<td>StreetPro Iberia</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Ireland</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Italia</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Netherlands</td>
<td>(c) 2005 GDTTeleAtlas.</td>
</tr>
<tr>
<td>StreetPro Norway</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Portugal</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Sweden</td>
<td>(c) 2005 GDTTeleAtlas</td>
</tr>
<tr>
<td>StreetPro Switzerland</td>
<td>(c) 2005 GDTTeleAtlas.</td>
</tr>
<tr>
<td>StreetPro US (Washington, DC)</td>
<td>MapInfo</td>
</tr>
</tbody>
</table>

**UK MWS/Geoset**

<table>
<thead>
<tr>
<th>Map Name</th>
<th>Copyright</th>
</tr>
</thead>
<tbody>
<tr>
<td>European Country Boundaries</td>
<td>MapInfo 1998</td>
</tr>
<tr>
<td>United Kingdom Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>United Kingdom Class A Roads</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>United Kingdom Motorways</td>
<td>GisNET data licensed to MapInfo by GISdata Limited. © GDC Ltd. 1993</td>
</tr>
<tr>
<td>World Ocean (Lat / Long)</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Map Name</td>
<td>Copyright</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td><strong>US MWS/Geoset</strong></td>
<td></td>
</tr>
<tr>
<td>Urban_CitiesPop10K_Plus</td>
<td>MapInfo</td>
</tr>
<tr>
<td>USA_Caps</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>US_Int_Shields1</td>
<td>MapInfo</td>
</tr>
<tr>
<td>US_Int_Shields2</td>
<td>MapInfo</td>
</tr>
<tr>
<td>US_Int_Shields3</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Interstates</td>
<td>US Department of the Interior: nationalatlas.gov</td>
</tr>
<tr>
<td>Secondary_Rds</td>
<td>US Department of the Interior: nationalatlas.gov</td>
</tr>
<tr>
<td>Streams_Rivers</td>
<td>US Department of the Interior: nationalatlas.gov</td>
</tr>
<tr>
<td>Lakes</td>
<td>US Department of the Interior: nationalatlas.gov</td>
</tr>
<tr>
<td>Urban_Areas</td>
<td>US Department of the Interior: nationalatlas.gov</td>
</tr>
<tr>
<td>Wilderness_Areas</td>
<td>US Department of the Interior: nationalatlas.gov</td>
</tr>
<tr>
<td>USA_Outline</td>
<td>MapInfo</td>
</tr>
<tr>
<td><strong>World MWS/Geoset</strong></td>
<td></td>
</tr>
<tr>
<td>World Capitals</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Countries</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>World Graticule</td>
<td>MapInfo</td>
</tr>
<tr>
<td>World Ocean (Robinson)</td>
<td>MapInfo</td>
</tr>
<tr>
<td>World Top 25 Cities</td>
<td>MapInfo from Digital Chart of the World</td>
</tr>
<tr>
<td>Mgrs_100K</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Mgrs_Nor_100K</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Mgrs_sou_100K</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Mgrs_UPS_Nor</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Mgrs_UPS_Sou</td>
<td>MapInfo</td>
</tr>
<tr>
<td>MGRS_Zones</td>
<td>MapInfo</td>
</tr>
<tr>
<td>Map Name</td>
<td>Copyright</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Other Files</td>
<td></td>
</tr>
<tr>
<td>US Raster images</td>
<td>Sure!MAPS ® RASTER © HTI 1995</td>
</tr>
<tr>
<td>US Grid</td>
<td>Public Domain</td>
</tr>
<tr>
<td>US Demographic data</td>
<td>US Census 2000</td>
</tr>
<tr>
<td>Business Points, Insurance, Telco,</td>
<td>MapInfo and others</td>
</tr>
<tr>
<td>Weather</td>
<td></td>
</tr>
</tbody>
</table>
This glossary defines terms used in this guide and in MapInfo products that are necessarily understandable or are used in a way specific to MapInfo products and technology.
Terms

Adornment
A MapXtreme map element that consists of either a legend, title, or scalebar.

Anti-aliasing
Smoothes the jagged edges of lines, curves, and the edges of filled areas when representing a high-definition rendition at a lower resolution.

Cartesian
A coordinate system using an x,y scale not tied to any real-world system. Most CAD drawing uses this method of registering objects (for example, a drawing of a ball-bearing assembly or a floor plan). If a drawing uses Cartesian coordinates, one corner of the drawing probably has coordinates 0, 0.

Cartesian Coordinates
The conventional representation of geometric objects by x and y values on a plane.

Centroid
Usually the center of a map object. For most map objects, the centroid is located at the middle of the object (the location halfway between the northern and southern extents and halfway between the eastern and western extents of the object). In some cases, the centroid is not at the middle point because there is a restriction that the centroid must be located on the object itself. Thus, in the case of a crescent-shaped region object, the middle point of the object may actually lie outside the limits of the region; however, the centroid is always within the limits of the region.

Character Encoding
A method of converting a sequence of bytes into a sequence of characters. See also Universal Character Set (UCS) on page 563 and Unicode Transformation Format-8 (UTF-8) on page 563.

Class
In an object-oriented language, a class is an object or a set of objects that contain(s) methods for performing some type of function, similar in meaning to a derived type in procedural languages.

Codespace
See MapInfo Codespace on page 561.

Convex Hull Buffer
A type of buffer that creates a region object that represents a polygon based on the nodes from the input object. You can think of the convex hull polygon as an operator that places a rubber band around all of the points. It consists of the minimum number of points so that all points lie on or inside the polygon. With convex hull buffers, no interior angle can be greater than 180 degrees.

COM+ Pooling
A Microsoft component service in which objects are pre-loaded and pooled to save resources.

Coordinate
An x,y location in a Cartesian coordinate system, or a Latitude, Longitude location in an earth coordinate system. Coordinates represent locations on a map relative to other locations. Earth coordinate systems may use the equator and the Greenwich prime meridian as fixed reference points. Plane coordinate systems describe a two-dimensional x,y location in terms of distance from a fixed reference and are usually in the first quadrant so that all coordinates are positive numbers.
Coordinate System
A coordinate system is used to create a numerical representation of geometric objects. Each point in a geometric object is represented by a pair of numbers. Those numbers are the coordinates for that point. In cartography, coordinate systems are closely related to projections. You create a coordinate system by supplying specific values for the parameters of a projection.

Data Binding
The association of a data source with a server control. The MapXtreme DataBinding class contains information about a single data-binding expression in an ASP.NET server control, which allows rapid-application development (RAD) designers in Visual Studio to create data-binding expressions at design time.

