

Migrating to MapXtreme 2004

The following table lists the objects in the MapX 5.0 object model and the equivalent in the current object model for MapXtreme 2004. Note that with the re-architecture of such an extensive product as MapXtreme 2004, the equivalent may approximate.

MapX 5.0	MapXtreme 2004
<p>AffineTransform</p> <p>AffineTransform is a creatable object. It can be changed after creation using the Set method.</p>	<p>AffineTransform class (MapInfo.Geometry namespace)</p> <p>Use the CreateAffineTransform method of the CoordSysFactory. The resulting AffineTransform can then be passed as an argument to the CoordSysFactory.CreateCoordSys method to define a rotated or skewed coordinate system. AffineTransform is immutable.</p>
<p>AllFeaturesConstraint</p> <p>In MapX 5.0 users can control which features in a remote database layer are cached by creating constraint objects from a layer - using the CreateAllFeaturesConstraint, CreateBoundsConstraint, and CreateFeaturesConstraint methods of the Layer Object.</p> <p>In MapX 5.0 you set the Cache parameter of a LayerInfo object (of type miLayerInfoTypeServer) to "ON" or "USER", then add the layer to the map, then call the Create*Constraint methods on that layer to determine which features are held in cache</p>	<p>AllFeaturesConstraint class (MapInfo.Data namespace)</p> <p>MapXtreme 2004 allows the user to control which features in a table are cached by defining constraint objects as well. However, the layer-based data access model from MapX has been replaced with the table-based data access model (see Layers).</p> <p>Instead of defining cache options for a layer you will specify those options on a table.</p> <p>You will set the CacheType property in the TableInfo.CacheSettings object (MapInfo.Data namespace) to CacheOption enumeration value that corresponds to "ON" or "USER". Then add the table via Catalog.OpenTable, then pass the table returned from the call to OpenTable to the AllFeaturesConstraint, BoundsConstraint or FeaturesConstraint constructor.</p>

MapX 5.0	MapXtreme 2004
<p>Annotation</p> <p>In MapX 5.0 the Annotation object and Annotations collection are used to draw text and symbols on a map. Annotations are drawn on top of the map, are not associated with any map layer and have no attribute data associated with them. As such they are not available to the mapping operations one would perform with Point or Text features (searching, combining, data aggregation, buffering). Likewise they do not persist beyond runtime, i.e., they are not written to geosets nor are they written to disc like features in a map layer. Annotations do have fixed locations on the earth, controlled by each Annotation's Graphic object.</p>	<p>In MapXtreme 2004 there are no Annotations. Like annotations, adornments (legends, titles, scale-bars) are drawn on top of the map and are not associated with any layer or table. Unlike annotations, adornments do not have a fixed location on the earth - they do not scroll with the map but rather stay in a fixed position in the map window. The functionality of Annotations can be mimicked in MapXtreme 2004 by creating a temporary table, typically a MemTable. The easiest way to do this is to use the MapInfo.Data.TableInfoFactory class to create a TableInfo and pass this to Catalog.CreateTable to add the temporary table to the map. You can create Geometries of type Point and LegacyText and add them to the table via the Table.InsertFeature method or by using the map tools (see MapTools and AddMapTool classes in the online help).</p>
<p>Annotations</p>	<p>See Annotation.</p>
<p>BindLayer</p> <p>In MapX 5.0 a BindLayer object is used as a parameter to Datasets.Add to specify information on how to perform a data bind. There are three types of data binds that one can perform, corresponding to the DatasetType constants (Normal, XY and PointRef).</p>	<p>In MapXtreme 2004 each of these types of data binding has been handled differently. There is no BindLayer object in MapXtreme 2004, but the underlying functionality is still there. See the Datasets entry in this table for details on how each of the three types of data binding is carried out in MapXtreme 2004.</p>
<p>BitmapSymbol</p>	<p>BitmapPointStyle class (MapInfo.Styles namespace)</p>
<p>BitmapSymbols</p>	<p>BitmapPointStyleRepository class (MapInfo.Styles namespace)</p>
<p>BoundsConstraint</p>	<p>BoundsConstraint class (MapInfo.Data namespace)</p> <p>See AllFeaturesConstraint.</p>
<p>CoordSys</p> <p>CoordSys is a creatable object. It can be changed after creation using the Set method</p>	<p>CoordSys class (MapInfo.Geometry namespace)</p> <p>You must create a CoordSysFactory and use its CreateCoordSys methods. CoordSys is immutable.</p>

