



**NetQuarry, Inc.**

**Training**

**300 - Metadata Advanced**

## Field References

In the NetQuarry studio, you can refer to the values of fields by means of field references.

The basic syntax for a reference is to enclose a fields key\_name inside square brackets.

E.g.

```
[first_name]
```

This means get the value of the field “first\_name” from the mappers current row.

There are modification characters that can be added to field references to change the format of the returned value

```
$ = Use DisplayText  
' = AnsiQuote  
` (back tick) = Double quote  
# = Use zero if null or empty  
< = Use old value if dirty  
& = Use label  
? = Escape for use in URL  
@ = Escape for HTML
```

The character modified can't be used in combination.

The most common reference modifier is the ' to AnsiQuote the returned value.

E.g.,

```
hello Mr Neill -> 'hello Mr Neill'
```

```
hello Mr O'Neill -> 'hello Mr O''Neill'
```

\$ - Would be used if the field had a picklist and you wanted to get the resolved display text for the id

' - Would be used when you want to use a field reference in a SQL statement, WHERE clause (typically in filters)

` - Would be used mainly for displaying information to a user

? - would be used when the field value is a URL that could have characters not supported in a URL

< - would be used when the field value contains an HTML fragment that needs to be displayed with the HTML tags, rather than as HTML.

## Embedded Functions

Embedded functions are like variables that you can refer to in meta data. The idea is that at run time the embedded function gets replaced by a string it represents. The main uses for embedded functions are for setting default values in fields and configuring filters.

The syntax for an embedded function is

```
!fnName[$]([param1 | ,param2 | ,...)
```

The embedded function always begins with the characters “!fn”, followed by the name of the function in proper case.

There is an optional modifier parameter “\$” which means resolve the embedded function and AnsiQuote the result.

Some embedded functions take parameters but that depends on the embedded function itself.

Embedded functions are created in four ways.

- Dynamically created due to meta data attribute in Session properties (and code generator)
- Programmatically created in code (typically at session startup)
- Dedicated functionality in the core to handle special case embedded function
- Dedicated functionality in custom handler for special case embedded functions

## How to add an embedded function

### Session Property

The easiest way to add an embedded function is to add a session property. In the studio, go to Session Properties and add your property to the list. Choose the data type for your property. For embedded functions they are always returned as string values so you’d want to choose a data type that can be converted to a string in a meaningful way. For the most part, however, you want to choose the session property as a string data type.

When adding a session property, you want to set the attributes Dynamic Only and Session Persist. Additionally you want to set the GenEmbeddedFunc attribute.

Re-generate your generated code for session. This will create declarations for the new session property and also the hooks for the auto creation of the embedded function.

In the Session initialization code (normally) you want to assign a value to the session property.

```
session.PropertywithEmbeddedFunc = some_value;
```

This would create an embedded function that you would refer to like

!fnPropertwithEmbeddedFunc()

### Manually in Code

If you want to add your own embedded function you can register it in the system. The ideal place for that registration is in the Session creation code

```
// IAppContext cxt

string val = "Some Value";

cxt.RegisterEmbeddedFunction("EmbeddedFuncName", val);
```

This would create an embedded function that you would refer to like

!fnEmbeddedFuncName()

### Custom Handler

You can add custom handlers for embedded functions by creating a component that is derived from the class NetQuarry.IFunctionParser

In your IFunctionParser object define values for the properties...

**Name** - for debugging purposes

**FunctionList** - Defines what embedded functions are handled/registered by this custom handler. The function names are returned as semi-colon separated list. The names must not include a "fn" prefix. Simple substitution resolutions can be specified in the form name=value (instead of just a name), e.g. "X=y". Functions so specified will never result in a callback, but instead will be resolved using a simple substitution of "y" for "!fnX()". When the name is provided it will result in a callback into your handler.

And then implement handlers for the following methods...

**Description** – Given a function name you return an appropriate descriptive string.

**Resolve** – The main function for handling the callback. You are given the name of the function that is being handled as well a list of parameters in the embedded function. Therefore you can create embedded functions that take parameters.

Your component may contain code like this

```
public class MyEmbeddedFunctions : NetQuarry.IFunctionParser
{
    string Name { get { return ("My Custom Embedded Functions"); } }
    string FunctionList { get { return ("CreditScore;MinPayment=15"); } } //--- register two functions. One
simple
    string Description(string functionName, bool summary)
    {
        switch (functionName)
        {
            case "CreditScore":
                return (summary ? "The credit score" : "Return the credit score. Provide the salary and credit limit
```

```

as parameters. !fnCreditScore(salary,limit)");
    break;
    case "MinPayment":
        return (summary ? "The minimum required payment" : "Return the minimum required payment currently
default is 15. !fnMinPayment()");
        break;
    default:
        break;
}
}

string Resolve(string functionName, IDatabase db, ResolveOptions options, params object[] args)
    // Resolve an expression
{
    switch (functionName)
    {
        case "CreditScore":
            int creditScore = 0;
            //--- verify you have at least two parameters
            if (args.length < 2)
                throw new ArgumentException("Arguments must be the salary and the credit limit");

            //... your code to calculate credit score given salary and credit limit

            //--- Take into consideration which database to connect to and whether to encode returned value
            //public enum ResolveOptions
            //{
            //    DbmsIndependent = 1,
            //    UseDataDB = 2,
            //    UseRepository = 4,
            //    Raw = 8,
            //    EncodeForUrl = 16, e.g., EAPEncode.ForUrl(creditScore);
            //}

            return (EAPUtil.ToString(creditScore));
            break;
        default:
            break;
    }
}

object Exec(int command, int flags, params object[] args)
{
    //--- future functionality
}
}

```

Then add your component as a FunctionParser in the studio, under components.

## Picklists

Picklists are used as the data sources for drop down lists and to resolve the id's of field values into the equivalent descriptive value the id represents.

There are five modes of picklists and three ways of defining a picklist. The following table shows a matrix of picklist modes and methods

<b>Mode</b>	<b>Definition</b>	<b>SQL</b>	<b>Standard</b>	<b>Enum</b>
Simple		X	X	
Limit To List		X	X	X
Discriminated		X	X	
Display Disabled		X		
Show Groupings		X	X	

## Picklist Definitions

There are three ways of defining a picklist in NetQuarry Studio

### SQL Picklists

As the name suggests, SQL picklists are constructed from select statements. The construction of the select statement follows the syntax described above for simple, limit to list and discriminated types. The SQL for the picklist can be as complex as you wish, but must only select up to three columns. If your SQL selects more than three columns, those additional columns will be ignored.

Any ordering of your picklist must be performed via the SQL statement itself using the standard ORDER BY clause.

The SQL picklist may use any of the datasources defined in the Datasources metadata. In addition to the list from the Datasources metadata, you can also select the Repository (the metadata database) as the source for the picklist. You would only choose the Repository datasource in rare circumstances, or when you may want to add functionality referring to system metadata. The most common source for SQL picklists is the data in your operational database.

### Standard Picklists

A standard picklist is a picklist in which the data for the picklist is stored in metadata and is associated with a module like any other type of metadata.

When you create a standard picklist, the following information should be specified.

Column	Description
Name	The name of the picklist item. The name is used in the code generation to provide a name for the enumeration. The caption/display value for the picklist is auto generated from the name when you enter a value in the name column
Sort Order	An integer value that denotes the order in which the picklist items are displayed.
Alternate Key Text	An alternate string value that may be used as the key of the picklist instead of the (hidden) auto generated picklist item key.
Alternate Key Int	An alternate integer value that may be used as the key of the picklist instead of the (hidden) auto generated picklist item key.
Discrim	The value of the discriminator for the pick list item. By default this is blank, but if you specify a discrim value, the picklist isn't configured to use it unless you set the "HasDiscrim" property for the picklist
Attributes	Attributes for the picklist item.

## Enum Picklist

An enum picklist gives you the ability to create a picklist from the values of a .Net Enumeration. To set an enum picklist, simply give the picklist a name and then specify the fully qualified (namespace and all) name for the enumeration you want as a picklist

## SQL Picklists or Standard Picklists?

When creating a picklist you are faced with the choice of creating a table in your operational database to store the picklist information and creating a SQL picklist, or specifying the same information in metadata as a Standard picklist.

If you prefer the SQL picklist route for operational data, you would have to create a mapper and additional UI in an administration section of your web application to manage the data. However, you will have the ability to create views where you can use the picklist table to resolve the id values to names.

Also, because you'll have to create a mapper to manage the values of your SQL picklist, it supports the idea that if the picklist data was cached, and you have added, modified or removed an item in the SQL picklist, the old picklist data is automatically flushed from the cache and re-loaded with new data next time the pick list is required. Therefore the picklist is never stale.

If you prefer the standard picklist, you don't have to create any mappers and pages in the web ui for managing the data. The Studio provides the management tool

The standard picklists are also included in the Code Generator. The picklists are converted to enumerations in the Picklists.cs generated file. Therefore you can refer to the picklist in your code in a type safe way and set the values of fields that use standard picklists using the generated typesafe enumerations.

For standard picklists, there is no way to automatically refresh the application cache if the standard picklists are modified in the studio.

## Picklist Attributes

The following table describes the attributes that can be selected for a picklist. Some attributes are only appropriate for certain types of picklist.

Attribute	SQL	Standard	Enum
AllowUserAdd	✓		
AllowUserDelete	✓		
AllowUserUpdate	✓		
Cache	✓	✓	
Disabled	✓	✓	✓
GroupByFirstChar	✓	✓	
GroupByTextPrefix	✓	✓	
HasDiscrim		✓	
HideUnknownItems	✓	✓	
IgnoreWhitespace	✓	✓	
KeySameAsText		✓	
LimitToList	✓	✓	
MarkUnknownItems	✓	✓	
NoNullEntry	✓		
SortByKey		✓	
SortByText		✓	
SortDesc		✓	
StoreAltInt		✓	
StoreAltText		✓	
StoreItemName		✓	

## Picklist Modes

As indicated earlier, there are five modes of picklist. Not all picklist definitions support all the picklist modes.

### Simple

A simple picklist consists of a single column of data. The value stored in the database is the same as the value displayed to the user.

<b>SQL</b>	SELECT name FROM view WHERE display=1 ORDER BY name
<b>Standard</b>	<p>Set the KeySameAsText attribute if the picklist is only text</p> <p>For numeric picklist you would have the text identical to the number of the Alternate Key Int field and mark the picklist attribute to StoreAltInt.</p> <p>For numeric picklists of contiguous numbers, the best approach is to set the Min/Max field property on a Combo cell type, and set the NumericDropdown CellTypeAttribute on the field.</p>

For a standard picklist to operate in the this mode

### Limit to List

A limit to list picklist consists of two columns selected from a table or view. The first column selected always acts as the key and is the value stored in the database. The second column selected is always the display value. The value displayed to the user and the value selected by the user. This is the most common type of picklist

<b>SQL</b>	SELECT id, name FROM view WHERE display=1 ORDER BY name
<b>Standard</b>	<p>Set the LimitToList picklist attribute.</p> <p>You have to decide which field is acting as the key value and set the appropriate picklist attribute.</p> <p>No attribute (the internal guid associated with the picklist item)</p> <p>KeySameAsText (the text value is the key – same as simple picklist)</p> <p>StoreAltInt (integer key)</p> <p>StoreAltText (none guid string key)</p> <p>StoreItemName (none guid string key)</p>

## Discriminated

A discriminated picklist is like a limit to list picklist, but three columns are selected. Column one is still the key, column two is still the display value, and column three is a discriminator column. The discriminator column allows the picklist to be filtered in memory, based on the value of the discriminated column.

Let's say the discriminated picklist contains the following data

id	name	company_id
1	Fran	Acme
2	Ian	Acme
3	Nathan	Acme
4	Cam	NetQuarry
5	John	NetQuarry
6	Ryan	NetQuarry

When the picklist is displayed to the user, it expects some value to be passed to the picklist on which to discriminate. If there is no discriminator value (Discrim), the picklist will actually be empty. The picklist will not display all the records.

If the Discrim value is Acme, the picklist will only display the options for Fran, Ian, Nathan.

SQL	SELECT id, name, discrim_value FROM view WHERE display=1 ORDER BY name
<b>Standard</b>	Set the LimitToList picklist attribute. You have to decide which field is acting as the key value and set the appropriate picklist attribute. No attribute (the internal guid associated with the picklist item) KeySameAsText (the text value is the key – same as simple picklist) StoreAltInt (integer key) StoreAltText (none guid string key) StoreItemName (none guid string key)

### Showing Disabled Items

For example, one scenario is that you have a set of items in a picklist but some of them are disabled and you don't want users to use those picklist items. However, you still would like to be able to see the original value of the item in the picklist. Normally for picklists, if an item is excluded from the picklist, either the picklist shows a blank value or it shows the value for the key.

To set up a picklist (SQL only) where an item may be displayed even if disabled, yet cannot be selected is to select 4 columns in the picklist definition.

The first three columns are the key, display and discrim. The fourth column is a Boolean field that denotes whether the picklist item should be selectable. If the value associated with an item is true, then the item can be selected from the list.