Decimal Degree
The decimal representation of fractions of degrees. Many paper maps express coordinates in degrees, minutes, seconds (for example, 40_30i10I), where minutes and seconds are fractions of degrees. Thirty minutes equal half a degree, and 30 seconds equal half a minute. MapXtreme expresses coordinates in decimal degrees (e.g., 72.558 degrees). Thus, the longitude: 40 degrees, 30 minutes, would be expressed in MapXtreme as 40.5 degrees.

Degrees Longitude, Degrees Latitude, Decimal Degrees
Degrees longitude and degrees latitude are coordinates used to represent locations on the surface of the earth. Longitude, or X-coordinate, represents a location's east-west position, where any location west of the prime meridian has a negative X value. Latitude, or Y-coordinate, represents a location's north-south position, where any location south of the equator has a negative Y value.

Derived Class
A class that contains all of the features of its base class but contains either additional functionality or enhanced functionality with respect to its base.

Enumerate
A type that contains all of the variables and their possible values.

Event Handler
An attribute for an object on a page that can be written in either JavaScript or VBScript. For example, an event handler describes what to do when a user clicks a button or selects text in a list box. Both VBScript and JavaScript support explicitly defined event handlers, for example, on Click and on Select. In addition, you can define functions that replace the explicit event handlers. Such functions are called implicit event handlers.

Feature
A row in a table that has geometry, style, and attributes. A Feature usually has a Table and Key to identify which row it represents.

FeatureLayer
A MapXtreme layer that displays Features from a Table. For example, a layer of region objects representing world countries is a FeatureLayer. A FeatureLayer must be added to a Map via the Map’s Layer collection. FeatureLayers can be native .TAB data, remote RDB, seamless or raster data.

Geocode
The process of assigning X and Y coordinates to records in a table or database so that the records can be displayed as objects on a map.
GeoDictionary
A MapXtreme file that contains information about tables (TAB files only). The GeoDictionary is used to automatically determine the table to which application data should be bound.

GeoDictionaryManager
A MapXtreme tool for maintaining the GeoDictionary.

Geographic Information System (GIS)
An organized collection of computer hardware and software designed to efficiently create, manipulate, analyze, and display all types of geographically or spatially referenced data.

Geometric Centroid
A centroid point (see Centroid on page 558) that does not need to be contained within the object (usually a FeatureGeometry).

Geography Markup Language (GML)
A markup language specific to mapping. The GML is being developed by the Open GIS Consortium (OGC), an international organization that develops and promotes geographic standards.

Graticule
A grid of horizontal (latitude) and vertical (longitude) lines displayed on an earth map, spaced at a regular distance (for example, every five degrees, every fifteen degrees). Used to establish a frame of reference.

Hypertext Markup Language (HTML)
A plain text (Ascii) language that enables developers to create Web pages that can be displayed by different Web browsers on different computing platforms. Html uses tags to specify the structure of the various parts of a document. Html supports links (using URLs) that point to other web documents and files.

Hypertext Transfer Protocol (HTTP)
The message-based network interface between a Web client and a Web server. HTTP runs on top of TCP/IP.

Internet Information Services (IIS)
The software services provided by Microsoft that support the creation, configuration, and management of web sites. Specifically some comment Internet Information Services include: FTP (File Transfer Protocol) and SMTP (Simple Mail Transfer Protocol). In MapXtreme, the WMS Server we provide must be configured to work with IIS to run the Server.

ISession Interface
The MapXtreme MapInfo.Engine namespace interface that provides the starting point for all MapXtreme applications. ISession manages the initialization of resources needed for a MapXtreme application, and defines all data and functionality applicable to an instance of an application.

Latitude
The horizontal lines on a map that increase from 0 degrees at the Equator to 90 degrees at both the North (+90.0 degrees) and South (-90.0 degrees) poles. Used to describe the north-south position of a point as measured usually in degrees or decimal degrees above or below the equator.
Layer
A basic component of map display in MapInfo products, typically consisting of several superimposed layers (e.g., a layer of street data superimposed over a layer of county or postal code boundaries). When a table appears in a Map window, it occupies a layer in that Map window. Typically, each map layer corresponds to one open table.

Longitude
The vertical lines on a map, running from the North to South poles, used to describe the east-west position of a point. The position is reported as the number of degrees east (to -180.0 degrees) or west (to +180.0 degrees) of the prime meridian (0 degrees). Lines of longitude are farthest apart at the Equator and intersect at both poles, and therefore, are not parallel.

Longitude/Latitude
The default coordinate system for representing geographic objects in a map in MapInfo products.

MapControl
A MapXtreme object that enables you to view a map on a form. The MapControl owns the window that the map draws to. It also controls the size of the map and interacts with the map tools. MapXtreme provides desktop and web versions of MapControl.

MapInfo Codespace
A list of definitions and standards that are commonly used in creating MapInfo maps and workspaces. The MapInfo Codespace includes coordinate system settings; pen, brush, and distance settings and abbreviations; image size settings; frequently used types and their abbreviations; a list of available operators; time, date, and temperature unit settings, and abbreviations. For details, see Appendix G: Defining the MapInfo Codespace.

Meridian
A line or a portion of a line running from the North to the South pole. A longitudinal line.

Namespace
A hierarchal naming system that provides a way to group classes together independently of inheritance. For example, two unrelated classes with the same name can exist in different namespaces: System.Utilities.FileFinder and MyCompany.Utilities.FileFinder could have the same name, but different functionality. Namespaces also help to prevent the compiler from referencing the wrong class (a 'collision').

Non-Earth Map
A map whose objects are not explicitly referenced to locations on the earth's surface. Floor plans are typical examples.

Persistence
Persistence refers to the way MapXtreme (and other MapInfo products) manages data and ensures that maps created using this API can be used by other MapXtreme users. Persistence is concerned with loading and saving XML-based workspaces and for parsing and publishing MapInfo Geometry objects from/to GML, and vice versa.

PointRef Schema
A spatial schema that can be applied to a non-mappable table to make it mappable. The schema references a Geometry object in another table by matching the values in a column (MatchColumn) of the non-mappable table with a column (RefColumn) in a mappable table. When the table is opened, it contains a read-only Geometry column. The table can then be added to a Map as a layer.
Pooling
To shared resources for better performance and scalability. In a MapXtreme web application, MapXtreme Session instances are available for use in a COM+ pool and ready to service requests from clients.