MapX 5.0	MapXtreme 2004
<p>Dataset</p> <p>In MapX 5.0 there are three types of data binds one can perform; Normal, XY and PointRef. A normal data bind in MapX 5.0 is the process of binding data from an external source (e.g. an Access database) to a layer in the map. Essentially this means adding columns to a map layer, the row values in these columns come from the external data.</p> <p>A PointRef data bind is the process of creating a new point layer (physical or in memory) from external data. The external data must have some geographic information (such as county or postal code) that can be matched against a layer registered in the geodictionary. For each row in the external data that can be matched to a corresponding region in the reference layer, a point is created at the centroid of that region (but in a new layer).</p> <p>An XY data bind is the process of creating a new point layer (physical or in memory) from external data. The external data must have X and Y (Longitude and Latitude) columns. Each row in the external data with valid X and Y values becomes a point in the newly created layer.</p>	<p>There is no Dataset class in MapXtreme 2004. The same functionality exists but it has been more logically dispersed around the object model.</p> <p>In MapXtreme 2004 the same thing can be achieved via the Table.AddColumn methods (MapInfo.Data namespace).</p> <p>In MapXtreme 2004 the same result can be achieved by creating and defining an instance of the SpatialSchemaPointRef class (MapInfo.Data namespace) and setting the SpatialSchema property of a TableInfo object (MapInfo.Data namespace) to that instance. See the SpatialSchemaPointRef class entry in the programmer's reference (online help) for code samples.</p> <p>In MapXtreme 2004 the same result can be achieved by creating and defining an instance of the SpatialSchemaXY class (MapInfo.Data namespace) and setting the SpatialSchema property of a TableInfo object (MapInfo.Data namespace) to that instance. See the SpatialSchemaXY class entry in the programmer's reference for code samples.</p> <p>Note: the COM dataset interface defined in Appendix F of the MapX 5.0 Developer's Guide is no longer supported. Use the existing supported data sources or the new ADO.NET-based table types</p>
<p>Datasets</p>	<p>Obsolete. See Dataset.</p>
<p>Datum</p> <p>Datum is a creatable object and can be changed after creation using the Set method.</p>	<p>Datum class (MapInfo.Geometry namespace)</p> <p>You must create a CoordSysFactory and use its CreateDatum method. The resulting Datum can then be passed as an argument to the CoordSysFactory.CreateCoordSys method to define a mathematical description of the earth's shape and orientation. A Datum is immutable.</p>
<p>Feature</p> <p>In MapX 5.0 the feature types available are region, line, symbol, text, multipoint, collection. All of these still exist in MapXtreme 2004, but the names are different.</p>	<p>FeatureGeometry class (MapInfo.Geometry namespace)</p> <p>The concept of a feature has not changed from MapX 5.0 to MapXtreme 2004. Features are still essentially rows in tables, geographies and the attribute data associated with them, that represent physical (geographic) entities.</p> <p>The Geometry types available in MapXtreme 2004 are Point, MultiPoint, MultiCurve, MultiPolygon, FeatureGeometryCollection, Rectangle, RoundedRectangle, Ellipse, LegacyArc and LegacyText. Each is its own class, derived from obstacle class FeatureGeometry</p>
<p>Features</p>	<p>FeatureCollection class (MapInfo.Data namespace)</p> <p>See Feature.</p>
<p>FeaturesConstraint</p>	<p>See AllFeaturesConstraint</p>

MapX 5.0	MapXtreme 2004
<p>FeatureFactory</p> <p>In MapX 5.0 the methods of the FeatureFactory object are used to create new geographic features where none existed before or create new geographic features by performing operations on existing features.</p> <p>To create new features from scratch you use the CreateArc, CreateCircularRegion, CreateCollectionFeature, CreateEllipticalRegion, CreateLine, CreateMultipoint, CreateRegion, CreateSymbol and CreateText methods of the FeatureFactory object, each of which return a Feature object. You would then call the Layer.AddFeature method to add the newly created feature to your map.</p> <p>You also use methods of the FeatureFactory object to create new features from existing features.</p>	<p>In MapXtreme 2004 there is no FeatureFactory class per se, but you can still create new features and perform the same set of operations on existing features.</p> <p>In MapXtreme 2004 each type of geographic object is its own class inherited from abstract class FeatureGeometry (MapInfo.Geometry namespace). Use each classes' constructors to create new instances of each class e.g. a new Point or Multi-Curve. To add this new Geometry to a table you create a new Feature (see Feature constructors in online help) and set Feature.Geometry to your newly created Geometry.</p> <p>In MapXtreme 2004 you will use the members of the FeatureProcessor class (MapInfo.FeatureProcessor namespace) to perform these operations. The MapX 5.0 FeatureFactory.BufferFeatures, CombineFeatures, EraseFeature, IntersectFeatures, IntersectionPoints methods are replaced by the FeatureProcessor.Buffer, Combine, ConvexHull and Intersect methods. All of these methods are overloaded to account for the many types of buffers, combines, convex hull and intersects you may wish to perform. Note that the names of the Geometry types available in MapXtreme 2004 are different than what you're used to from MapX. See the GeometryType Enumeration in the online help. See also Feature.</p>
<p>Field</p>	<p>Column class (MapInfo.Data namespace) See Fields.</p>