<b>SQL</b>	SELECT id, name, discrim_value, display FROM view ORDER BY name Note this example where display field is no longer part of where clause. We want the picklist to be able to display/resolve hidden items, but not be selectable Example where want display feature, but not discrim SELECT id, name, NULL, display FROM view ORDER BY name
<b>Standard</b>	Not possible

### Grouping Items

You can specify a SQL picklist to be grouped by the first character or a set of characters (the grouping is on characters in display text enclosed by []). This grouping occurs by default if the either of the GroupBy attributes is specified for the picklist.

When using a SQL picklist you can specify a fifth column to select in the SQL statement. This column will act as the field on which grouping is performed and not the display text field.

<b>SQL</b>	SELECT id, name, discrim_value, display, group_value FROM view ORDER BY name Example where want group feature and disabled feature, but not discrim SELECT id, name, NULL, display, group_value FROM view ORDER BY name Grouping for SQL picklists automatically groups on the group value.  An alternative to providing a dedicated group column would be to perform the grouping on the second column and use the GroupBy... picklist attributes  SELECT id, name, discrim_value FROM view ORDER BY name And set one of the picklist attributes GroupByFirstChar, GroupByTextPrefix depending on the grouping requirements.
<b>Standard</b>	Set the GroupByFirstChar picklist attribute and the picklist will have a single letter grouping. Set the GroupByTextPrefix picklist attribute and the picklist will be grouped by any of the characters between the [] in the display text. Note that all the text inside the brackets will not appear in the select list, just the grouping.

## Late Bound Picklists

Although picklists are useful in that they provide ways to resolve key values to display text, they do have the downside in that they can take a long time to load if the number of items is large. This problem can be reduced if the picklist is cached. The large list is only ever loaded into memory once and subsequent requests, are filled from the cache. The performance benefit now gained is now potentially lost by a huge increase in the amount of memory required to store large picklists.

If you take a discriminated picklist of say 5,000 rows that has three rows, of about 100 bytes per row. That makes the picklist use up 500kB of memory. The picklist itself has some overhead so the size of the picklist in memory is effectively doubled, to 1MB. If there are 250 sessions currently in use, each with a copy of this picklist, then that means there is 250MB of memory being consumed by one cached picklist.

There have been a number core modifications to solve the memory usage problem that are not directly configurable to the developer, other than to turn on the special low memory usage mode. However, Late Bound picklists is one area where a developer has more control.

A late bound picklist is a picklist where only the items required to be displayed by the picklist are loaded into memory and cached. Typically these picklists are discriminated type picklist where we only select enough items to satisfy any criteria. A stricter type of late bound picklist can be used where items are added to the picklist one item at time, when needed.

### Latebound Picklist using Discrim Filter

To set up a Latebound Picklist using a discrim filter, you set up the picklist like any other discriminated picklist, by specifying the SQL with three columns. For example...

```
SELECT building_id, building, company FROM companies_buildings WHERE display=1
```

To make this picklist latebound, you specify a second (but similar SQL) statement in the LateBoundSQL property, as follows

```
SELECT building_id, building, company FROM companies_buildings WHERE display=1 AND company_id={{DISCRIM}}
```

When the Picklist is first populated, the base SQL statement is used to define the internal structure of the picklist object (how many columns, etc) but the base SQL is modified by adding a TOP 0 clause which means no records are returned, except for the columns themselves.

Later, when the picklist needs to resolve a value, or display the picklist values, the discrim value is taken from the mapper (from the Discrim) property and the latebound SQL is executed to display only what is needed.

### Latebound Picklist using Key Filter

To set up a Latebound Picklist using a key filter, you set up the picklist like any other discriminated picklist, by specifying the SQL with three columns. For example...

```
SELECT people_id, display_name FROM people
```

To make this picklist latebound, you specify a second (but similar SQL) statement in the `LateBoundSQL` property, as follows

```
SELECT people_id, display_name FROM people WHERE people_id={{KEY}}
```

When the Picklist is first populated, the base SQL statement is used to define the internal structure of the picklist object (how many columns, etc) but the base SQL is modified by adding a `TOP 0` clause which means no records are returned, except for the columns themselves.

Later, when the picklist needs to resolve a value, the underlying key value is taken from the mapper and a lookup for that item is made and cached. With a key filter based picklist, you cannot use it for a picklist that is required to be used for selections. Only for resolving id's to display values.

## Templates

Templates can be specified in what we call high and low resolution formats. A given template can have one form of each type that are defined by template properties

The high resolution template can be specified as a fragment of HTML text in the Source property of a template

The high resolution source may also be specified as an htm file in the FileName property which is defined as the relative path (relative to the web root) to the htm file itself.

The Source property is used if the FileName property is not defined, or if the htm file pointed to by the FileName property cannot be found.

The low resolution template is used for sending SMS messages to users and is specified by the SourceLow property. These are simple formatted text strings (not html)

Templates are used for the following purposes

- Email Notification
- Invoicing Templates (through conversion to PDF)
- Hover Summaries
- MiniDetail layouts

Here is a simple example of how we use a template to transform a mapper record into a VCard file for importing into Outlook Contacts (this example is extracted from core functionality)

The template is called “VCARD”

The Source property is defined as a text string in this format...

```
BEGIN:VCARD
VERSION:2.1
N:{{last_name}};{{first_name}}
FN:{{file_as_name}}
ADR;WORK;POSTAL:;{{address}};{{city}};{{state}};{{postal_code}};{{country}}
TEL;WORK:{{phone_number}}
TEL;FAX:{{fax_number}}
TEL;CELL;MSG:{{cell_number}}
EMAIL;PREF;INTERNET:{{email_address}}
TITLE:{{job_title}}
ORG:{{dealership_nm}}
URL;WORK:[{{detail_url}}]
UID:{{dealer_id}}
END:VCARD
```

The locations where we want our data to be inserted are enclosed in double braces e.g. {{dealership\_name}}, or square brackets e.g. [{{detail\_url}}]. The name of the field in the double braces corresponds to a name of a name/value pair from a NameValueCollection object.

A field name in braces {{ }} will have its value escaped for html (effectively using the System.Web.HttpUtility.HtmlEncode method ) (This behavior can be overridden entirely in the Replace method by setting the NoHtmlEscape TemplateReplaceFlag)

If you do not want the HTML escape to occur for a specific field, then you would enclose the field in square brackets [[ ]], in this example, [[detail\_url]].

The names are typically the same as the mapper fields because we have a way of generating the collection automatically from the current mapper row. As long as there is a name/value pair in the collection that matches a name in the template, it will be replaced.

## Handling a command that sends resolved template back to user

The basic steps are to get the template object, get a collection of Name/Value pairs and then do the replacement...

```
using System.Collections.Specialized;
using NetQuarry;

Template templ = Application.Templates["VCARD"];
NameValueCollection nv = mapper.Exec(MapperExecCmds.KeyDisplayCollectionGet, 0) as
NameValueCollection;
//--- nv now contains a collection of name value pairs corresponding to the field values in the
current row of the mapper. On this collection you may add more pairs of data if you want
nv["detail_url"] = "http://www.acme.com"
```

To perform the replacement operation, call the Replace Method on the template object.

```
string vcard = templ.Replace(nv, ContentResolution.High)
```

There are various options in the Replace method that indicates which template to use and what behaviour to follow when performing the replace. The help file will go into more detail about when you should use these options.

In this case, we stream the VCard template back to the user so they can add to their Outlook.

This is how we do it in case you might need to do something similar...

```
using System.Web;

HttpContext cxt = HttpContext.Current;
HttpResponse rsp = cxt.Response;

rsp.ClearContent();
rsp.ClearHeaders();
rsp.ContentType = app.Properties.GetStringValue("MimeType", "text/x-vcard");

//--- Note: content-length needs to be the byte count, not the char count so
//--- get the byte count per the content encoding (e.g. UTF8)

rsp.AppendHeader("content-length", rsp.ContentEncoding.GetByteCount(vcard).ToString());
rsp.AppendHeader("content-disposition", "attachment; filename=vcard.vcf");
rsp.Write(vcard);
rsp.End();
```

## Handling a command that sends an email

Sending email is very similar. You define the template layout either as HTML, or plain text and inside the template specify the replacement tokens. As I said, the names you use can match the mapper fields because of the exec we use to generate the NameValueCollection object.

You decide whether you want the template to be defined inside the Source property, or as an htm file in the FileName property.

```
using System.Collections.Specialized;
using NetQuarry;
using NetQuarry.Services;

ServiceInfos svcs = this.Application.Services;
IEmailService mail = (IEmailService)svcs.GetServiceInstance("SmtpMail");

Template templ = Application.Templates["Email Notification"];
NameValueCollection nv = mapper.Exec(MapperExecCmds.KeyDisplayCollectionGet, 0) as NameValueCollection;
string body = templ.Replace(nv, ContentResolution.High);
string subject = "Email Alert";
bool isHTML = true; // set your flag whether you're sending html email or not

string sendTo = GetSendToEmailAddressList();// your function here to get list of email addresses

if (!string.IsNullOrEmpty(sendTo))
{
    mail.Send(string.Empty, sendTo, subject, body, isHTML);
}
```

In the mail.Send method, the first parameter is the senders address. If you leave it blank (like this example), the email will be sent from the email address defined by the DefaultSenderAddress property of the SmtpMail service.

## Full example for VCard

```
using System.Web;
using System.Collections;
using NetQuarry;
using NetQuarry.Data;

public override void MapperCommand(IMapper sender, EAPCommandEventArgs e)
{
    if (e.CommandName == "exportVCard")
    {
        ArrayList lstKeys = null;
        using (IMapper clone = new Mapper())
        {
            lstKeys = sender.Exec(MapperExecCmds.SelectedKeys, 0) as ArrayList;

            ///--- clone without adding flavors so far
            ///--- this is another way to not use the same mapper passed to the handler
            sender.Clone(clone, 0);

            string vCards = ConstructVCard(clone, lstKeys).ToString();

            HttpContext cxt = HttpContext.Current;
            HttpResponse rsp = cxt.Response;

            rsp.ClearContent();
            rsp.ClearHeaders();
            rsp.ContentType = Properties.GetStringValue("MimeType", "text/x-vcard");

            ///--- Note: content-length needs to be the byte count, not the char count so
            ///--- get the byte count per the content encoding (e.g. UTF8). [10/3/06 CW]

            rsp.AppendHeader("content-length", rsp.ContentEncoding.GetByteCount(vCards).ToString());
            rsp.AppendHeader("content-disposition", "attachment; filename=vcard.vcf");

            rsp.Write(vCards);
            rsp.End();
            ///--- remember to close any mapper we open
            clone.Close();
        }
    }
}
```

```

private StringBuilder ConstructVCard(IMapper mapper, ArrayList lstKeys)
{
    StringBuilder sb = new StringBuilder();
    ///--- read from the extension properties which template to use for this command
    ///--- the template is tied to the mapper

    string templName = Properties.GetStringValue("Template");

    if (string.IsNullOrEmpty(templName))
        throw new ArgumentNullException("VCard Template", "No VCard Template associated with the mapper");

    Template vcardTempl = Application.Templates[templName];

    if (vcardTempl == null)
        throw new ArgumentNullException("VCard Template", "No VCard Template associated with the mapper");

    if (mapper.HasRecords)
    {
        ///--- Iterate through the rows...
        mapper.MoveFirst();

        do
        {
            ///--- If we have a set of selected records and this record isn't one of them
            ///--- then skip this record.

            if (lstKeys != null && lstKeys.Count > 0)
            {
                if (!lstKeys.Contains(mapper.RowKey)) continue;
            }

            ///--- resolve the values into the template
            NameValueCollection nv = mapper.Exec(MapperExecCmds.KeyDisplayCollectionGet, 0) as NameValueCollection;

            NameValueCollection nvLink = new NameValueCollection(4);
            nvLink.Add("req", "nav");
            nvLink.Add("mop", mapper.MOP);
            nvLink.Add("flt", EAPEncode.ForUrl(SavedFilter.RegisterReqFilter(mapper.Application, mapper.RowKeyFilter(mapper.RowKey))));
            nv["detail_url"] = EAPUtil.ConstructExternalLink(Application, nvLink);

            ///--- add the vcard record to the string builder
            sb.AppendLine(vcardTempl.Replace(nv, ContentResolution.High, TemplateReplaceFlags.NoHtmlEscape));

        } while (mapper.MoveNext());
    }

    return (sb.ToString());
}

```

## Named Filters

Filtering is an important part of any application. And, because filters are able to be applied to many different objects, it is necessary that some filters should be defined once and then re-used in many places.

This is the basic idea behind named filters.

Named filters are defined in the Studio as follows

Property	Description
Module	The module to associate the named filter to.
Filter Name	The name of the filter. This is the name you will refer to when referencing in metadata or code.
Moniker	A helpful description of the filter.
Filter Sql	The actual filter clause of the filter. The filter can contain references to other named filters and embedded functions.
Priority	The priority of the filter definition. The filters are loaded into a NamedFilter collection on the application and it is possible to define more than one filter with the same name. The filters can be permissioned by role and if a user has more than one role and therefore has more than one named filter of the same name, the filter with the highest priority will be used. Priority of 1 is higher than 2.
Attributes	You can disable the named filter if it's no longer used.
Permissions	Allows you to enable named filters based on role. This gives you the possibility of defining a named filter per role, but with the same name. Where such a filter is used means it can return different data to the user depending on the user's role.