Projection
A mathematical model that transforms the locations of features on the earth's surface to locations on a two-dimensional surface, such as a paper map. Since a map is an attempt to represent a spherical object (the earth) on a flat surface, all projections have some degree of distortion. A map projection can preserve area, distance, shape or direction but only a globe can preserve all of these attributes. Some projections (for example, Mercator) produce maps well suited for navigation. Other projections (for example, equal-area projections, such as Lambert) produce maps well suited for visual analysis.

Region
A region is a MultiPolygon with one exterior Ring and zero or more interior Rings (holes).

Serialization
Serialization is the process of converting an object into a stream of data in order to preserve it on the server. This process is an essential part of maintaining objects in MapXtreme web applications. If the objects are not maintained the server would need to recreate the object (such as a map) for each web request.

Spatial Schema
A service that can be applied to a table to enhance its spatial capabilities. There are two types of spatial schemas in MapXtreme: PointRef and XY. Non-mappable tables that contain either a column that can be referenced to a column in a mappable table, or columns that represent XY values, can use these schemas to create a Geometry column. These tables can then be added to a Map as a layer. See the PointRef Schema and XY Schema glossary definitions for information about each type of schema.

State Management
A general term in web application development that deals with saving and restoring information from a browser session.

Table
A collection of data organized in row and column format. In MapXtreme, tables contains the data you wish to display on the map. Tables hold information that describe the features, including their geometry, style, and attributes. MapXtreme supports tables from a wide variety of sources including, native tables (MapInfo .TAB), relational database management systems (RDBMS), dBase, MS Access, ASCII files, and ESRI ShapeFiles. Speciality tables include raster, grid, seamless, views, WMS, and ADO.NET. The type of table is available through the TableInfo class. Tables are opened and closed via the Catalog in the Data namespace. See Chapter 10: Working with Data.

Uniform Resource Locator (URL)
The underlying implementation of a hypertext link or image map that contains the address of a Web page or file somewhere on the World Wide Web. A URL contains information about the network protocol to use (usually HTTP) and the path to the page or file. An URL example is “http://www.mycompany.com/index.html,” which points to the index page for the “my company” web site.
Universal Character Set (UCS)
The international standard ISO 10646 defines the Universal Character Set (UCS). UCS is a superset of all other character set standards. UCS also defines several methods for encoding a string of characters as a sequence of bytes, such as UTF-8 and UTF-16.

Unicode Transformation Format-8 (UTF-8)
An octet (8-bit) lossless encoding of Unicode characters. MapXtreme supports UTF-8 only as indicated in the workspace persistence schema.

Web Controls
An element on a web page that users interact with to send requests to the web server.

Web Map Service (WMS)
WMS is an OGC-compliant Web service that provides map images for use as layers in a mapping application. MapXtreme provides a server implementation of WMS if you wish to host a WMS server and a client for accessing any OGC-compliant (1.0.0, 1.1.0, or 1.1.1) WMS.

Web Feature Service (WFS)
An OGC-compliant Web service that offers geo-referenced map features for use in mapping applications. MapXtreme provides an implementation of WMS Basic, a read-only service and a client for accessing WMS servers.

Web server
A computer system that runs the Hypertext Transfer Protocol and Web Server software. A Web server accepts URL-based HTTP requests from a Web user's browser and sends HTML pages back to the browser. A Web server can manage one or many Web sites. A commercial server, for example, would typically have many Web sites.

Workspace
An XML-based persistence file format that allows users of MapXtreme to share maps they have created in a wide variety of environments. This is the file format that all future MapInfo products will conform to. For more about creating workspaces, see Chapter 23: Workspace Manager. For more about the structure of workspaces, see Appendix D: Understanding the MapInfo Workspace.

XY Schema
A spatial schema that can be applied to a non-mappable to make it mappable. The table must contain X and Y coordinate values, which the schema accesses to create a Geometry column for the table.
Index

A
abbreviations file
   editing 255–256, 259
   substitution keywords 256
   substitution methods 256–259
accessing data
   IDBCommand 180
accessing mappable tables 186
accessing non-mappable data 464
accessing non-mappable tables 163, 166, 178, 185–186
accessing remote tables
   attribute data 218
   cache management 219–221
   DataTable 178
   DBMS databases 210
   defining mappable tables 216
   IDBCommand 180
   MapInfo Map Catalog 221–222
   mappable tables 217
   mapping to X/Y columns 211
   ODBC connection string format 214–215
   performance 200, 219
   specifying styles 226
   with a .tab file 210
   without a .tab file 210
add tools
   adding geometry objects to a map 141
   drawing geometric features 142
adding
   dialog box to application 132
   geometry objects to a map 141
   grid image to a map 314
   raster image to a map 310
address candidates
   geocode responses 319
address dictionaries
   custom 322
address numbers
   matching to with Find 248
address ranges
   Find class result 250
AddressCandidates class 320
addresses
   Find result codes 253
   matching to with Find 246
ADO.NET
   supported table type 168
ADO.NET interfaces
   analyzing data 198
ADO.NET serialization 179
adornments
   defined 558
   map elements 240
ADRG
   supported raster format 309
AJAXDemo
   sample application 89–90
analyzing data 198
   ADO.NET interfaces 198
   Catalog 198–199
   MICommand 198
   OGC query interface 198
anti-aliasing
   defined 558
   enabling in Workspace Manager 398, 406, 408
   MapStyleControl 479
   when printing 478
   with curved labels 403
application
   architecture 68–71
   beginning state 105
   deploying 447
   desktop architecture 71
   licensing 31–32
   packaging a desktop 446
   packaging a web 452–453
   performance 118–121
Index

planning a desktop 124–125, 137
planning a web 74–75, 77, 99, 102–104, 106
startup workspace 105, 114–115

application pool settings
  health 120
  identity 120
  IIS 120
  performance 120
  recycling 120

application state
  defined 97
  thematic sample 112–113

application templates
  ASP.NET 45, 47
  MapForm 42–45

AppStateManager
  implementing 110–111

architecture
  application 68–71
  desktop application 71
  object model implementation 424
  overview 64
  web applications 69–70

ASCII
  supported table type 167

ASP.NET AJAX 89
  web controls 89–90

ASP.NET applications
  creating without MapXtreme templates 48
  session management 152
  using ASP.NET template 45, 47
  Web controls 47