MapX 5.0	MapXtreme 2004
<p>Fields</p> <p>In MapX 5.0 a Fields collection has three primary uses.</p> <ul style="list-style-type: none"> Used to define the columnar structure of a new layer. In this case the fields collection is a parameter of a LayerInfo object which is passed as an argument to the Layers.Add method. Used as an optional argument in a call to Datasets.Add to specify which fields from the source data to include in the dataset i.e. to which columns in the source data to bind to a map layer. Used as an optional argument to Themes.Add to specify which columns from an existing dataset to base a theme on. <p>In each of these MapX operations you are using a Fields collection to specify which fields from a data table will be involved in the current operation, be it the creation of a new layer, a dataset or a theme.</p>	<p>Columns class (MapInfo.Data namespace)</p> <p>In MapXtreme 2004 a table is described or defined via the TableInfo class (MapInfo.Data namespace). The columnar structure of a Table is maintained in a Columns collection and accessed via the TableInfo.Columns property.</p> <p>In MapXtreme 2004 each of the three operations described for MapX is fundamentally different.</p> <ul style="list-style-type: none"> To specify the columnar structure of a new table you will create a TableInfo object (MapInfo.Data namespace) and call its Columns.Add method one or more times before calling Catalog.CreateTable and passing that TableInfo object. Datasets have been replaced with the Table.AddColumns method. See Datasets in this table and the Table.AddColumns entry in the online help. Columns on which themes are based are now specified as expressions in the theme constructor. See online help for RangedTheme, IndividualValueTheme, DotDensityTheme, PieTheme, BarTheme, and GraduatedSymbolTheme classes (MapInfo.Mapping.Thematics namespace).
Find	MapInfo.Data.Find namespace
FindFeature	FindResult class
FindMatch	FindCloseMatch, FindAddressRange classes
FindMatches	FindCloseMatchEnumerator, FindAddressRangeEnumerator classes
FindResult	FindResult class
Geoset	Geosets have been replaced with a new MapInfo Workspace format (.mws). MapXtreme 2004 can still read your old .gst files, but it cannot create new ones. See The Developer Guide appendix entitled "Understanding the New MapInfo Workspace".
Geosets	See Geoset .
Graphic	Obsolete. There are no Annotations in MapXtreme 2004. See Annotations .
IndividualValueCategory	IndividualValueTheme class
IndividualValueCategories	IndividualValueTheme class