## Referring to Named Filters

Only mappers can directly reference named filters. On the Filter subform of a mapper, you can add a filter (give a name) and then select one of the named filters from the list. If there is a mapper filter with both a regular SQL filter and a named filter, the named filter wins.

Elsewhere you can reference named filters using the embedded function syntax...

```
!fnNamedFilter(<filter_name>)
```

It is possible to refer to a named filter in another named filter, since named filters support the use of embedded functions.

## Flavors

Flavors are normally used to modify the behavior of the UI. Either show or hide a field or lock it depending on where the mapper is currently being displayed and whether any customizations have been applied. Whenever a mapper is displayed on a page, that page assigns a flavor to the mapper. The flavor of the fields in the mapper are tested against the mapper's flavor (bit tested) to see if there is anything special needed to be done.

There are four types of flavor

- Include flavor
- Exclude flavor
- Hide flavor
- Lock flavor

### Include Flavor

If the flavor of the field matches the flavor of the mapper, the field is included in the mapper's field collection. If a field has an include flavor and the flavor of the mapper is zero, the field will not be in the mapper's field collection.

### Exclude Flavor

If the flavor of the field matches the flavor of the mapper, the field is excluded from the mapper's field collection otherwise it is included in the mapper's field collection. If a field has an exclude flavor and the flavor of the mapper is zero, the field will be in the mapper's field collection.

### Hide Flavor

If the hide flavor of the field matches the flavor of the mapper, the field is included in the mapper's field collection but it is hidden from display.

### Lock Flavor

If the lock flavor of the field matches the flavor of the mapper, the field is included in the mapper's field collection but it is visible, but locked from being edited.

Flavors can be applied to a mapper from the following places

- Core code – e.g., List view, Detail view, Wizard, Save, Find, etc
- Pages
- Page elements in wizards
- Page elements in consoles

- Subforms
- Menu targets

For majority of situations, you can get by with flavoring by what's added by the core. For situations where you would like to add your own flavoring, there are eight custom flavors you can manipulate and add to your own objects to change behavior.

## Mutual Exclusion

It's vitally important for flavors (especially include/exclude flavors) that you take into account every possible combination of flavor. An example of using flavors is to have the same field display slightly differently depending whether the record is new, or existing.

An example is a field that selects a document to attach to a record for a new record, but is then not linkable for the existing record. The mapper field might look like this

KeyName	Cell Type	Include Flavor	Exclude Flavor
document_name	FilePath	New	0
document_name	TextBox	0	New

The fields are configured by flavors to be mutually exclusive. On a new record, the flavor "New" is applied to the mapper. Therefore the FilePath field will be included and the textbox field would be excluded. In contrast on an existing row, the New flavor is not set. Because the flavor is not new, the FilePath field is excluded and the TextBox field is now included.

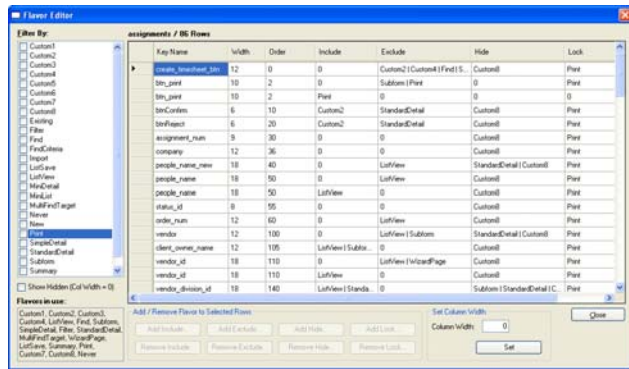
If you get the flavoring wrong and for some combination of flavors both fields are on the mapper then both fields are automatically excluded. It's impossible to decide which of the two fields to show. In this instance there will be a devlog entry indicating there are two fields with the same key name and are being excluded. Of course you can quickly tell there's a problem as the field you want to see is not there, or there's an error in the code that access that field.

## Flavor Editor

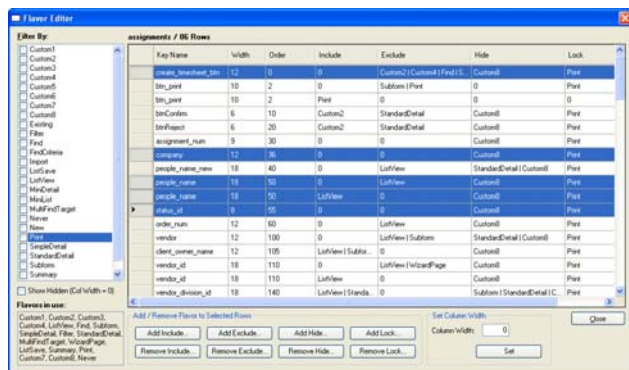
In the studio, if you right click on a mapper row, the last option is to open the flavor editor. This editor allows you test out your flavor scheme on the fields in the mapper. You can apply various flavors and see what the effects of those flavors are on the mapper's fields collection.

The editor also allows you to add and remove flavors from many fields at once.

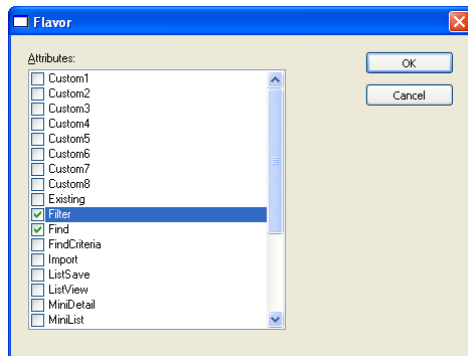
If you are starting with flavors and are still unsure about flavoring in general, you should use the flavor editor to check what you are doing. In any case, if you are changing the flavor of many fields at once, then the Flavor Editor is the recommended approach.



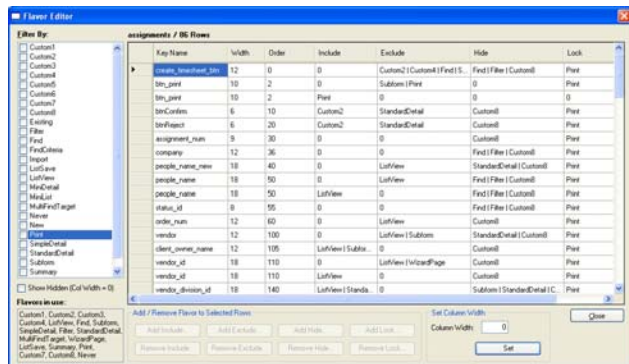
To add or remove any combination of fields, you must first select one or more rows using shift or ctrl and click like a regular multi-select list box. Once selected the 8 flavor editor buttons become enabled.



Click on one of the buttons to pop up the Flavor widget



Select one or more flavors and click OK to apply or remove the flavors.





## Using Wizards

When you create a wizard you must bring to bear all parts of the platform in one location. Concepts such as flavors, field references, navigation parameters, page layout and mappers are heavily used in creating wizards. In addition to these core features, there are some additional concepts that only apply to wizards.

### Instances (Page Element)

On a wizard, an instance is like a category. It allows you to group together one or more pages of the wizard so that the values presented on each page are associated with the same physical instance of a mapper object. The best way to describe how instances work is to provide some examples.

You want a wizard that records a users name, address, phone information. The name and phone information are in a person mapper, the address in a separate address mapper. The wizard requires that you have three pages in the order of, name, then address, then phone info.

Page1 – name uses the person mapper

Page2 – address uses the address mapper

Page3 – phone uses the person mapper

Now, we want Page1 and Page3 to refer to the same person mapper because the name and phone info belong to the same person. Therefore when we save the information on the wizard, the name and phone information are saved on the same person record. Therefore we have to specify that Page1 and Page3 use the same instance of the person mapper.

Page1 – name, (mapper – person), (instance – name\_phone)

Page2 – address, (mapper – address), (instance – address\_only)

Page3 – phone, (mapper – person), (instance – name\_phone)

The names of the instances are arbitrary but should be meaningful for their purpose so that other developers can understand the functionality of the wizard.

At the end of the wizard there will be two inserts. One on the person mapper, one on address mapper.

If you had not specified the same instance for page one and three, for example...

Page1 – name, (mapper – person), (instance – name)

Page2 – address, (mapper – address), (instance – address\_only)

Page3 – phone, (mapper – person), (instance –phone)

There would be three inserts at the end of the wizard. One insert on the name instance of a person mapper, containing just the name information . A second insert for address on the address mapper. A third insert on the phone instance of a person mapper with just phone information. The records from the inserts of each instance of person mapper would not be related to each other.

It is important to remember that instances of the same name **do not** have to be on consecutive pages.

### Primary Instance

There is a special type of instance used in wizards, called the primary instance. The primary instance of a wizard generally is the instance associated with the first page of the wizard. The primary instance of the wizard can be thought of as the instance that performs the main purpose of the wizard. Also, when navigating from the wizard, it is the rowkey of the primary instance that is used by default as a navigation parameter.

You can override which instance is the primary instance by setting the PrimaryInstance property at the page level.

### Types of Page Elements

There are three main types of page element you would use in a wizard. Detail, List and Summary. These are specified by which component is used to render that information.

- WizPhantomDetail – Detail page
- WizPhantomEditList – List page (editable)
- WizPhantomList – List page (uneditable)
- MapperSummary – Summary info

The WizPhantomList page type is probably used the least as it does not provide the ability to edit data and would only be used to display a list of information. And if there is a read only summary, then a MapperSummary page might be used instead.

## WizardPhantomDetail Pages

The most common requirement for wizard page is to use a detail layout to provide the appropriate data entry. To use a detail page, you would create a new page element using the WizPhantomDetail component. When using detail pages on a wizard, you would typically use the following properties.

Property	Description								
FieldList	On detail layouts of wizard pages, you can specify which fields should be displayed on that page without resorting to complex flavoring. The fields you want to show on a page, you specify in a semi colon delimited list in the FieldList property of the page element. If you don't specify any fields, you get all the fields in the mapper.								
GroupList	If your mapper has some groups defined, you may select the fields in that group by the group name. Multiple groups are specified in a semi colon delimited list of group names.								
LockFieldList	If you want to display some fields that are normally editable, but need to be locked for some reason (again without resorting to flavors) then you can provide a semi colon delimited list of fields that should be locked.								
InstanceName	The name of the mapper instance for this page								
CopyValues	<p>The CopyValues property of a WizPhantomDetail page element provides a way in metadata to set values in your pages from data in the query string, from other instances in your wizard, or set literal values. The copy values property is specified as a semi colon delimited list of field values to copy, in the form &lt;dst-instance&gt;.&lt;field-key&gt;=&lt;src-instance&gt;.&lt;field-key&gt;. The copy values parameters will almost always require the use of field references.</p> <p>If your wizard has multiple instances, then you would normally specify the copy values property on the first page of each instance.</p> <p>To provide an example, we'll take the wizard example earlier of the creating a new person, address, phone data. In the navigation to this wizard, we added a custom query string parameter "company_id=some_guid" so we can relate the new person to an existing company.</p> <table border="1"> <thead> <tr> <th>Copy From</th><th>How</th></tr> </thead> <tbody> <tr> <td>Incoming query string parameters</td><td>On the first page element, the CopyValue parameter would be specified as <code>company_id=[req.company_id]</code> note: we don't have to provide the name of the current instance as dest-instance. By default it's the value in the current instance we're setting.</td></tr> <tr> <td>Another wizard page instance</td><td>In the second page of the wizard, we're adding a new address record, but we have to relate the address to a person and also we have to relate the address to a company. We are creating a new person record on the name_phone instance which will have a newly generated people_id and also have the company_id available from the request, or the mapper of the anme_phone instance. So there are two ways to set the people_id and company_id field on the address mapper <code>people_id=[name_phone.people_id];company_id=[req.company_id]</code> or <code>people_id=[name_phone.people_id];company_id=[name_phone.company_id]</code></td></tr> <tr> <td>Literal value</td><td>The copying of a literal value into an instance is rarely used as the data you want to pass around and set is usually dynamic in nature. When you may want to set a literal value is if your wizard simply needs to set a status_id value and all you want to show is a summary page of information to confirm what the user wants to do. <code>status_id=4</code></td></tr> </tbody> </table>	Copy From	How	Incoming query string parameters	On the first page element, the CopyValue parameter would be specified as <code>company_id=[req.company_id]</code> note: we don't have to provide the name of the current instance as dest-instance. By default it's the value in the current instance we're setting.	Another wizard page instance	In the second page of the wizard, we're adding a new address record, but we have to relate the address to a person and also we have to relate the address to a company. We are creating a new person record on the name_phone instance which will have a newly generated people_id and also have the company_id available from the request, or the mapper of the anme_phone instance. So there are two ways to set the people_id and company_id field on the address mapper <code>people_id=[name_phone.people_id];company_id=[req.company_id]</code> or <code>people_id=[name_phone.people_id];company_id=[name_phone.company_id]</code>	Literal value	The copying of a literal value into an instance is rarely used as the data you want to pass around and set is usually dynamic in nature. When you may want to set a literal value is if your wizard simply needs to set a status_id value and all you want to show is a summary page of information to confirm what the user wants to do. <code>status_id=4</code>
Copy From	How								
Incoming query string parameters	On the first page element, the CopyValue parameter would be specified as <code>company_id=[req.company_id]</code> note: we don't have to provide the name of the current instance as dest-instance. By default it's the value in the current instance we're setting.								
Another wizard page instance	In the second page of the wizard, we're adding a new address record, but we have to relate the address to a person and also we have to relate the address to a company. We are creating a new person record on the name_phone instance which will have a newly generated people_id and also have the company_id available from the request, or the mapper of the anme_phone instance. So there are two ways to set the people_id and company_id field on the address mapper <code>people_id=[name_phone.people_id];company_id=[req.company_id]</code> or <code>people_id=[name_phone.people_id];company_id=[name_phone.company_id]</code>								
Literal value	The copying of a literal value into an instance is rarely used as the data you want to pass around and set is usually dynamic in nature. When you may want to set a literal value is if your wizard simply needs to set a status_id value and all you want to show is a summary page of information to confirm what the user wants to do. <code>status_id=4</code>								

Property	Description
WizardPageAttributes	You would use this property to specify certain behaviors on this page. The majority of the attributes for this property refer to the visibility of the default summary information. A full explanation of summaries and summary options are given below.
Mapper	The name of the mapper for this page element. If no mapper is specified at the page element level, the page will use the mapper associated with the page.
Caption	A caption to display on the wizard in the top right hand corner of the wizard page describing the purpose of that step to the user.