ASRP
  supported raster format 309

assemblies
  creating from source code 88–89
  creating new 88–89
  redirecting to new 41–42

attribute data
  accessing 218

autolabels
  Workspace Manager 402

AvoidList class 341

Azimuthal projections 533

B

background map
  defined 96

bar themes
  creating 266

BaseGeocodeMatchCode class 320

beginning state
  application 105
  defined 97

best match from multiple candidates matches
  geocoding result codes 330

.bil files
  supported raster format 309

binding data 182

bitmap symbols 508

BitmapPointStyleRepository class
  customizing 462

bitmaps
  creating custom 283

bivariate themes 271–272

.bmp files
  supported raster format 309

Boolean expressions 203

browser session
  defined 96

browsers
  supported 28

brush styles
  in remote spatial tables 226

buffers
  selecting objects outside 204
  selecting objects within 204

building
  a desktop application 124–125, 137
  a web application 74–75, 77, 99, 102–104, 106

C

cache management
  accessing remote tables 220–221
  remote tables 219

CADRG
  supported raster format 309

CandidateAddress class
  geocoded addresses 320

cartographic legends 273

Catalog
  analyzing data 198–199
  data access 177
  search methods 193
  table registry 166

center tool
  desktop 140

centroid
  defined 558
Index

character encoding
  defined 558
  UTF-8 in workspace 469
CIB
  supported raster format 309
circle tools
  drawing circles 143
class
  defined 558
close matches
  Find class result 251
closing tables 170
codespace
  defined 558
codespace definitions 512–516
column hints 542–545
columns
  adding 205
  adding expressions 175
  supported data types 163–164
COM object model
  .NET interoperability 124
COM+ pooling
  defined 558
composite styles 280
conic projections
  standard parallels 532
constraints
  geocoding 325
contacting technical support 25
controls
  web 77–78, 80, 84–85, 91, 93
convex hull
  defined 558
coordinate systems
  adding to MapXtreme 299–302
  converting Geometry objects 298
  defined 559
  geometries 297–299, 301–302
  how to determine 299
  mapping term description 61
  origin 532–533
  publications and resources 539
  raster images 310
  registering EPSG and SRID codes 299–302
  registering in CoordSysFactory object 297
coordinate units 531
coordinates
  converting between NAD 27 and NAD 83 533–534
  converting to another datum 533–534
  defined 558
CoordSys object
  creating 297
CPU usage
  pooled web application 119
CreateThemeWizard
  Windows dialog boxes 133–136
creating
  a desktop application 124–125, 137, 436–438, 442–444
  a MapXtreme application 435
  a webXtreme application 74–75, 77, 99, 102–104, 106
LineStrings 291
CreateThemeWizard
  Windows dialog boxes 133–136
creating applications
  in Visual Studio 43–45, 47
  without templates 48
creating geometries
  curves 293
  LineStrings 291
  MultiPoint objects 290
  Points 290
  Polygons 294
  Rings 294
creating label themes
  individual value 270
  ranged 269
creating layers
  UserDrawLayer class 460
creating maps
  MapFactory 231
creating themes
  bar 266
  dot density 271
  graduated symbol 263–264
  individual values 269
  pie 265
  ranged 267
creating workspaces
  programmatically 474
  via Workspace Manager 386
curved labels 411, 413–414
curves
  creating 293
custom bitmaps
  point styles 279
custom tools
  behavior 141
customizing
  dialog boxes 137
  EngineCustomProperties class 458
  FeatureOverrideStyleModifier 458
  FeatureStyleModifier 459
  GMLFeatureCollection 463
search functions 458
styles 462
tools 462
UserDrawLayer 460
web tools 86
Windows controls 460–461
Windows tools 144
WorkSpaceLoader 463

data
analyzing 198
using in MapXtreme 182
data access
DataTable 178
deploying applications 54
ExecuteFeatureCollection 190
expressions 202–206
features 191
GeoDictionary Manager 184–185
IDBCOMmand 180
IDbCommand 180
MapInfo ADO.NET Data Provider 187
MapInfo SQL language 191
MICommand 188–189
MIConnection 187
MIDataReader 190
optimizing performance 200
planning for 177, 183, 185, 198–200
source rows 184
SQL commands 188–189
data binding
defined 559
description of 182
registering tables in GeoDictionary 418
Data namespace
overview 65
UML diagram 162
data provider
choosing 177
data sources
choosing 177
Data.Find namespace
overview 66
UML diagram 249
databases, supported 29
Date data type 163
SQL expressions 204
DateTime data type 163
datums
converting coordinates 533–534
defining custom 534–538
list of 522, 524–530
projections 522
dBase
supported table type 167
DBMS databases
accessing remote tables 210
importing spatial tables to 222
decimal degrees
defined 559
default styles
composite styles 280
defining mappable tables 216–217
degrees latitude
defined 559
degrees longitude
defined 559
 dém files
supported grid format 313
deploying
a web application 454
deploying applications
installers 49, 53
deployment applications
installers 49, 53
deployment installations 30
deployment licenses 32
derived class
defined 559
deserialization
process description 154
restoring state 111–112
desktop application
architecture 71
creating 436–438, 442–444
MapControl 128
MapTools 137
packaging 446
packaging tutorial 446
planning 124–125, 137
prototyping 124
session management 152
tutorial 436–438, 442–444
updating 41
Desktop controls 126
desktop tools namespace 67
development frameworks 28
development installations 30
development model
InProc 101–103
pooled 103
dialog boxes
adding to applications 132
CreateThemeWizard 133–136
customizing 137

Developer Guide

567
Index

distributed applications
  licensing 32

documentation set 25

dot density themes
  creating 271

drawing circles
  circle tools 143

drawing ellipses
  ellipse tools 142

drawing lines
  line tools 142

drawing points
  custom bitmap styles 279
  font point styles 280
  MapInfo 3.0 Compatible fonts 282

drawing polylines
  polyline tools 142
  styles 281

drawing rectangles
  rectangle tools 143

drawing regions
  interior styles 280
  styles 279

drawing text
  font styles 280
  styles 282

driving directions
  routing 339

DTED files
  supported grid format 313

dynamic selection 140

E

ECW files
  supported raster format 309

ellipses
  drawing ellipses 142

.emf files
  supported raster format 309

EnableTranslucency property 398, 478

Engine namespace
  exception classes 159
  overview 66

enumerate
  defined 559

EPSG codes
  registering with MapXtreme 299–302

error handling
  web controls 85

event handler
  defined 559
  web controls 85

events
  tool 145

exception classes
  Engine namespace 159

ExecuteFeatureCollection 190

exporting
  maps to an image 230

expressions
  AddColumns 205
  Boolean 203
  creating 202–206
  examples 204–206
  feature search 205
  functions 203
  InfoTip 206
  label 206
  label priority 403–404
  selecting objects outside buffer 204
  selecting objects within buffer 204
  thematic 206
  uses 203