MapX 5.0	MapXtreme 2004
<p>Label</p> <p>In MapX 5.0 map labeling is controlled by three objects; Label, Labels and LabelProperties. LabelProperties hangs off of the Layer object - i.e., each layer has its own LabelProperties object. The properties therein are used to control label display (zoom range, style, position, etc.) for all features in a layer - i.e., the properties apply to all of the labels in a layer.</p> <p>The Label object and Labels collection are added to MapX to allow more control over labeling. The Labels collection also hangs off of the Layer object, but it affords you more control than the LabelProperties object in that you can control the display of each label in a layer independently of the others.</p> <p>MapX 5.0 also has label thematics - the ability to specify label display based on data values from the layer being labeled.</p>	<p>Label, Labels, LabelProperties, LabelLayer, LabelSource, LabelModifier, OverrideLabelModifier classes (MapInfo.Mapping namespace)</p> <p>In MapXtreme 2004 labeling has changed significantly, providing the programmer with more control over label display than ever before. Labels live in their own LabelLayer (derived from abstract class MapLayer) and they are independent of the FeatureLayer they annotate. This means you can place the LabelLayer into a map in any position you want to control when labels are drawn relative to other layers in the map.</p> <p>Given a LabelLayer you need to specify an Instance of the LabelSource class to associate a table with your LabelLayer. You will then define an expression based on the columns from that table that will determine the text of the labels.</p> <p>For example, you have a feature layer that contains points that represent store locations. The same data table that contains those points also contains attribute data associated with those points: mailing address, managers' names, sales from last quarter and Boolean columns indicating whether or not that location has an ATM machine or a pharmacy. The expression could indicate that you wish to label each point with the manager's name and "ATM" or "Rx" where appropriate. This information is controlled via an instance of the LabelProperties class which you get from the LabelSource.DefaultLabelProperties property.Y</p> <p>You can modify the settings specified in the LabelProperties object by defining one or more LabelModifiers. When drawing labels MapXtreme first looks to the DefaultLabelProperties to determine how to draw the labels for a LabelLayer. It then looks for LabelModifiers and applies them where and how you specify. Note that LabelModifier is actually an abstract class; see the online help topics for LabelModifier and OverrideLabelModifier for implementation details.</p> <p>MapXtreme 2004 also has label thematics, they are implemented as a type of LabelModifier. See these entries in the online help, and see the Theme and Themes.</p>
Labels	See Label .
LabelProperties	See Label .

MapX 5.0	MapXtreme 2004
<p>Layer</p>	<p>IMapLayer interface; FeatureLayer class, ObjectThemeLayer class, LabelLayer class, GroupLayer class (MapInfo.Mapping namespace)</p> <p>In MapX 5.0, Layer objects carried a number of properties that have been moved elsewhere in MapXtreme 2004.</p> <p>Two particularly important changes are related to MapX's Layer.Editable and Layer.Selectable, which no longer exist in MapXtreme 2004. The reason for this is that abilities to edit and select are now better handled as properties of the Tools collection, or of particular tools: it is useful to be able to create custom tools that can select or edit specific layers on a map. Each tool is allowed to have its own set of LayerFilters to specify its particular area of influence. The MapTools collection (MapInfo.Tools namespace, accessed through MapControl.Tools) provides the functionality needed to set up LayerFilters for editability and selectability, either for individual tools or for groups of tools.</p> <p>For example, defaults for all Add tools can be set up through a MapControl's Tools.AddMapToolProperties object. Other tool-specific settings, e.g., Snap mode and Infotips, are also available through MapTools.</p>
<p>Layers</p> <p>In MapX 5.0, labels for feature layers are treated as properties of feature layer, not as independent layers in their own right. Similarly, themes are not classified as layers; instead, they are properties of a dataset associated with a layer.</p>	<p>Layers class (MapInfo.Mapping namespace)</p> <p>In MapXtreme 2004, there is a new type of layer called a LabelLayer, which can be displayed even if no associated feature layer is visible. Some types of themes are also given their own layers in MapXtreme 2004 - namely object themes (pie-chart and bar-chart themes), which are treated as separate feature layers. See Themes.</p> <p>The new GroupLayer class allows several "nested" layers to be treated as a single layer - common settings can be applied to all of them at once. The standard enumerator for the Layers collection only enumerates the top-level layers, not any layers contained inside GroupLayers. However, the MapLayerEnumerator returned by GetMapLayerEnumerator can enumerate all nested layers depending on the options specified</p>

MapX 5.0	MapXtreme 2004
<p>LayerInfo</p> <p>A LayerInfo object is used to hold information about a layer to be added to the map via a call to the Layers.Add method.</p>	<p>TableInfo Class (MapInfo.Data namespace).</p> <p>In MapXtreme 2004 a table-centric data access model has replaced the MapX layer-centric model. With that change the LayerInfo object has gone away. Its functional replacement is the TableInfo class and the many implementation classes derived from it: TableInfoNative, TableInfoBase, TableInfoMSAccess, TableInfoAscii, TableInfoServer, TableInfoSeamless, TableInfoRaster, TableInfoGrid, TableInfoWMS, TableInfoShapefile, TableInfoMemTable, TableInfoView, TableInfoResultSet, TableInfoAdoNet. See also the MapTableLoader, MapGeosetLoader, MapWorkspaceLoader in the online help, all of which are derived from the MapLoader abstract class.</p>
<p>Legend</p> <p>In MapX 5.0, there is a Theme.Legend object for every Theme object (though it may or may not be visible).</p>	<p>Legend Class (MapInfo.Mapping.Legends namespace)</p> <p>In MapXtreme 2004, no Legend object exists for a theme until it is explicitly created. Legend objects are generic -- legend frames for any type of theme may be created with the LegendFrameFactory and added to a Legend. Cartographic and custom legend frames may also be created and added.</p> <p>When the required legend frames have been added to a Legend object, the Legend itself is generally appended to a map's Adornments collection so that it appears as part of the map.</p> <p>Alternatively, the legend can be converted to an image via a LegendExport object, and the result can be displayed in a separate control or window.</p>
<p>LegendText</p>	<p>LegendFrameRows class (MapInfo.Mapping.Legends namespace)</p>
<p>LegendTexts</p>	<p>ILegendRow interface, CartographicLegendFrameRow class, ThemeLegendFrameRow class, CustomLegendFrameRow class, AllOthersLegendFrameRow class (MapInfo.Mapping.Legends namespace)</p>