## WizardPhantomEditList Pages

If you wanted to use an editable list as a wizard page, then you would create a new page element using the WizPhantomEditList component.

Property	Description												
InstanceName	The name of the mapper instance for this page												
WizardListOptions	<p>You use these options to specify whether the editable list requires the following features.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>AllowAddNew</td><td>The editable list supports addition of new rows via a popup dialog window. When this attribute is set, you need to additionally specify the NewTarget page element property.</td></tr> <tr> <td>AllowAddViaFind</td><td>The editable list supports addition of new rows via a popup multifind dialog. When this attribute is set, you need to additionally specify the FindFilter, ParentMapper and ViewKey properties.</td></tr> <tr> <td>AllowRemoveItems</td><td>Allow items to be deleted from the editable list. Items which already exist in the list when the page is opened, or items added to the list once the page is opened can be deleted from the list.</td></tr> <tr> <td>NoRefresh</td><td>If items can be added to the list, then a refresh button is automatically added to the page. You can hide the refresh button if not necessary.</td></tr> <tr> <td>ShowLinksAsButtons</td><td>When items can be added and we have links New..., New From Find... or Refresh, by default these are rendered as links. You set this attribute and you can have these links rendered as buttons.</td></tr> </table>	Attribute	Description	AllowAddNew	The editable list supports addition of new rows via a popup dialog window. When this attribute is set, you need to additionally specify the NewTarget page element property.	AllowAddViaFind	The editable list supports addition of new rows via a popup multifind dialog. When this attribute is set, you need to additionally specify the FindFilter, ParentMapper and ViewKey properties.	AllowRemoveItems	Allow items to be deleted from the editable list. Items which already exist in the list when the page is opened, or items added to the list once the page is opened can be deleted from the list.	NoRefresh	If items can be added to the list, then a refresh button is automatically added to the page. You can hide the refresh button if not necessary.	ShowLinksAsButtons	When items can be added and we have links New..., New From Find... or Refresh, by default these are rendered as links. You set this attribute and you can have these links rendered as buttons.
Attribute	Description												
AllowAddNew	The editable list supports addition of new rows via a popup dialog window. When this attribute is set, you need to additionally specify the NewTarget page element property.												
AllowAddViaFind	The editable list supports addition of new rows via a popup multifind dialog. When this attribute is set, you need to additionally specify the FindFilter, ParentMapper and ViewKey properties.												
AllowRemoveItems	Allow items to be deleted from the editable list. Items which already exist in the list when the page is opened, or items added to the list once the page is opened can be deleted from the list.												
NoRefresh	If items can be added to the list, then a refresh button is automatically added to the page. You can hide the refresh button if not necessary.												
ShowLinksAsButtons	When items can be added and we have links New..., New From Find... or Refresh, by default these are rendered as links. You set this attribute and you can have these links rendered as buttons.												
WizardPageAttributes	You would use this property to specify certain behaviors on this page. The majority of the attributes for this property refer to the visibility of the default summary information. A full explanation of summaries and summary options are given below.												
WizardPageSelectors	<p>The editable list page supports special functionality around the row selectors.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>InitiallySelected</td><td>The selector checkboxes are initially checked when the page is opened.</td></tr> <tr> <td>ProvideSelectors</td><td>Record selector checkboxes are added to each data row. This attribute must be specified for all other attributes to take effect.</td></tr> <tr> <td>RequireSelection</td><td>When the next button is clicked, you can specify that at least one of the rows must be selected.</td></tr> <tr> <td>SaveSelectedOnly</td><td>When the next button is clicked, typically an update event is fired on each row of the mapper, regardless of selection. However, if you only want to have update events fired for selected rows, then you would set this attribute. You would normally use this property when your editable list is displayed only on one page.</td></tr> <tr> <td>ShowSelectedOnly</td><td>The typical usage scenario for editable lists with selector is to have the editable list display on one page to allow a user to select one or more rows. Then, clicking next, the next page displays the same editable list, but filtered down to the selected rows from the previous page. If you want this behavior, then this attribute will be set on the editable list of the next page, not the previous page.</td></tr> </table>	Attribute	Description	InitiallySelected	The selector checkboxes are initially checked when the page is opened.	ProvideSelectors	Record selector checkboxes are added to each data row. This attribute must be specified for all other attributes to take effect.	RequireSelection	When the next button is clicked, you can specify that at least one of the rows must be selected.	SaveSelectedOnly	When the next button is clicked, typically an update event is fired on each row of the mapper, regardless of selection. However, if you only want to have update events fired for selected rows, then you would set this attribute. You would normally use this property when your editable list is displayed only on one page.	ShowSelectedOnly	The typical usage scenario for editable lists with selector is to have the editable list display on one page to allow a user to select one or more rows. Then, clicking next, the next page displays the same editable list, but filtered down to the selected rows from the previous page. If you want this behavior, then this attribute will be set on the editable list of the next page, not the previous page.
Attribute	Description												
InitiallySelected	The selector checkboxes are initially checked when the page is opened.												
ProvideSelectors	Record selector checkboxes are added to each data row. This attribute must be specified for all other attributes to take effect.												
RequireSelection	When the next button is clicked, you can specify that at least one of the rows must be selected.												
SaveSelectedOnly	When the next button is clicked, typically an update event is fired on each row of the mapper, regardless of selection. However, if you only want to have update events fired for selected rows, then you would set this attribute. You would normally use this property when your editable list is displayed only on one page.												
ShowSelectedOnly	The typical usage scenario for editable lists with selector is to have the editable list display on one page to allow a user to select one or more rows. Then, clicking next, the next page displays the same editable list, but filtered down to the selected rows from the previous page. If you want this behavior, then this attribute will be set on the editable list of the next page, not the previous page.												
Mapper	The name of the mapper for this page element. If no mapper is specified at the page element level, the page will use the mapper associated with the page.												
Caption	A caption to display on the wizard in the top right hand corner of the wizard page describing the purpose of that step to the user.												
Filter	This is where you specify the filter criteria for the editable list values. The filter criteria will probably contain references to either other page instance values, or values from the initial navigation.												

Property	Description
SummaryFilter	This filter applies to the summary of this editable list. Here you can provide a different filter for the summary information than used for the editable information. Typically your summary filter will be identical to the filter with perhaps one or two additional criteria. If no summary filter property is provided the Filter property is used for the summary filter.
SummaryMapper	You can specify an alternative mapper to use to generate the summary data for a editable list page. The SummaryFilter will apply to the SummaryMapper if a SummaryMapper is specified.
FindFilter	Used when you want to support the multi-add user interface to add new records to the list. The filter is applied to the search results of the multi-add .
ParentMapper	Required when the page supports addition of records via the new dialog, or the multi-add interface. The parent mapper is the parent to the editable list mapper.
ViewKey	Required when the page supports addition of records via the new dialog, or the multi-add interface. The foreign key is a field on the mapper to which the records are added (the editable list mapper).
NewTarget	Required when the page is set up to add new rows to the list. The target is a MOP that can add records to the same mapper as the list. The new page is opened as a popup in the popup dialog mode which provides the "Save and New" button.
NewTargetRowKey	This property is used for adding new rows to the list, either through new dialog, or multi-add interface. It is optional to set this property. If it is not set, the primary key of the primary instance is passed as the parent row key to the new popup dialog. This ensures that the newly saved record is related to the record associated with the mapper. If the primary instance is an existing record, the primary key on the wizard will be specified and this property is not necessary. If the primary instance mapper is a new record, the default RowKey parameter passed in the navigation to the new popup detail is blank. You can specify a fixed value, or a reference to an instance value (most likely property value for this property).

## WizardPhantomList Pages

If you wanted to use a static list as a wizard page, then you would create a new page element using the WizPhantomList component. You can just use this type of page to present the list with a way to choose one or more rows of the list when page selectors are in use. Without page selectors, the use of this type of page is somewhat limited.

Property	Description												
InstanceName	The name of the mapper instance for this page												
WizardPageAttributes	You would use this property to specify certain behaviors on this page. The majority of the attributes for this property refer to the visibility of the default summary information. A full explanation of summaries and summary options are given below.												
WizardPageSelectors	The editable list page supports special functionality around the row selectors.												
	<table><tr><th>Attribute</th><th>Description</th></tr><tr><td>InitiallySelected</td><td>The selector checkboxes are initially checked when the page is opened.</td></tr><tr><td>ProvideSelectors</td><td>Record selector checkboxes are added to each data row. This attribute must be specified for all other attributes to take effect.</td></tr><tr><td>RequireSelection</td><td>When the next button is clicked, you can specify that at least one of the rows must be selected.</td></tr><tr><td>SaveSelectedOnly</td><td>When the next button is clicked, typically an update event is fired on each row of the mapper, regardless of selection. However, if you only want to have update events fired for selected rows, then you would set this attribute. You would normally use this property when your editable list is displayed only on one page.</td></tr><tr><td>ShowSelectedOnly</td><td>The typical usage scenario for editable lists with selector is to have the editable list display on one page to allow a user to select one or more rows. Then, clicking next, the next page displays the same editable list, but filtered down to the selected rows from the previous page. If you want this behavior, then this attribute will be set on the editable list of the next page, not the previous page.</td></tr></table>	Attribute	Description	InitiallySelected	The selector checkboxes are initially checked when the page is opened.	ProvideSelectors	Record selector checkboxes are added to each data row. This attribute must be specified for all other attributes to take effect.	RequireSelection	When the next button is clicked, you can specify that at least one of the rows must be selected.	SaveSelectedOnly	When the next button is clicked, typically an update event is fired on each row of the mapper, regardless of selection. However, if you only want to have update events fired for selected rows, then you would set this attribute. You would normally use this property when your editable list is displayed only on one page.	ShowSelectedOnly	The typical usage scenario for editable lists with selector is to have the editable list display on one page to allow a user to select one or more rows. Then, clicking next, the next page displays the same editable list, but filtered down to the selected rows from the previous page. If you want this behavior, then this attribute will be set on the editable list of the next page, not the previous page.
	Attribute	Description											
	InitiallySelected	The selector checkboxes are initially checked when the page is opened.											
	ProvideSelectors	Record selector checkboxes are added to each data row. This attribute must be specified for all other attributes to take effect.											
	RequireSelection	When the next button is clicked, you can specify that at least one of the rows must be selected.											
	SaveSelectedOnly	When the next button is clicked, typically an update event is fired on each row of the mapper, regardless of selection. However, if you only want to have update events fired for selected rows, then you would set this attribute. You would normally use this property when your editable list is displayed only on one page.											
ShowSelectedOnly	The typical usage scenario for editable lists with selector is to have the editable list display on one page to allow a user to select one or more rows. Then, clicking next, the next page displays the same editable list, but filtered down to the selected rows from the previous page. If you want this behavior, then this attribute will be set on the editable list of the next page, not the previous page.												
Mapper	The name of the mapper for this page element. If no mapper is specified at the page element level, the page will use the mapper associated with the page.												
Caption	A caption to display on the wizard in the top right hand corner of the wizard page describing the purpose of that step to the user.												
Filter	This is where you specify the filter criteria for the editable list values. The filter criteria will probably contain references to either other page instance values, or values from the initial navigation.												
SummaryFilter	This filter applies to the summary of this editable list. Here you can provide a different filter for the summary information than used for the editable information. Typically your summary filter will be identical to the filter with perhaps one or two additional criteria. If no summary filter property is provided the Filter property is used for the summary filter.												

## Summary Pages

Wizard pages have the ability to show summaries of information that has been entered by the user. As the user proceeds through the wizard entering data the summary of previous entries gradually increases. For each instance of the wizard there is a summary available for it. In addition to these default summaries, you are able to add your own summary information to represent information about another piece of related data.