F

False Eastings 533
False Northings 533

FeatureCollection 192

FeatureGeometry 191

FeatureLayer
  defined 559

FeatureOverrideStyleModifier
  customizing 459
  overriding styles 243, 284

features
  defined 559
  described 191
  ExecuteFeatureCollection 190
  FeatureCollection 192
  FeatureGeometry 191
  mapping term description 59
  modifiers themes 262
  modifying 192
  retrieving from table 192
  searching 193
  searching for 194
  Selection class 156

FeatureStyleModifier
  changing layer appearance 459
  customizing 459
  modifying style 242
  UML diagram 243
Index

file formats
  grid images 313
  raster images 309
fill patterns 490
fill styles 490
  drawing regions 280
  in remote spatial tables 226
Find class
  requirements 249
  returned objects 250, 253
finding map features
  incorrect address elements 259–260
  place names 246
  refining boundary table 248
  street addresses 246–247
  street intersections 246
font point styles
  drawing points 280
font styles
  drawing text 280

G
GDI+ translucency and anti-aliasing 406, 408, 478
.gen files
  supported raster format 309
geocode
  defined 559
GeocodeClientFactory class 319
geocoded addresses
  CandidateAddress class 320
GeocodeMatchCode class 320
GeocodeRequest class 322
  sending geocoding requests 319
GeocodeResponse class
  candidate addresses 319
geocoding
  adding geocoding client to applications 318
  address candidates 319
  address dictionaries 322
  close matches 328
  constraint preferences 326
  constraints 325
  defined 318
  displaying data on maps 62
  example 323
  input addresses 321
  overview 62, 320, 322
  postal codes 325
  reference addresses 321
  result codes 328, 330–331
  sending requests 319
  street addresses 323
  street intersections 324
geocoding classes 318
Geocoding namespace
  overview 67
Geocoding World component 322
GeocodingConstraints class 320
GeoDictionary 184–185
  defined 560
  utility 418–419, 421
GeoDictionaryManager 421
  defined 560
Geographic Information System (GIS)
  defined 560
Geography Markup Language (GML)
  defined 560
geometric centroid
  defined 560
generators
  coordinate systems 297–299, 301–302
Geometry class 288–289
geometry column
  mappable tables 217
Geometry Markup Language (GML) 368
Geometry namespace
  overview 66
  UML diagram 289
geometry objects
  converting to FeatureGeometry objects 293
GeoTIFF files
  supported raster format 309
GIF files
  supported raster format 309
glossary of terms 558–563
GML
  See Geometry Markup Language (GML)
GmlFeatureCollection class 463
  customizing 463
graduated symbol themes
  creating 263–264
gricule
  defined 560
gricules
  enabling in Workspace Manager 414–415
  layer settings 404
.grc files
  supported raster format 309
.grd files
  supported grid format 313
  supported raster format 309
Grid
  supported table type 169
Index

grid images 313  
    adding to a map 314  
    hillshade parameters 281  
    inflection 281  
    main classes 314  
    retrieving data from 314  
    styles 280  

GTOPO30 files  
    supported grid format 313

H

hillshading  
    grid parameter 281

Hotine Mercator projection  
    Oblique Azimuth 532

HTTP Session  
    defined 96

Hypertext Markup Language (HTML)  
    defined 560

Hypertext Transfer Protocol (HTTP)  
    defined 560

I

IIS  
    application pools 120  
    defined 560  
    server settings 120–121  
    tuning your application 120–121

IIS 7.0  
    developing web applications with 37, 45  
    installation requirements 37

importing  
    features from XML files 463

individual value label themes  
    creating 270

individual value themes  
    creating 269

inflection  
    grid images 281

InfoTips 143

InProc application  
    development model 101–102  
    versus pooling 101–102

InProc development model  
    defined 96  
    versus pooling 100, 103  
    web applications 100, 103

installation  
    prerequisites 36  
    procedure 39  
    system requirements 28  
    types of 30

installed bitmap images  
    accessing 282

installers  
    deploying applications 49, 53

interior styles  
    BaselInterior class 281  
    drawing regions 280  
    SimpleInterior 280

InteriorStyleRepository class  
    customizing 462

Internet Information Services (IIS)  
    defined 560

I SearchResultProcessor interface 459

I Session interface  
    defined 560  
    session management 152  
    UML diagram 152

isoChrone  
    drive-time 343

isoChronePreferences class 343

isoDistance  
    drive-distance 345

isoDistancePreferences class 346

isogramPreferences class 343

J

JavaScript  
    web tools 81–82, 86

JPEG files  
    supported raster format 309

JPEG2000 files  
    supported raster format 309

K

Key  
    MI_Key column 165

keywords  
    Mapinfow.abb file substitutions 256

L

label themes  
    creating 269–270

Label tool  
    desktop 140  
    removing manually added labels 402  
    repositioning curved labels 413