MapX 5.0	MapXtreme 2004
Map	<p>Map Class (MapInfo.Mapping namespace) and MapControl Class (MapInfo.Web.UI.WebControls namespace)</p> <p>The MapX 5.0 Map object has been separated into two entities; Map and Map Control.</p> <p>The MapControl can be embedded in a WinForm and many of its properties can be specified at design time. It contains the MapTools collection and it has a number of events associated with it.</p> <p>Each MapControl has exactly one Map. The Map contains the Layers and Adornments collections.</p> <p>Many of the objects that hang off of the map in MapX 5.0 have been moved to other areas of the object model. Some of the major differences between the MapX 5.0 map object and the MapXtreme 2004 Map class are highlighted below.</p> <p>Map.Dataset, Datasets, DatasetGeoField and DatasetTheme have gone away. See Dataset section above.</p> <p>Map.DefaultStyle has gone away. There is no default style associated with the entire map. Styles are applied as needed throughout the object model - to features, text, legends, layouts, themes, overrides and selections. See Style section below.</p> <p>Map.FeatureFactory has gone away. See FeatureFactory section above.</p> <p>Map.Geoset, Geosets have gone away. See Geosets section above.</p> <p>Map.InfoTip and InfoTipPopupDelay have gone away. InfoTips are now controlled via the MapTools class.</p> <p>Map.Title and TitleText have gone away. Titles are now Adornments.</p> <p>Note that the Map does contain the Adornments collection now.</p>
MultivarCategory	MultiVariableTheme, MultiVariableThemeCategory Classes (MapInfo.Mapping.Thematics namespace)
MultivarCategories	MultiVariableTheme, MultiVariableThemeCategory Classes (MapInfo.Mapping.Thematics namespace)
NotesQueryInfo	Obsolete.
NotesViewInfo	Obsolete.
OCIQueryInfo	Obsolete.
ODBCQueryInfo	Obsolete.
Parts	<p>The MapXtreme 2004 Geometry model (MapInfo.Geometry namespace) exposes a richer model for feature creation and editing. This replaces the MapX 5.0 Parts and Points objects.</p>

MapX 5.0	MapXtreme 2004
Point	DPoint class (MapInfo.Geometry namespace)
Points	Obsolete. See Parts .
RangeCategory	RangedTheme, RangedThemeBin, RangedThemeBins, ModifierThemeBin classes (MapInfo.Mapping.Thematics namespace)
RangeCategories	RangedTheme, RangedThemeBin, RangedThemeBins, ModifierThemeBin classes (MapInfo.Mapping.Thematics namespace)
Rectangle	DRect structure (MapInfo.Geometry namespace)
ResolveObject	MatchResolver class (MapInfo.Data.Geodictionary namespace). The MatchResolver class provided a new way to interact with the automatching process. It replaces the MapX 5.0 ResolveObject, ResolveObjects collection and ResolveDataBindEx event.
ResolveObjects	See ResolveObject .
RowValue In MapX 5.0 the RowValue object and RowValues collection are used to get and set attribute data in a Layer. A RowValue represents a single cell (value) and a RowValues collection represents a single row. One would get attribute data values by obtaining a RowValues collection from a Dataset and set attribute data values via the Feature.Update method, passing in the optional RowValues argument.	In MapXtreme 2004 there are no Datasets (see Dataset above) and hence no RowValues. You can get and set attribute data associated with Features directly via the Item property of the Features class (MapInfo.Data namespace). You can also use the methods of the MICommand class (MapInfo.Data namespace) to get and set attribute data values. The ExecuteReader and ExecuteScrollableReader methods will create instances of MIDataReader and MIScrollableReader which will in turn allow you to iterate through rows of a table and column values in those rows. The MICommand.ExecuteNonQuery method will allow you to perform inserts, updates and deletes on entire rows or selected columns within rows.
RowValues	See RowValue .
Selection	Selection class (MapInfo.Engine namespace)
SourceRow	SourceRow Class (MapInfo.Data namespace)
SourceRows	SourceRows Class (MapInfo.Data namespace) The SourceRows class in MapXtreme 2004 performs the same function as the SourceRows collection in MapX 5.0. It identifies the rows from the source data that are bound to a row in a table via a call to Table.AddColumn (the MapXtreme equivalent of adding a dataset).