For example on a submit candidate wizard, you are creating a candidate activity record and the default summaries show information about the candidate. You also want to show a summary of the order related to the candidate summary. Then you can add a manual summary to summarize the order information.

## Default Summaries

To manage the way default summaries are displayed, you have to set the `WizardPageAttributes` on each page element of the wizard.

## Summary visibility choices

You manage the look of summaries by setting these `WizardPageAttributes`

- `ShowSummary` – necessary to set to true if you want summaries to be visible
- `SummaryHidePrimary` – do not show primary instance summary on the current page
- `SummaryInitOpen` – the pages summary should be initially opened
- `SummaryOmitPage` – this page's summary is not available on other pages
- `SummaryShowCurrentPage` – include summary for current page even if page is not in primary instance
- `SummaryShowPrimaryOnly` – Show only the primary summary on this page, rather than other instance summaries.

## Manual Summaries

To add a manual summary to a wizard page, you add a page element to the wizard that uses the MapperSummary component. You specify a filter for your summary so that your summary shows the data you need. Your filter will almost always contain field references, or references to query string parameters, or embedded functions.

For manual summaries the main decision you have to make is whether you are summarizing detail or list information. For detail information you don't have to specify any attributes as long as your filter filters down to one row of records. With no attributes set, the summary is laid out based on the mapper fields metadata and mapper's default columns properties.

For details you can also specify that the layout is vertical and all the fields will be laid out vertically in two columns. Label and Value.

For summarizing lists views of data, you would specify the layout is horizontal. The summary would then look like a locked list view.

When you do specify a manual summary, those manual summaries are the first elements on a wizard page, followed by default summaries, followed by the page layout itself.

## Insert or Update, Detail or List?

When you design your wizard, you must decide whether you want your wizard to insert a record, update a record, show list of records. That's easy enough, but the wizard has certain limitations as to when it can update or insert.

Type of page	Insert Allowed	Update allowed
Detail Layout (primary instance)	yes	yes
Detail Layout (additional instances)	insert only	never
List Layout (any instance)	never	update only

So, we're saying that on a detail page, only the pages related to the primary instance can be set as updatable. Other detail instances always result in inserts. And for List pages, they are only ever allowed to be updated.

When you navigate to a wizard (if it has a detail as the first page) you define whether the primary instance is designed to update an existing record by specifying the query parameter "req=nav". If the primary instance is intended to create a new record, then you would add the query param "req=new".

Setting the request type can be done in a manual navigation, or by setting the NavNew CommandAttribute in a navigation target of a Menu Command, or the RequestType property of a Navbar target.

If the wizard is always intended to create new records, you can set the `CreateNew WizardPageAttribute` on the first page element.

## Wizard Page Properties

For wizard pages (those using the `WizTemplate.aspx`, template), there are a set of properties unique to wizards

### Navigation Properties (Page)

On a wizard page there are two options to exit out of the wizard. You click on the Finish button, or you click on the Cancel button. When you design your wizard, you have to decide where to navigate when you click on these buttons. There are two common options. When you cancel you navigate back to the page that originally opened the wizard. When you finish, you navigate to the record you just created, or you navigate to the page that originally opened the wizard.

There are three properties related to the navigation. `Action`, `QueryParams`, `Target`

### Action Property

There are 6 options for an action...

Action property	Description
Blank Page	Navigate to a blank page (hardly ever used)
Repeat Wizard	Navigate back to the first page of the wizard using the same navigation parameters when the wizard was first navigated to.
MOP	Navigate to the specified MOP (in Target property) with no default filtering applied. For this type of action, you would additionally need to specify <code>QueryParams</code> for filtering or use on the target. You would use this type of navigation if your wizard submitted a candidate to an order, and you wanted to navigate back to the related order detail that the candidate was submitted to. The navigation to the target mop would include the <code>req=nav</code> parameter.
MOP with item	Navigate to the specified MOP (in Target property) passing the primary key value from the primary instance as the primary key for navigation. You can optionally add custom <code>QueryParams</code> . You would use this type of navigation if you create a new order wizard, and want to navigate to the order detail page of the order you just created. The navigation to the target mop would include the <code>req=nav</code> parameter. Don't use this option on the Cancel action because the primary instance value may not yet be defined.

Action property	Description
MOP, new record	Navigate to the specified MOP (in Target property) as a new target. You can optionally add custom QueryParams. You would use this type of navigation if you wanted to navigate to a new page at the end of the wizard. The navigation to the target mop would include the req=new parameter.
Return to Caller	Navigate back to the page that called the wizard. You would use this option if you navigated to the wizard from a detail and you wanted to return back to the same detail. You don't need to supply any additional QueryString properties. Internally, this acts like setting the MOP action with a target of [req.origmop:parmop] and a QueryParameter of pk=[req.origrk:parrk]. See below for more details.

### Target Property

Specify a MOP for the navigation. For this property you can use a literal static mop target, or use a reference to a mop. e.g.,

`assignments!detail`

or

`[req.parmop]` – navigate to the parent mop (caller)

or

`[req.origmop:parmop]` – navigate to the original parent mop (original caller) or the last page (previous caller)

Imagine you have the following scenario of a navigation chain, the table shows what the parmop and origmop query parameters refer to

Page navigated in chain	parmop is...	origmop is...
DetailPage	-	-
Wizard1	DetailPage	-
Wizard2	Wizard1	DetailPage
Wizard3	Wizard2	DetailPage

### QueryParams Property

With this property you can add any custom query parameters you might require for your navigation from your wizard. It's a useful way to pass contextual information to a target page if necessary.

An example of using QueryParams property you can use queryparam instance values, page instance values, literal values.

```
parkey=[req.parkey]&parmop=[req.parmop]&parmap=[req.parmap]&parrk=[req.parrk]&fk=[req.fk]&people_id=[newperson.people_id]&display_name=[newperson.display_name]
```

### WizardAttributes

The Wizard also provides a WizardAttributes property that allow certain behavioral changes to the wizard.

Attribute	Description
DescBelowSummary	A wizard can provide a descriptive text that appears at the top of each page of the wizard above any summary region. This text is obtained from the Wizard's Description property. Additionally, some descriptive text can specified at the PageElement level that appears below the Page text (and above the summary region), but is specific to each page element. By setting this attribute, you can force all the description text (Page and PageElement) to be rendered between the summary elements and the page controls.
DescTwoPart	By setting this attribute, the descriptive text will be split into two pieces. The Page level description will be rendered at the top of each Page above the summary, and the PageElement description will be rendered between the summary elements and the page controls.
ForceSortOrder	The wizard pages already support a sort order on each page element and this is how the pages are ordered through the wizard. However, if the wizard page is derived from another base wizard page. The order of the base page elements is not ordered within the ordering of the actual page elements. This means that the base page elements will appear after the actual page elements in any wizard. To force all page elements from base page and actual page to be sorted, you would need to set this attribute.
NoStepCaption	Do not display a step caption on each page. E.g. Step 1 of n ...

### Caption Overrides

We support the ability to customize the caption on the buttons and popup messages. There are text items you can add to the text collection of the page to override.

Override	Description
Finish_Caption	A caption to put on the finish button other than "Finish". This is added as a text record of the Page object
Continue_Caption	A caption to put on the continue button on the Report Wizard pages.

Override	Description
PopupContinueText	The text to display to the user when they are about to run a report. Overrides the default text.
WizardCancelPrompt	A message to show to a user after clicking the cancel button overriding the default cancel prompt. This is a dedicate property on the Page object.
Cancel_Caption	A caption to put on the cancel button other than "Cancel". This is added as a text record of the Page object
Wait_Text	A message to display to the user after clicking the finish button overriding the default Save message. This is added as a text record of the Page object.
Next_Caption	A caption to put on the Next button, other than "Next >". This is added as a text record of the Page object
Description	Puts the text in this property at the top of the wizard page just below the page caption. This is a dedicated property on the Page Element object.
Back_Caption	A caption to put on the Previous button, other than "< Previous". This is added as a text record of the Page object
Continue_Text	A message to display to the user after clicking the next button. This is added as a text record of the Page object.

### *Saving*

When you click Finish on a wizard each instance of mapper is saved. The order in which the mappers are saved is governed by any dependencies set up in the wizard through setting the CopyValues property. If there are no dependencies configure (through copy values), the mapper associated with the primary instance will be saved first.

## Session Properties/Preferences

Session properties provide a way to store contextual information about an object. The session properties that are added to an application are converted into a session object via the Code Generator. That object is then referenced by custom code to create full customer specific session object.

When you create a session property, you add a record to the Session Properties screen in the studio. The attributes chosen for the property will affect both the code generation (and therefore accessibility to the property) and the behavior of the property.

### Session Property Attributes

Attribute	Description
SessionPersist	This is the most important attribute to be specified. It will result in the code generation process creating a session property for this item and therefore can be referenced in code. You would set this attribute (by itself) if you wanted to create a session property that has its value determined from an identically named application property.
DynamicOnly	Setting this attribute implies that the session property will not be initialized from an identically named application property. The session property will have a default value (set from the DefaultValue property of the Session Property), but the property will be set at runtime, either after loading from the preference table, or from some custom code during start up.
GenEmbeddedFunc	Setting this attribute will result in an embedded function being registered that performs a simple replacement. The embedded function will have the same name as the session property. E.g. a session property called CustomerId will have an embedded function created called !fnCustomerId. The value replacement associated with the embedded function is obtained from the value of the session property. Typically your session property value is assigned at run time in the session startup event handler.
LongText	If the session property is created by the studio as a by-product of creating a new application level parameter (that is, a new application custom property) that process can specify whether the property sheet should be displaying a special edit box that supports the editing of data more than 255 characters. This attribute does not have any effect at runtime for a session property.
ReadOnly	Setting this attribute means the session property will only have a read only read only property generated (assuming SessionPersist). A session property marked as ReadOnly is not going to be marked as DynamicOnly because the property would need to be initialized. In this case, from an identically named Application property.
SupportsReferences	Not applicable to Session Properties
TreatAsField	Not applicable to Session Properties

## Session Property Examples

Requirement	Method
Session property that is initialized from an application property of the same name. FOR APPLICATION PROPERTIES	<p>There are a couple of ways to do this.</p> <p>The easiest method is to add a custom application property from the link on the application property page. In creating an application property through the new property wizard, it will also create a session property of the same name.</p> <p>For the value of the session property to be derived from the application property set in meta data, only the SessionPersist attribute should be set.</p> <p>If you need to create a session property manually, the key is to make the name of the session identical in spelling and case to the name of the session property. Once done, make sure that only the SessionPersist attribute is set.</p>
Session property that is initialized from code	<p>For other levels of preferences (the level is in fact arbitrary), where the session property is not associate with an application level property, you would manually create the Session Property. When creating such a property, you must set the SessionPersist and the DynamicOnly attributes.</p> <p>They DynamicOnly setting means not to default the value from an application property. The default value in this case is set by the meta data Default property of the Session Property.</p> <p>To set the value of the property, you must manually assign the value in some code. Ideally that code would be executed as soon as the application is loaded. The most typical place to execute this code is in the AfterLoad method of the application event handler.</p>
Session property that provides an embedded function for use in filtering or SQL statements (referring to the value of a session property in meta data)	<p>Basically you can use either of the above methods to create a session property and have the value assigned. To have an embedded function created for the property, you would additionally set the GenEmbeddedFunc attribute.</p> <p>If your session property is called MyProperty, then the embedded function created would be referred to as !fnMyProperty. The case of the embedded function must match the case of the property name. Usual syntax rules apply for embedded value replacement.</p>

## Session Property Levels

A session property is typically associated with a level. A level is just an abstract idea that helps define the purpose of the property. The Level of a session property is defined by the code that loads and saves the preference values and not related to any meta data definition. The names of the levels are also completely arbitrary.

## Creating a Session Object

Let's say you have three levels of preference. Country, Company, Person. The first step is to create three static methods that create an instance of a session object that automatically loads the appropriate property values for a given item and preference level. The static methods should be defined in an implementation specific session object, derived from the code generated session object.

It is also necessary to create a generic CreateInstance method that will be assigned as the Session object on the current Application object. That session object assignment will be set during the application event handler, AfterAuthenticate. It's at that point that the sufficient user information is available to construct a session object. It's typical that instance object associated with the application is equivalent to the person level of session object.