LabelLayer 237

LabelModifier 238

LabelProperties 238

labels  
    curved 411, 413–414  
    individual value theme 270
mapping term description 59
overview 237
priority expressions 403–404
ranged theme 269
styles 402
UML diagram 237
Workspace Manager 401–403
LabelSource 237
latitude
defined 560
layer
defined 561
layer control
Workspace Manager 394, 396–397, 399, 401
layer override styles
composite styles 280
layer types
FeatureLayer 233
GroupLayer 234
MapLayer 234
ObjectThemeLayer 234
UserDrawLayer 234
LayerControl 78
creating custom tabs 460
customizing 460
layer nodes 461
styles 283
Windows control 130–131
LayerNodeHelper classes
customizing 461
layers
applying translucent effects 409
customizing appearance 459
mapping term description 59
types 233
UML diagram 233
UserDrawLayer 234
LegendControl 79
legends 272–273
adornments 240
aligning frames 274
cartographic 273
mapping term description 59
licenses
file locations 33
obtaining 32
types of 31–33
licensing
applications 31–32
known issues 36
troubleshooting 34–36
line styles 504
BaseLineStyle class 281
drawing polylines 281
in remote spatial tables 226
line tools 142
LineStrings
creating 291
LineStyleRepository class
customizing 463
loading data
MapLoader 232
longitude
defined 561
Longitude/Latitude
defined 561
M
M category matches
geocoding result codes 330
Map Application template
rapid prototyping 124
Map class
map contents 231
map legends 272–273
aligning frames 274
Map object
determining coordinate system 299
map title
adornments 240
map tools
architecture 81
JavaScript-enabled 81
process diagram 83
using 83
map view
View tools 140
MapAlias property
web controls 84
MapControl 78
defined 561
in table cells 93
planning a desktop application 128
web control 75
Windows control 127, 232
MapControlModel
web applications 82
MapExport
exporting to an image 230
MapFactory
creating maps 231
MapForm applications
using MapForm template 42–45
Windows controls 47
MapInfo 3.0 Compatible fonts
   drawing points 282
   VectorSymbolRepository class 282
MapInfo ADO.NET Data Provider
   data access 187
   data provider 177
   MICommand 188–189
   MIConnection 187
MapInfo Codespace
   defined 561
MapInfo codespace definitions 512–516
MapInfo Grid files
   supported grid format 313
MapInfo Map Catalog
   adding rows 223–226
   creating 222
MapInfo SQL language 191
   MIDataReader 190
   reference 199
MapInfo Workspace format (.MWS) 386
MapInfo.Engine.Session.State
   defined 97
MapInfoCoordinateSystemSet.xml file
   coordinate units 531
   projection 520
   projection ID numbers 519
MapInfo.abb file
   editing 255–256, 259
   matching to street abbreviations 246–247
   substitution keywords 256
   substitution methods 256–259
MapLoader
   loading data 232
MapMarker server
   result codes 320
mappable tables
   adding rows to Map Catalog 223–225
   defining with server table queries 216–217
   making tables 185
Mapping namespace
   overview 66, 230
   UML diagram 230
mapping terms and concepts 58–61
Mapping.Legends namespace
   overview 66
Mapping.Thematics namespace
   overview 66
   UML diagram 262–263
MapPrinting
   class 480
maps
   cartographic legends for 273
   mapping terms and concepts 58–61
MapStyleControl 479
MapToolBar
   Windows control 128, 130
MapTools
   for desktop applications 137
MapView
   list of views 232
MapX 5.0
   migrating to MapXtreme 423–433
MapXtreme
   architecture overview 64
   creating an application 435
   feature overview 20
   installing 39
   migrating to 22
   object model 65–67, 424
   product overview 20
   support resources 25
   upgrading 39, 42
web controls architecture 80
MapXtreme 2005/2008 web controls
   migrating from MapXtreme 2004 91
MapXtreme ASP.NET Web applications
   building without templates 48
   using templates 45
MapXtreme Session
   defined 96
MapXtreme Session instance
   clean 113
   description 114
MapXtreme Windows Map applications
   using templates 42
matches
   geocoding 328
matching data
   automatching 184–185
   GeoDictionary 184–185
matrix routing 337
MatrixRouteRequest class 338
MatrixRoutingPreferences class 339
memory footprint
   non-pooled web application 118
   pooled web application 119
MemTable
   supported table type 168
merge modules
   creating an installer 50, 53
meridian
   defined 561
metadata
   TAB files 172
   TableInfo 172
MI Command  
analyzing data 198
MI_Geometry column 165
MI_Key column 165
MI_Style column 165
miAware 3.0 Location Utility  
result codes 320
MICommand 189  
data access 187
SQL statements 188
MIConnection  
data access 187
MapInfo ADO.NET data provider 187
Microsoft Data Access Components (MDAC)  
deploying applications 54
MIDataReader 190
migrating to MapXtreme 22, 423–433
modifier styles  
composite styles 280
modifier themes  
derivation 262
dot density theme 271
individual value label theme 270
individual value theme 269
ranged 267
ranged label theme 269
modifying features 192
MrSID files  
supported raster format 309
MS Access  
supported table type 167
MSI installers  
creating an installer 50, 53
MultiPoint objects  
creating 290
multipoint routing 336
.mws  
MapInfo Workspace format 386
MXTRunNCP.exe  
See runtime installer

N

NAD 83 datum 533
NADCON  
converting between NAD 27 and NAD 83 533–534
named connections  
types 388
workspace persistence 468
namespaces 65–67  
defined 561
National Transformation v. 2 (NTv2) 302–305

.NET object model  
COM interoperability 124
NITF  
supported raster format 309
non-earth map  
defined 561
non-match codes  
geocoding result codes 331
non-pooled web application  
memory footprint 118
North American Datum Conversion (NADCON)  
See NADCON

O

object model  
arquitectura 65
data access 177, 183, 185, 198–200
implementation 424
MapXtreme versus MapX 424
namespaces 65–67
object themes  
bar 266
description 263
graduated symbol 263–264
pie 265
Oblique Azimuth  
Hotine Oblique Mercator projection 532
ODBC connection string format 214–215
OGC query interface  
analyzing data 198
OpenGIS Web Feature Service Implementation 368
OpenGIS Web Map Service Implementation Specification 354
opening tables  
Catalog 166
operating systems  
supported 28
Oracle applications  
troubleshooting 227
Oracle Spatial  
connection string format 216
gometry conversion 211–213
overriding styles 284

P

packaging  
a desktop application 446
a web application 452–453
packing tables 170
pan tool  
desktop 140
Index

parameters  518
  projections  518
pattern scaling  483
PCX files  supported raster format  309
pen styles  504
  in remote spatial tables  226
performance  application  118–121
  optimizing data access  200
per-record styles  225
persistence  defined  561
  process description  155
  saving information  463
Persistence namespace  overview  66
Photoshop files  supported raster format  309
place names  246
  Find result codes  254
planning  data access  185, 198, 200
planning an application  desktop  124–125, 137
  web  74–75, 77, 99, 102–104, 106
PNG files  supported raster format  309
point styles  504
  BasePointStyle class  281
  in remote spatial tables  226
  using bitmap images as  282
PointExclude class  341
PointPercentageSpeedUpdate class  350
PointRef schema  185–186
  defined  561
PointRelativeSpeedUpdate class  350
points  creating  290
PointSpeedUpdate class  349
point-to-point routing  335
Polyconic projection  533
polygons  creating  294
polyline tools  142
polylines  drawing styles  281
pool size  configuring  105, 114–115
pooled application  development model  103
pooled development model  defined  96
  saving state  104
pooled web application  CPU usage  119
  memory footprint  119
pooling  advantages  104
  defined  103
  development model  104
  versus InProc  100, 102–104
pooling, defined  562
postal centroid matches  geocoding result codes  330
postal code geocoding  325
pre-defined styles  282
printing from MapXtreme applications  476–488
product architecture  64
product features  highlights  20
project templates  shipped with MapXtreme  125
projections  datums  522, 524–530
  defined  562
  MapInfoCoordinateSystemSet.xml  520
  MapInfoCoordinateSystemSet.xml ID numbers  519
  mapping term description  61
  parameters  518
  publications and resources  539
prototyping  rapid desktop application development  124
Q
queries  customizing with QueryDefinition class  458
QueryDefinition  customizing  459
QueryFilters  customizing  458
R
radius tool  desktop  140
range  in projections  533
ranged label themes  269
  creating  269
ranged themes  creating  267
  distribution types  267–268
Index