MapX 5.0	MapXtreme 2004
<p>State</p> <p>The State object is used to save and restore MapX objects. This is particularly useful in MapXtreme applications where an instance of a map object (MapXServer) is potentially shared among users of the application. A user accesses your application and views a map, then performs some operation on that map such as zooming or panning. The map object is then returned to a pool of shared instances. There is no guarantee that the same user will get the same map instance back the next time they perform a map operation so it is important to restore the map to its previous state before performing the next map operation.</p>	<p>MapInfo.Persistence namespace</p> <p>The classes in the MapInfo.Persistence namespace are used to save and load map settings from workspace documents.</p>
<p>Style</p> <p>A Style object contains attributes for drawing symbols, lines, regions, and text. The object contains attributes for all feature types, even though a particular feature type only uses a subset of the properties</p>	<p>Style class (MapInfo.Styles namespace).</p> <p>The abstract Style class is the base class for all MapInfo Styles. It has several subclasses: AreaStyle, BaseLineStyle, TextStyle, BasePointStyle, RasterStyle, and GridStyle. There is also a CompositeStyle which may contain one of each of the above styles.</p> <p>You cannot instantiate the abstract Style classes (Style, BaseLineStyle, BaseInterior, or BasePointStyle); you must create a particular type like SimpleLineStyle, or create a combination of several style types as a CompositeStyle. Thus, a MapXtreme 2004 CompositeStyle object is the nearest equivalent to a MapX Style object.</p> <p>For example, developers 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.</p>
<p>Theme</p>	<p>ITheme interface; IModifierTheme interface, FeatureStyleModifier class, ObjectTheme class, ObjectThemeLayer class (MapInfo.Mapping.The-matics namespace)</p>

MapX 5.0	MapXtreme 2004
<p>Themes</p> <p>In MapX 5.0, several disparate types of themes are collected together in the Themes collection.</p>	<p>MapInfo.Mapping.Thematics namespace - see Theme.</p> <p>In MapXtreme 2004, there is no single collection containing all of the themes on a map.</p> <p>Object themes (pie and bar charts and graduated-symbol themes) are separate layers.</p> <p>Themes that override the style of objects in an existing layer (ranged, individual-value, and dot-density themes) are now classified as special types of FeatureStyleModifiers.</p> <p>Similarly, Label themes (ranged or individual-value) are special types of LabelModifiers.</p>
<p>ThemeProperties</p>	<p>properties of BarTheme class, PieTheme class, GraduatedSymbolTheme class, DotDensityTheme class, RangedTheme class, IndividualValueTheme class, RangedLabelTheme class, IndividualValueLabelTheme class(MapInfo.Mapping.Thematics namespace).</p>
<p>Title</p>	<p>TitleAdornment Class (MapInfo.Mapping namespace).</p>
<p>Variable</p>	<p>See Variables.</p>
<p>Variables</p> <p>In MapX 5.0, Variable objects are used to associate or bind a value into an expression that is passed to Layer.Search.</p>	<p>MICommand class (MapInfo.Data namespace)</p> <p>In MapXtreme 2004, the MICommand object (MapInfo.Data namespace) is used to execute Select, Insert, Update, and Delete statements. These statements may refer to bound values by name just as the expressions passed into Layer.Search did. The values are captured as instances of MIParameter and are managed in an MIParameterCollection which is accessible through the MICommand.Parameter property. Variable names in MapXtreme 2004 should be preceded by an "@" sign or ":" for clarity and good practice. Thus, whereas in MapX you would perform a Layer.Search with an expression of the form "Obj within var1", in MapXtreme 2004 you would have a query of the form "Select * From States Where Obj CentroidWithin @var1".</p>