An example is given below.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using NetQuarry;

namespace CompanyName.Common
{
    /// <summary>
    /// Types of preference levels
    /// </summary>
    public sealed class PreferenceLevel
    {
        /// <summary>User level preferences</summary>
        public const string PERSON = "person";
        /// <summary>Company level preferences</summary>
        public const string COMPANY = "company";
        /// <summary>Country level preferences</summary>
        public const string COUNTRY = "country";
    }

    /// <summary>
    /// CompanyName Session object.
    /// </summary>
    [Serializable()]
    public class Session : CompanyName.Data.Generated._companySession
    {
        /// <summary>
        /// Creates and initializes the Session object.
        /// </summary>
        /// <param name="appCxt">The application context object.</param>
        /// <returns>The created session object.</returns>
        internal static Session CreateInstance(IAppContext appCxt)
        {
            Session session;

            if ((0 == string.Compare(appCxt.UserContext.ID, "EAP.Reporting.Util", StringComparison.Ordinal))
                || (0 == string.Compare(appCxt.UserContext.ID, "EAP.Scheduler", StringComparison.Ordinal))
                || (0 == string.Compare(appCxt.UserContext.ID, "EAP.Studio", StringComparison.Ordinal)))
            {
                ///---a real user has to be chosen for non interactive processes
                ///---unless of course you create users with id's described above
                ///---for each implementation it will be specific to that
                ///---these are just examples, but the purpose of this code is required.
                appCxt.UserContext.ID = "system_user";
                appCxt.UserContext.Name = "Non Interactive User";
                appCxt.UserContext.EmailAddress = "system@companyname.com";
                session = CreateInstancePerson(appCxt, "system_user");
            }
        }
    }
}
```

```

        else
        {
            session = CreateInstancePerson(appCxt, appCxt.UserContext.ID);
        }
        return (session);
    }
}
/// <summary>
/// Creates and initializes the Session object for person values only
/// </summary>
/// <param name="appCxt">The application context object.</param>
/// <param name="personID">The value of the person id.</param>
/// <returns>The created session object.</returns>
public static Session CreateInstancePerson(IAppContext appCxt, string personID)
{
    Session session = (Session)NetQuarry.Session.CreateInstance<CompanyName.Common.Session>(appCxt);
    session.PersonID = personID;
    session.LoadPreferences(personID, PreferenceLevel.PERSON);
    return (session);
}
/// <summary>
/// Creates and initializes the Session object for company values only
/// </summary>
/// <param name="appCxt">The application context object.</param>
/// <param name="companyID">The value of the company id.</param>
/// <returns>The created session object.</returns>
public static Session CreateInstanceCompany(IAppContext appCxt, string companyID)
{
    Session session = (Session)NetQuarry.Session.CreateInstance<CompanyName.Common.Session>(appCxt);
    session.CompanyID = companyID;
    session.LoadPreferences(companyID, PreferenceLevel.COMPANY);
    return (session);
}
/// <summary>
/// Creates and initializes the Session object for country values only.
/// </summary>
/// <param name="appCxt">The application context object.</param>
/// <param name="countryID">The value of the country id.</param>
/// <returns>The created session object.</returns>
public static Session CreateInstanceCountry(IAppContext appCxt, string countryID)
{
    Session session = (Session)NetQuarry.Session.CreateInstance<CompanyName.Common.Session>(appCxt);
    session.CountryID = countryID;
    session.LoadPreferences(countryID, PreferenceLevel.COUNTRY);
    return (session);
}
}
}
}

```

## Referring to a Session Object

In the startup process of the application we've already mentioned that the application object is assigned a session object. That session object created from static method declared above. The session object created is of type `CompanyName.Common.Session`, but the application session property is of type `NetQuarry.Session`. Therefore, when you need to refer to your company specific session object, you have to cast the `NetQuarry.Session` to the appropriate implementation specific object.

When the session object is created

```

/// <summary>
/// Handles the AfterAuthenticate event.
/// </summary>
/// <param name="sender">The application context object.</param>
/// <param name="e">Event arguments.</param>
public override void AfterAuthenticate(IAppContext sender, EAPEventArgs e)
{
    Comensura.Common.Session session = CompanyName.Common.Session.CreateInstance(sender);
    sender.Session = session;
}

```

```
}
```

And how to use it. Basically cast the session object on the app context to the implementation specific session object...

```
CompanyName.Common.Session cs = (CompanyName.Common.Session)appCxt.Session;
```

## Loading and Saving Session Properties

Session properties may be persisted to the database table `xot_preferences`. This is a system table that is added to all NetQuarry databases. That means, Studio, Meta Data and Operational Data.

### Load Preferences

To load preferences from the database, you simply use the `LoadPreferences` method on the `Session` object. The method is declared as

```
public void LoadPreferences(string ownerID, string level)
```

You've seen this method used already in the example above. Basically it takes an Id value of the owner of the preferences and the level of preference to load. The level parameter as already discussed defines the category of the preferences to load. The `ownerID` parameter defines which set of preferences in that category should be loaded into the session object.

You would typically not need to directly create and load a session object because you will invariably have defined a helper function on the implementation specific session object to create and load your session.

### Save Preferences

To save preferences to the database, you use the `SavePreferences` method on the `Session` object. The method is declared as

```
public void SavePreferences(string ownerID, string ownerLevel)
```

The parameters of this method are identical to `LoadPreferences` method. Basically it takes an Id value of the owner of the preferences and the level of preference to save. The level parameter as already discussed defines the category of the preferences to save. The `ownerID` parameter defines which set of preferences in that category should be saved back to the database.

Inside this function call the preference collection is interrogated and those preferences that are marked as `Dirty` (ie. been changed since loaded) and `Persist` are saved to the database. Preferences are always created with the `Persist` attribute specified.

## Preference Handling

The platform provides a mechanism for creating, storing and retrieving preference settings. The preference handling allows for a hierarchy of preferences to inherit down from higher to lower level.

For example, a preference exists to set a limit on expense expenditure., e.g MaxExpense. For a company there may be default limit. However, for specific users, the limit may be different. So the hierarchy of preferences is from company to person where the person level preferences overrides the company preference.

Preference	Company Level Value	User A Value (Person Level)	User B Value (Person Level)
MaxExpense	\$400	\$500	Not specified

When User A creates an expense, they see a limit of \$500. When User B creates an expense, they see a limit of \$400 as they take the default value from the company level (next level up from person).

### Preference Storage

All preferences are stored in a platform table `xot_preferences`. At no time should you manually write any data to this table. It is possible to select from this table, but doing so, you potentially lose the ability to read values in the correct hierarchy order.

As already mentioned, preferences are associated with 'levels'. The level is simply a name in which to group preferences related to the same business object. Within a level, preferences are associated with specific object data. For example, the Company preference level contains the preferences associated with companies in the system. Within the Company preference level, there are preferences specific to a company.

### Preference Schema (not all schema described)

Column	Description
<code>param_nm</code>	The name of the preference.
<code>owner_id</code>	The key value of the object associated with the preference level.
<code>owner_level</code>	The name of the preference level.
<code>pref_value</code>	The value of the preference for the <code>owner_id</code> .

### Preference Hierarchy

The preference hierarchy is a completely arbitrary construct. You will create a hierarchy based on your own requirements. The majority of situations you will have a Company and Person level of preference where Person level preferences default from the parent Company level preferences.

Your hierarchy can be branched, rather than a single trunk if it makes business sense. You just need to make sure that your branching does not create circular references between preference levels.

You express the hierarchy through metadata and code added to your session object.

A basic preference hierarchy is as follows. We'll use this hierarchy as an example of how to configure preferences on an application.

Global -> Company -> Person

## Setting Up Hierarchy in Metadata

To configure your hierarchy in metadata, open the studio and navigate to the Preference Levels tool (near bottom of tree)

You are presented with a list view to specify the preference hierarchy and necessary data to manage the preference hierarchy. The following table describes what the fields represent.

Field	Description	
Module	The module for the meta data. Probably best to create a new preference specific module for these purposes. For this example,. companyname_preferences	
Preference Level	REQUIRED FIELD. The name of the preference level. This will be the name associated with preferences when saved to the database	
Parent Preference Level	Optional Field. The name of the parent preference level in the preference hierarchy. If this is the highest level in the hierarchy, then this field is blank. Otherwise you set this to a name you specified in the Preference Level column.	
Session Instance Creator	REQUIRED FIELD. The name of a method that will be used to populate the set of preferences for that preference level. You will create a method with the same name in your Session object.	
Sort Order	Optional Field. The ordering of the preferences. This is not yet used.	
Attributes	Optional Field. A way to modify the behavior of the preference level. These attributes are based on the PreferenceLevelInfoAttrs enumeration	
	PreferenceLevelInfoAttrs Enum Value	Description
	1 – Disabled	The preference level is not in use
	2 – NoParent	The preference level has no parent and therefore should not interrogate a parent for a default set of preferences.
	4 – ParentsGenericInstance	The parent of this preference level is a generic session object, not a parent specific to a preference hierarchy level. The generic instance is an informal approach to having global preferences.
Owner Key Name	REQUIRED FIELD. This is the name of a field in preference mapper that defines which field is to act as the key for the preference UI. This will be explained in more detail, later when describing the UI requirements to manipulate preference values	
Owner Lookup Sql	REQUIRED FIELD. This is a parameterized SQL statement used by the preference UI to determine the value of the owner key name. Typically when you provide Preference UI, it's as a subform and therefore the foreign key is passed down to the preference page as a filter. It's this filter that is tagged onto the parameterized SQL to determine the value of the owner key.	
Parent Lookup Sql	Optional Field. This is a parameterized SQL statement used by the preference UI to determine the key value of this level's parent preference. If the current preference has no parent then this value is not provided. If the preference has a parent level, then a SQL statement must be provided.	

The following table describes the meta data used to define the Global -> Company -> Person preference hierarchy.

### *Setup of preferences as described*

Preference Level	Parent Preference Level	Session Instance Creator	Attributes	Owner Key Name	Owner Lookup Sql	Parent Lookup Sql
global		CreateInstanceGlobal	4	global_id	select 'global_id'	
company	global	CreateInstanceCompany	0	company_id	select company_id FROM company WITH(NOLOCK) WHERE {0}	select 'global_id'
person	company	CreateInstancePerson	0	person_id	select person_id FROM person WITH(NOLOCK) WHERE {0}	select company_id FROM person WITH(NOLOCK) WHERE person_id = {0}

Here the global preference level has a parent instance which is the default session instance.

### *Setup of preferences with Global preference as top level (no generic instance as parent)*

Preference Level	Parent Preference Level	Session Instance Creator	Attributes	Owner Key Name	Owner Lookup Sql	Parent Lookup Sql
global		CreateInstanceGlobal	2	global_id	select 'global_id'	
company	global	CreateInstanceCompany	0	company_id	select company_id FROM company WITH(NOLOCK) WHERE {0}	select 'global_id'
person	company	CreateInstancePerson	0	person_id	select person_id FROM person WITH(NOLOCK) WHERE {0}	select company_id FROM person WITH(NOLOCK) WHERE person_id = {0}

Here the global preference level has no parent instance as specified by the attribute value of 2.

The owner lookup sql for global (also parent lookup) is set to return the value 'global\_id' which is the value key associated with the global preferences.

Once the metadata is specified, the next step is to run the Code Generation tool. The new preference level metadata is interrogated and adds some code to the generated Session object.

For each non disabled preference level row, the CodeGenerator creates...

- 1) A function stub that throws an error when a session object is created for that preference level. This is to remind you to create an actual implementation of the session creator for that preference level in the derived session object.

e.g.

```
/// <summary>
/// Stub method for CreateInstanceCompany
/// </summary>
/// <param name="appCxt">The application context</param>
/// <param name="company_id">The owner id value for the session</param>
public static CompanyName.Session CreateInstanceCompany(IAppContext appCxt, object company_id)
{
    string msg = string.Format(appCxt.TextItems.GetText(NetQuarry.Globalization.IDS.Session.SessionCreatorNotImpl,
        "You must implement a method for '{0}' in your derived session object.")
    throw new EAPEException(msg, "CreateInstanceCompany");
    return new CompanyName.Session();
}
```

- 2) A preference level constant name that you can refer to in your code.

```
namespace CompanyName
{
    #region PreferenceLevel Generated Code
    /// <summary>
    /// Types of preference levels defined in the PreferenceLevels meta data
    /// </summary>
    public sealed class PreferenceLevel
    {
        #region PreferenceLevel Constants
        /// <summary>company level preferences</summary>
        public const string COMPANY = "company";
        #endregion
    }
    #endregion
}
```

## Session Instance Creators

The next step is to create session instance creators in your derived class matching the stub methods in the generated session class.

If you look carefully at the example stub method, you will see that the `company_id` has a .NET type of object. This is because we don't know the actual data type of the `company_id` field. In your own implementation of this method you will know the data type of the `company_id` field, so you would set the method signature to match your data type. Assuming the `company_id` is string, in your derived session object (derived from the generated session class), create the function ...