Raster
  supported table type 168
raster handlers 311
raster images
  adding to a map 310
  coordinate systems 310
  limitations 310
  main classes 309
  printing translucent 398
  styles 280
  supported file formats 309
Raster namespace
  overview 67
  UML diagram 308
raster reprojection 310, 400–401
RDBMS Server
  supported table type 168
rectangle tool
  desktop 140
rectangle tools
  drawing rectangles 143
redirecting assemblies 41–42
refining boundary table
  Find result codes 253
  finding street addresses 248
relating tables 183, 185
relief shading
  See hillshading
remote spatial tables
  adding rows to Map Catalog 223–226
  importing to a DBMS database 222
  MapInfo Map Catalog 221–222
  per-record styles 225
  specifying styles 226
  styles 226
remote tables
  accessing attribute data 218
  cache management 219–221
  DBMS servers 210
  defining mappable tables 216
  mappable tables 217
  mapping to X/Y columns 211
  ODBC connection string format 214–215
  performance 219
  with a .tab files 210
  without a .tab file 210
repository classes 462
request/response cycle
  web tool 83
result codes 254
  classes 320
  geocoding 328, 330–331
result properties
  Find matches 252–253
result sets 183
  select statement 183
ResultSet
  supported table type 169
retrieving features 192
rings
  creating 294
RoadTypePercentageSpeedUpdate class 351
RoadTypeRelativeSpeedUpdate class 351
RoadTypeSpeedUpdate class 351
route geometry 340
RouteInstructionsRequest class 339
RouteRequest class 336–337
RouteSegmentDataRequest class 347
routing
  adding routing client to applications 334
  avoiding points, features, segments 340
  defined 334
  driving directions 339
  matrix 337
  multipoint 336
  overview 62
  point-to-point 335
  preferences 338
  returning segment information 347
  start and end times 341
  stop times 342
  time-based 341
  transient updates 348
routing classes 334
Routing namespace
  overview 68, 334
  UML diagram 335
RoutingPreferences class 339
runtime installations 30
runtime installer
  deploying applications 49
runtime licenses 31–32
S
S category matches
  geocoding result codes 329
sample applications
  AJAXDemo 89–90
  search 195
  shipped with MapXtreme 125
sample connection strings 216
sample data copyright information 547–549, 551–553
scale factor
- Transverse Mercator projection 532
scalebar
- adornments 240
scaling patterns 483
SDK installations 30
SDK licenses 31
Seamless
- supported table type 169
seamless raster layers 396
Search functions
- customizing 458
search sample application 195
searching for features 193
- Catalog methods 193
- Searchlnfo 194
- SearchInfoFactory 194
SearchResultProcessor class
- customizing 459
segment information
- routing 347
SegmentExclude class 341
SegmentPercentageSpeedUpdate class 349
SegmentRelativeSpeedUpdate class 349
SegmentRoadTypeUpdate class 348
SegmentSpeedUpdate class 348
select statement
- MICommand 188–189
- result sets 183
select tools
- selecting points and regions 140
selecting objects
- outside buffers 204
- within buffers 204
Selection class 156
serialization
- ADO.NET 179
- defined 562
- ManualSerializer 111–112
- process description 154
- restoring state 112
- saving state 111–112
server table queries
- defining mappable tables 216–217
server tables
- caching data 219
session instance
- pooled 103
session management 152
- ASP.NET 152
- ISession interface 152
sessionState
- defined 97
ShapeFile
- supported table type 168
.sid files
- supported raster format 309
single close address matches
- geocoding result codes 329
SmoothingMode property 478
source code
- MapXtreme web controls 88–89
source rows 184
spatial schemas
- defined 562
- PointRef 185–186
- XY spatial 185
SpatialWare applications
- troubleshooting 227
special characters
- Mapinfo.abb file substitutions 259
Spot files
- supported raster format 309
SQL language
- MapInfo reference 199
SQL statements
- MICommand 188–189
SRID codes
- registering with MapXtreme 299–302
standard parallels
- conic projections 532
state management
- automatic 104
- configuring manual 107
- defined 96, 98, 562
- manual 104, 110–111
- manual example 106–107
- options 98
- pooled applications 104
- session object 152
- terms 96–97
- thematic sample 108–113
- web application planning 96
- web applications 77
- web controls 85
StateManager class
- defined 97
- implementing 110–111
stock dialogs
- provided with MapXtreme 132
stock styles
- default styles 281
stock tools
- desktop 144
- web 81
- web tools 76
Index