```
public static Session CreateInstanceCompany(IAppContext appCxt, string companyID)
{
    Session session = (Session)NetQuarry.Session.CreateInstance<CompanyName.Common.Session>(appCxt);
    session.CompanyID = companyID;
    session._appCxt = appCxt;
    session.LoadPreferences("global_id", PreferenceLevel.GLOBAL);
    session.LoadPreferences(companyID, PreferenceLevel.COMPANY);
    return (session);
}
```

This example shows how the global preferences are loaded, followed by the company level preferences.

## Creating Preferences

That's the basics covered for the preference infrastructure. The next step is to define what preferences are to be used. Open the Studio and navigate to the Session Properties. If you want a preference to represent the maximum value of expenses created, you add Session Property with the name "MaxExpenses". The type would be set to Decimal, the category set to Custom and the attributes set to 160 (Dynamic Only and SessionPersist).

In terms of what preferences to add, you should probably add two further items. UpdatedBy (type: string, attrs: 160) and DateUpdate (type: string, attrs: 160). These will store information about who updated preferences and when.

Now you need to regenerate the session object again to get the new property associated with the session object.

## Preference UI

### Preference Mapper

The first step to create preference UI is to create a preference mapper. You can base the mapper on any view/table and one as good as any is xot\_preferences. The key to remember is that this table will not be interrogated directly when the preferences are displayed and saved.

For example purposes we assume you are creating preferences for companies.

Create a new mapper, "company\_prefs" using View Name of "xot\_preferences".

Manually add the fields to this mapper.

Key Name	Col Width	Cell Type	.Net Type	Data Type	Attrs	Notes
company_id	20	text	string	string	4	Attrs: Set to Primary Key of mapper
UpdatedBy	20	text	string	string	514	Attrs: Locked and UseDefaultOnUpdate. Set properties DefaultValue to <b>!fnUserID()</b> Caption to Updated By
Date Updated	20	text	datetime	datetime	514	Attrs: Locked and UseDefaultOnUpdate. Set properties DefaultValue to <b>!fnNow()</b> Timezone to <b>0</b> , Format to <b>g</b>
MaxExpenses	15	Currency	Decimal	money	0	Set properties Culture to field reference pointing to a currency value (or hard coded)

You can add further preferences to this mapper. The important point to remember is that your field keyname should match exactly the case and spelling of the Session properties you add.

### Preference Handling Extension

Now you have to create a preference extension. You only have to create one extension to be used on all preference mappers. There is a platform extension template class from which you derive your preference extension.

Create a new extension class, for example PreferenceHandler. You can use your own name, whatever makes sense to you.

In your project, add a reference to EAP.Extensions.Preferences. Then derive your extension class from the NetQuarry.Extensions.Preferences template class. You specify the type of your session object as the template parameter.

```
/// <summary>
/// Implementation of generic preferences extension using CompanyName.Common.Session as template type.
/// </summary>
public class Extension : NetQuarry.Preferences.Extension<CompanyName.Common.Session>
{
}
```

To get the preference handling to work, there is no further coding required.

Once compiled and added as an extension in the components list in studio, you can associate your extension with the preference mapper.

### *Preference Pages*

Now that you have a mapper, you can create a preference page to display the preferences. The page must use the TabbedSubformTemplate template. The main slot uses the phantomdetail component. For this example, we'll call the page companyname\_preferences!company\_prefs.

### *Preference Subform*

Associate this page with a subform hanging off your main company list, or detail page. Set up the ViewKey relationship to pass down the company\_id from the parent to the subform

### *Preference Levels Revisited*

There is one last task to perform and that is associate a preference level with a page that presents the preferences for that level to the user. It's quite possible to have many different pages associated with the same level of preferences. You might have different types of companies, that have different preference pages.

So back in the studio, navigate to the Preference Levels metadata and select the row associated with the company preference level. In the Pages subform, add the page companyname\_preferences!company\_prefs.

Now the preference support is fully configured. All that's left is for you to visit your preference page on your company subform and layout the fields as you prefer.

### *Adding More Preferences*

That process may seem a little complicated, but once the preference handling has been set up for one preference level, add preference UI support for other levels, is a question of repeating the same metadata steps. No more code changes are required, other than to add your Session Instance creator methods.

### *Adding New Preference Level*

- Add Session Instance Creator
- Add new Preference Mapper
- Add new Preference Page
- Add new Preference Subform
- Add Preference Level and associate with Preference Page.
- Regenerate Session class.

### *Adding New Preference*

- Add to Session Properties
- Add to preference mapper
- Modify layout on page
- Regenerate Session Class

## Using Stored Procedure as the DataSource for a mapper

There may be occasions where the built in functionality of mappers is not sufficient to give the appropriate results and the only way to get the right results is to use a stored procedure. Once you've decided to use a stored procedure, you will clearly want to have the results of your procedure use any filtering specified by the developer and any filtering specified by the end user. Because you are also relying on the stored procedure to send back results to the mapper to be displayed, it is also up to the creator of the procedure to support Sort criteria and to provide a count of records matching the specified criteria.

### The Procedure

If you want to fully support all the available functionality, your procedure must accept the following type of parameters.

FILTER	The filter criteria specified by mapper, filters, page filters, navigation filters as well as filters entered by the user on the filter by form row for fields not marked with the FilterOptions.PreFilter attribute.	
PREFILTER	The filter criteria specified by users entered on the filter by form row for fields that are marked with the FilterOption.PreFilter attribute.	
SORT	The sort criteria for the results.	
OPTIONS	A flags parameter hints what type of results are required to be returned to the mapper. The flags are	
	0	The procedure must return a result set to be displayed to the user. This populates the list view or detail of a page.
	1 (aggregate)	The procedure must return a single row result set containing the expected aggregated data.
	2 (count)	The procedure must return a number representing the count of records matching the filter criteria. The count must match the number of records returned when the procedure is called with OPTIONS of 0
Any other parameters you want	Any static parameter, or dynamic parameter represented by an embedded function.	

### Filter Clauses

The PREFILTER is intended to be used to perform filtering to obtain an intermediate set of results. If you are performing an aggregate, you would want to filter the data being aggregated by a certain set of criteria before filtering the post aggregated data.

e.g. you could prefilter data to be aggregated by asking for data only from a certain day and then filtering the sums or averages of the aggregated data using the FILTER criteria.

### Example

Assuming you have the following filter criteria...

Mapper (static filter): `company_id=!fnCompanyID$()`

Page (static filter): status\_id=1

FBF Field (date\_submitted PreFilter): date\_submitted = Last Month

FBF Field (revenue not PreFilter): revenue > £50000

The PREFILTER criteria will be constructed as follows

```
(date_submitted >= '2008-12-01 00:00:00' AND date_submitted < '2009-01-01 00:00:00')
```

The FILTER criteria will be constructed as follows

```
company_id='SOME GUID' AND status_id=1 AND revenue > 50000
```

### *Problems With Filter Clauses*

You may have noticed a problem with the current implementation in that we expect the static filters (because they are munged with the non PreFilter criteria) to be applied after the pre filter. When you are performing aggregations, you really want to aggregate as little data as possible to improve performance. You would want the static filters to be applied initially in the PreFilter stage and then aggregate a few records to be filtered again in the final FILTER stage with the regular FBF filters.

[Fogbugz Case 4405](#) has been created to implement the filter clauses correctly so the static filters may be applied during pre or post filtering, leaving that decision to the developer.

Here's an example skeleton of a procedure.

```
IF EXISTS (SELECT name FROM sysobjects WHERE name = 'my_procedure')
    DROP PROC dbo.my_procedure
GO
```

```
CREATE PROCEDURE [dbo].[my_procedure]
-- Add the parameters for the stored procedure here
    @pPreFilter varchar(4000) = '',
    @pFilter varchar(4000) = '',
    @pSort varchar(200) = '',
    @pOptions int = 0,
    @pExtraData varchar(4000) = ''
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- CREATE A TemporaryTable
    CREATE TABLE #MyTemp
    (
        company_id varchar(32),
        revenue money,
        -- etc
    )

    -- insert the temp table

    INSERT INTO #MyTemp
    SELECT company_id,
        revenue,--etc
    FROM MyTable
    WHERE @pPreFilter
```



```
IF @pOptions & 0x00000001 = 0x00000001 -- AGGREGATE RESULTS
BEGIN
    SELECT
        null AS company_id, -- select NULL's for non aggregated fields
        SUM (revenue) as revenue
    FROM #MyTemp
    WHERE @pFilter