<table>
<thead>
<tr>
<th>street abbreviations</th>
<th>matching to with Find 246–247</th>
</tr>
</thead>
<tbody>
<tr>
<td>street address geocoding</td>
<td>323</td>
</tr>
</tbody>
</table>
| street addresses | Find result codes 253  
matching to with Find 246 |
| street intersection geocoding | 324 |
| street intersections | matching to with Find 246 |
| street names | matching to with Find 246 |
| StyleRepository | style storage 282 |
| StyleRepository classes | accessing predefined styles 282 |
| styles | area 279  
composite 280  
creating custom bitmaps 283  
customizing with repository classes 462  
fill 280–281  
font 280  
font point 280  
grid 280  
grid hillshading  
in remote spatial tables 226  
infection 281  
label 402  
LayerControl 283  
line 281  
lookup tables 489  
modifying 242  
overriding 243, 284, 396  
points 279, 281  
raster 280  
stock 281  
supported custom bitmap symbols 508  
supported fill patterns 490  
supported line styles 504  
supported vector symbols 504  
text 282  
vector points 282  
Workspace Manager 402 |
| Styles namespace | overview 67  
UML diagram 278 |
| substitutions | Mapinfow.abb file keywords 256 |
| supported file formats | grid images 313  
raster images 309 |
| symbol font sets | 504 |
| symbol styles | in remote spatial tables 226 |
| system requirements | 28 |
| T | .TAB  
supported table type 167 |
| TableInfo | metadata 172 |
| tables | adding expression columns 175  
alias 163  
Catalog 166  
closing 170  
columns 163–164  
creating permanent native 173  
creating temporary MemTable 175  
creating temporary native 174  
defined 562  
events 171  
mappable 185–186  
mapping term description 58  
MI_Geometry column 165  
MI_Key column 165  
MI_Style column 165  
non-mappable 163, 166, 178, 185–186, 561–563  
opening 166  
packing 170  
relating 183, 185  
result sets 183  
supported types 167  
views 183 |
| Targa files | supported raster format 309 |
| technical support, contacting | 25 |
| templates | ASP.NET 45, 47  
for MapXtreme projects 125  
MapForm 42–45 |
| terminology | pooling 96–97  
state management 96–97  
web application planning 96–97 |
| text | drawing 280  
drawing styles 282 |
| thematic sample | initial request 112–113 |
| themes | applying translucent effects 411  
bar 266 |
bivariate 271–272
CreateThemeWizard 133–136
dot density theme 271
graduated symbol 263–264
individual value label theme 270
individual value theme 269
legends 272–273
mapping term description 60
modifier 262
object 263
pie 265
ranged 267
ranged label theme 269
recalculating bins 135
Workspace Manager 391, 401
TIFF files
 supported raster format 309
Time data type
 SQL expressions 204
tools
customizing 144, 462
designing custom 141
desktop namespace 67
for desktop applications 144
for web applications 76
tool events 145
web architecture 81
Workspace Manager 392–393
Tools namespace
 overview 67
transient updates
 routing 348
translucency
 applying to map features 408–411
enabling in Workspace Manager 398, 406, 408
MapStyleControl 479
when printing 478
Transverse Mercator projection
 scale factor 532
TravelTime class 341
trial licenses 32
troubleshooting
 licensing 34–36
Oracle applications 227
SpatialWare applications 227
TrueType fonts
 StyleRepository class 282
tutorial
 creating a desktop application 436–438, 442–444
packaging a desktop application 446
packaging a web application 452–453
tutorial applications 42
U
UML diagram
 Data namespace 162
Data.Find 249
Geometry 289
Mapping 230
Mapping.FeatureStyleModifier 243
Mapping.Labels 237
Mapping.Layers 233
Mapping.Thematics.IModifierTheme 262
Mapping.Thematics.ObjectTheme 263
Raster 308
Routing 335
Styles 278
Unicode Transformation Format (UTF-8)
defined 563
 supported in workspace 469
Uniform Resource Locator (URL)
defined 562
Universal Character Set (UCS)
defined 563
updating existing applications 41
upgrading to MapXtreme 39, 42
URL request
 client side 82–83
 MapControlModel 82
 server-side 82–83
user state
defined 97
managing 107
user-created installers
deploying applications 50, 53
user-defined metadata
 542–545
UserDrawLayer class 460
USGS DEM files
 supported grid format 313
utilities
 GeoDictionary Manager 184–185, 418–422
 Workspace Manager 386
V
VectorPointStyleRepository class
customizing 463
Vertical Mapper Classified Grid
 supported raster format 309
Vertical Mapper Continuous Grid
 supported grid format 313
 supported raster format 309, 313
ViaPoint class 342
View
 supported table type 169
view tables 183
Index

view tools
  changing map view 140

Views
  MapView 232

Vista
  See Windows Vista

Visual Basic .NET
  ASP.NET application template 45, 47
dev development framework 28
  Map application template 42
  MapForm application template 43–44

Visual C#

Visual Studio
  creating applications in 43–45, 47
development framework 28

W

watermarks
  with runtime licenses 32
  with SDK licenses 31
  with trial licenses 32

web application
  adding ASP.NET AJAX controls 89–90
  architecture 69–70
  background map 75
  client side command execution 82
  client side map interaction 81
  deploying 454
design considerations 74
design time view 76
initial request 113
MapXtreme Session instance 113–114
packaging 452–453
packaging tutorial 452–453
planning 74–75, 77, 96, 98–104, 106
pooling 77
run-time view 76
server-side command architecture 82
state management 77
terminology 96–97
tools 76
updating to MapXtreme 40
web controls 77–78, 80, 84–85, 91, 93
  architecture 80
  ASP.NET AJAX 89
  ASP.NET Web applications 47
defined 563
description 78–80
directory structure 78
  distributing 88–89
error handling 85
event handling 85
  in frames 93
  localizing 94
  MapAlias property 84
  migrating from MapXtreme 2004 91
  modifying source code 88–89
  state management 85
tools 76
  using 78, 84–85, 91, 93

Web Feature Service (WFS)
  client 379, 381–382
  configuring server 373–376
  creating map layers 382
defined 563
requests to server 368–372

Web Map Service (WMS)
  configuring server 356–361
defined 563
  layer configuration on server 362–364
  map requests via client 364–365
  requests to server 354–355
test case 119

Web server
  defined 563

web tools
  architecture 81
customizing 86
description 78–80
interacting with MapControl 81
javascript-enables 78
using 83
using with MapControl 93
versus full page postback 78

web.config file
  preloaded workspace 84
  session management 109, 152
  settings 84

WebControls namespace
  overview 67, 77

WFS
  See Web Feature Service (WFS)

where clause 203

Windows 2000
  supported operating system 28

Windows Bitmap files
  supported raster format 309

Windows controls
  desktop applications 126
  LayerControl 130–131
  MapControl 127
  MapForm applications 47
  MapToolBar 128, 130
Index

Windows dialog boxes
  adding to applications 132
  CreateThemeWizard 133–136
  customizing 137

Windows enhanced metafiles
  supported raster format 309

Windows metafiles
  supported raster format 309

Windows Server 2003
  supported operating system 28

Windows Vista
  installation requirements 37
  with IIS 7.0 45

Windows XP
  supported operating system 28

Windows.Controls namespace
  overview 67

Windows.Dialogs namespace
  overview 67

.wmf files
  supported raster format 309

WMS
  See Web Map Service (WMS)
  supported table type 169

Workspace Manager
  feature summary 386
  Label tool 402, 413
  labels 402
  layer control 394, 396–397, 399, 401
  layer control tools 394, 397
  menu commands 387–389, 391, 393
  named connections 388
  overriding styles 396
  save as XML 386
  themes 401
  tools menu commands 392

WorkspaceLoader class
  customizing 463

workspaces
  creating programmatically 474
  defined 563
  loading 105, 114–115
  pre-loading 75
  reprojected images 310
  structure 469–470, 472–473
  XML format 387

X

XML files
  GeoDictionary 418–419, 421
  importing features from 463
  Workspace format (.MWS) 386

XY schema 185
  defined 563

Z

Z category matches
  geocoding result codes 330

zoom tool
  desktop 140

zoom-in tool
  example 83