    RETURN
END
ELSE IF @pOptions & 0x00000002 = 0x00000002 -- COUNT RECORDS
BEGIN
    SELECT COUNT(*) FROM #LEDGER_TRANSACTIONS
    RETURN
END
ELSE -- RETURN RESULTS
BEGIN
    SELECT * FROM #MyTemp WHERE @pFilter ORDER BY @pSort
    RETURN
END
END
GO
```

## Setup the Mapper

Now you have the procedure, you can move on to setting up the support in metadata. The first thing you need to do, however, is create a dummy table or view that contains the same schema as your procedure would return. You would use this to initially create your mapper using the database slurper. It's much quicker to do it this way than manually adding fields.

If you do create a dummy table, or a dummy view, you should remember that the slurper is going to try its best to identify primary key fields to support insert and update of data. Be careful here. Your procedures are obviously not updateable and the tables to update that your slurper may identify probably shouldn't be accepting data changes from a stored procedure based mapper. We'd recommend making your procedure based mapper read only.

In the mapper attributes, set the SkipCodeGeneration attribute. The code generator attempts to perform a query on the underlying data and may fail if the stored procedure parameters are not set up properly. It's unlikely you will want to use such a mapper in code as it's not updateable.

Finally, in the QuerySQL property of the mapper, you specify your procedure and any arguments that are necessary.

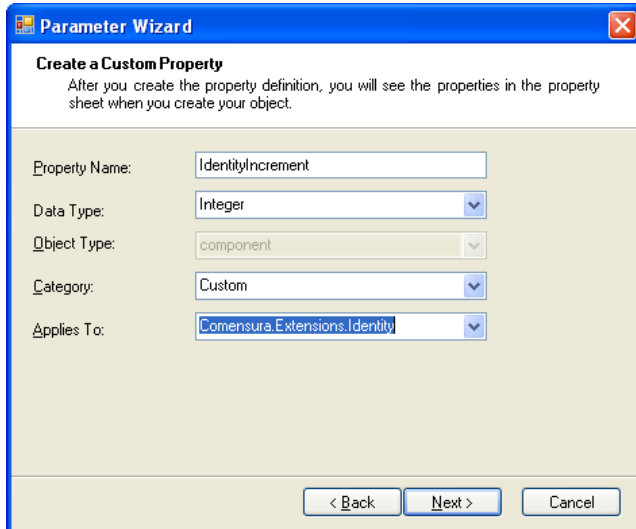
```
my_procedure [[PREFILTER]], [[FILTER]], [[SORT]], [[OPTIONS]], !fnUserID$()
```

## Custom Properties

When you add re-usable components to your application, you may want to allow configuration properties to be added to your object so that in different implementations of your object, you can have different behavior.

### Adding Custom properties

At the bottom of each property sheet, there is a link “Add Property” that pops up a wizard to let you specify the details of the property.



**Parameter Wizard**

**Create a Custom Property**  
After you create the property definition, you will see the properties in the property sheet when you create your object.

Property Name:

Data Type:

Object Type:

Category:

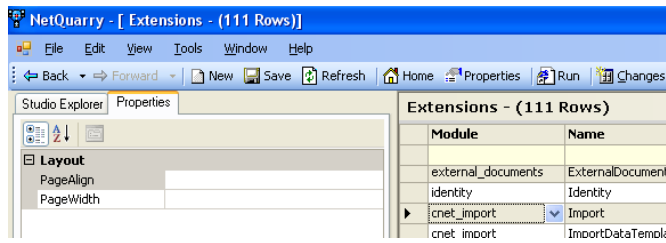
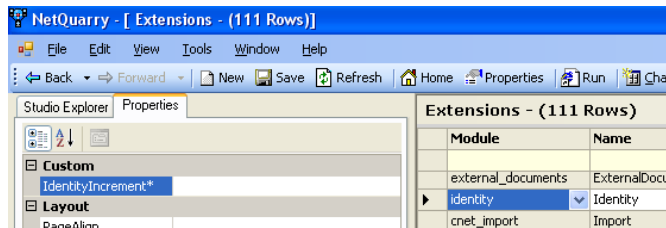
Applies To:

< Back   Next >   Cancel

Field	Description
Property Name	The name of the property that you want to add. This will appear in the property sheet of object you’re adding the property to. The property name should not contain spaces
Data Type	The data type of the property. This is not just a standard set of .Net datatypes, like string, int, etc. You can set the datatype to be Mappers, for example and then when you use the property, you will have a picklist from which to select one of the mappers in the application. The full list of data types is found in the studio under System Maintenance, Parameter Types.
Category	The category of the property. This is a simple categorization of what your property belongs to. In the property sheet of an object you’ll see different categories of properties. Here, you decide which category your property goes into. The list of available categories can only be modified by NetQuarry.
Applies To	This setting may not be visible when you add a custom property. It’s only supported for certain objects. If you don’t see this property, it means that your property will be added to the property sheet and will be visible on the property sheet of every other object of the same type. If you do see this property, you will only see the custom property on the object to which you added the property. You’ll only see this property on Components and Scheduled Tasks.

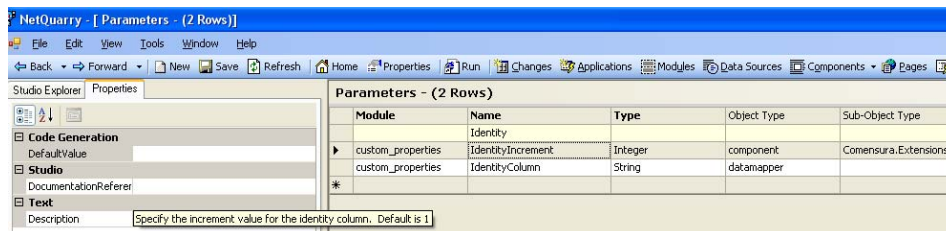


After adding a new custom property, you'll see it in the property sheet with a little asterisk next to the name, indicating it is a custom property.

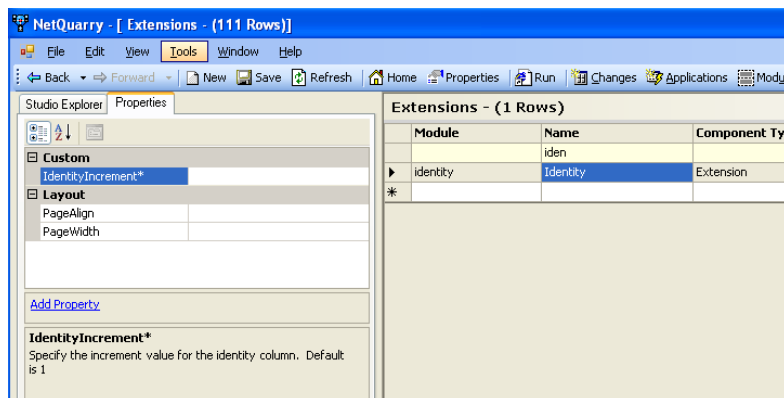


Finally, you also might like to give your property a description so that other developers will know how to use your property. This is especially important on properties added to objects where the property appears for all objects of that type. Your property may only be valid on that single object.

To add a description, go to System Maintenance, Parameters. Find your property and set the Description property.



Your property will now have a description (you have to shutdown and restart studio to see the immediate effect).





### **Deleting Custom Properties**

To delete a custom property, you go to System Maintenance, Parameters and delete your custom property.

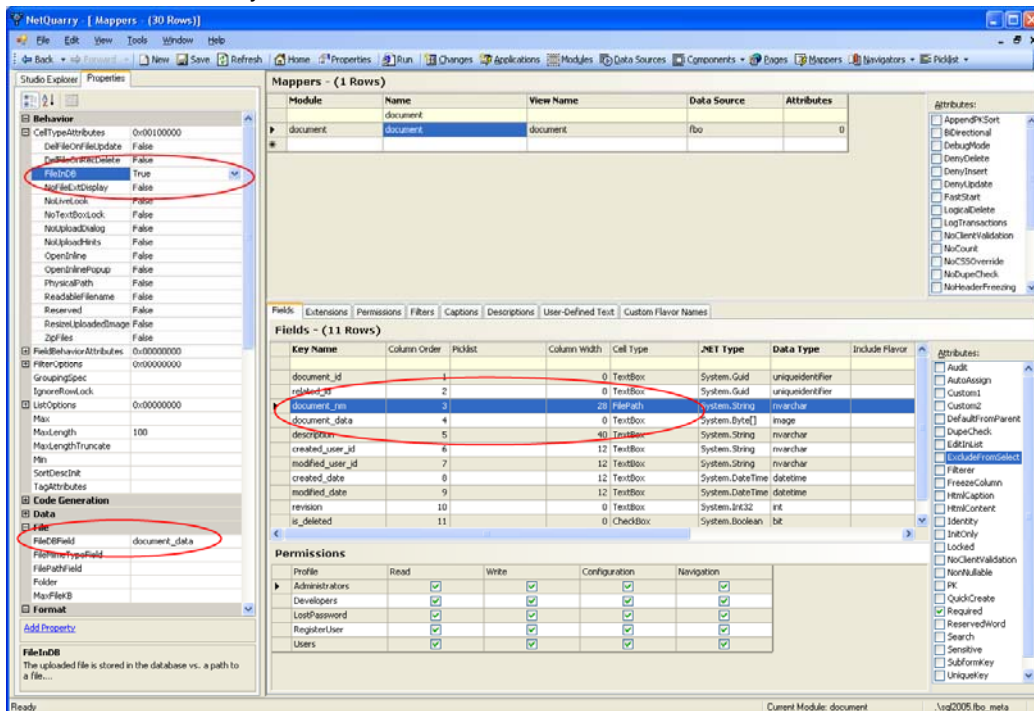
You should be careful when deleting custom properties, just in case you delete a property of the same name associated with a different object.

## How to store documents and images into a BLOB field

1. In order to store a file in a table in the database, you must have 2 fields defined– 1 varchar field for the name of the file and 1 image field for the file data. For the purposes of this example, we have

	Column Name	Data Type	Allow Nulls
🔑	document_id	uniqueidentifier	<input type="checkbox"/>
	related_id	uniqueidentifier	<input checked="" type="checkbox"/>
	document_nm	nvarchar(100)	<input checked="" type="checkbox"/>
	document_data	image	<input checked="" type="checkbox"/>
	description	nvarchar(200)	<input checked="" type="checkbox"/>
	created_user_id	nvarchar(64)	<input checked="" type="checkbox"/>
	modified_user_id	nvarchar(64)	<input checked="" type="checkbox"/>
	created_date	datetime	<input checked="" type="checkbox"/>
	modified_date	datetime	<input checked="" type="checkbox"/>
	revision	int	<input checked="" type="checkbox"/>
	is_deleted	bit	<input checked="" type="checkbox"/>

2. Set the Field Type of the text field (document\_nm) to **FilePath**, make the field required, and set the FileDBField to the key name of the FilePath field.



The screenshot shows the NetQuarry Mappers - (30 Rows) window. The 'document' table is selected. The 'document\_nm' field is highlighted with a red circle. In the 'Fields' section, the 'document\_nm' field is configured with the following properties:

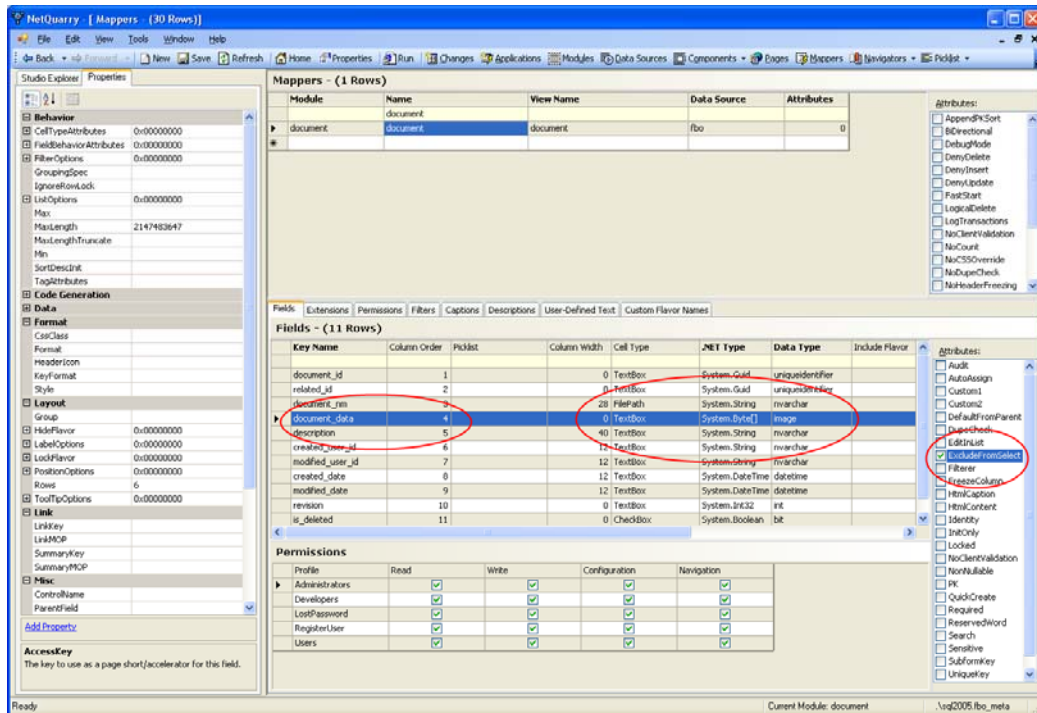
- Key Name: document\_nm
- Column Order: 3
- Picklist: False
- Column Width: 20
- Cell Type: FilePath
- NET Type: System.String
- Data Type: nvarchar
- Include Flavor: True

In the 'Data' section, the 'FileDBField' is set to 'document\_data'.

The 'Fields' section also lists other fields in the table:

Key Name	Column Order	Picklist	Column Width	Cell Type	NET Type	Data Type	Include Flavor
document_id	1	False	0	Text	System.Guid	uniqueidentifier	False
related_id	2	False	0	Text	System.Guid	uniqueidentifier	False
document_nm	3	False	20	FilePath	System.String	nvarchar	True
document_data	4	False	0	Text	System.Byte[]	image	False
description	5	False	40	Text	System.String	nvarchar	False
created_user_id	6	False	12	Text	System.String	nvarchar	False
modified_user_id	7	False	12	Text	System.String	nvarchar	False
created_date	8	False	12	Text	System.DateTime	datetime	False
modified_date	9	False	12	Text	System.DateTime	datetime	False
revision	10	False	0	Text	System.Int32	int	False
is_deleted	11	False	0	Text	System.Boolean	bit	False

- Set the BLOB field's width to 0 (hidden) and set the **ExcludeFromSelect** attribute as well. *Note that you should NOT set any exclude flavor or the FilePath field will display 'ERROR!'.*



## How to retrieve data image files in a template

You can use an image stored in the database in a template with simple substitution. The image is returned as a binary object and will display normally in an image tag. For this example, assume the table is called companies, with the image stored in the field logo\_data, and the primary key is company\_id.

```

```

Parameter	Description	Value
req	Request Type	getdbf
dbservice	Datasource Key	<datasource key>
dbkeyfilter	Row filter that will return 1 row	The SQL clause escaped for URL
dbtable	Name of the database table	<table name>
dbfield	Name of the database field	<field name>
src	The file name	<name of the file>
inl	1 for inline, 0 for attachment	1



## How to programmatically add a file using a mapper

The following code example adds a static method to a TypedMapper based on the document mapper defined above:

```
using System.Web;
using System.IO;

public class Document : FBO.Data.Generated.document<Document>
{
    /// <summary>
    /// Adds a new record and attaches the document
    /// </summary>
    /// <param name="appCxt">The application context object</param>
    /// <param name="relatedID">The primary key of the related item.</param>
    /// <param name="fileName">The full path to the file</param>
    /// <returns>The ID of the new record, as a string.</returns>
    public static string AddNew(IAppContext appCxt, Guid? relatedID, string fileName)
    {
        string documentID = null;
        using (Document doc = Document.OpenNew(appCxt))
        {
            ///--- This has to be done in 4 steps:
            ///--- Step 1: Copy the file to the appropriate location. The mapper expects the file to be in the folder
            ///--- specified by the Folder property on the field. By default, we're using 'UserFiles' as the folder as
            ///--- it's created during the install.
            ///--- Step 2: Copy the file to the location, and set the name of the file to ONLY the name,
            ///--- not the full path. (Looks better and the path has no meaning once the file is stored in the database.
            ///--- Step 3: Add the related document record.
            ///--- Step 4: Delete the document

            string filePath = HttpContext.Current.Server.MapPath(
                doc.Fields.document_nm.Properties.GetStringValue("Folder", "UserFiles"));

            string tempFileName = Path.Combine(filePath, Path.GetFileName(fileName));

            if (!File.Exists(tempFileName))
            {
                File.Copy(fileName, tempFileName, true);
            }
            doc.document_nm = Path.GetFileName(fileName);
            doc.related_id = relatedID;
            doc.Save();
            documentID = doc.document_id.ToString();
            ///--- Step 4: (optional) delete the temporary file
            File.Delete(tempFileName);
            doc.Close();
        }
        return documentID;
    }
}
```

## Showing icons in the list

You can show icons in a list using a Picklist set on a column that has images as its text.

Example:

[Account Opportunities](#) [My Opportunities](#)

				<b>Subject</b>	<b>Priority</b>
				<input type="text"/>	<input type="text"/>
				<a href="#">Elf Point Pre-commercial Thinning</a>	Medium
				<a href="#">Blanchard Recreation Area Clean Up</a>	High
				<a href="#">Refugio County Bank Stabilization - EWP</a>	
				<a href="#">Riparian Fisheries Consultation Services</a>	

For this example, the mapper has a field **priority\_id** (int) which has a Picklist with the values High, Medium, Low. To add an image column linked to the priority\_id follow these steps:

1. Copy the field and rename it. (priority\_icon)
2. Set the field's type to Icon
3. Set the field's AliasName to priority\_id
4. Create a new picklist with the following values:
  - 1 – images/1x1.gif
  - 2 – images/1x1.gif
  - 3 – apps/<your appid>/images/high.png (16x16 image)

PickList Items - (3 Rows)			
	Name	Alternate Key Text	Alternate Key Int
▶	images/1x1.gif	images/1x1.gif	1
	images/1x1.gif	images/1x1.gif	2
	apps/fbo/images/high.png	apps/fbo/images/star_red.gif	3
*			

5. Set the following attributes on the list:
  - Cache, KeySameAsText, LimitToList, StoreAltInt (1048645)
6. Set the Picklist on the new field to the new Picklist.
7. Set the field's width to 3
8. Set the field's Caption to a single hyphen (-).

## Using Built In Auditing

You can set up a mapper to be audited where all the field changes of a mapper will be recorded in a platform table. There are two types of audit. A very rich “Readable Audit” and a much simpler “Simple Audit”. We recommend that you always use the Readable Audit process. The auditing process is provided by platform extensions. The display of audit information is the responsibility of the developer.

### Auditing Changes with Readable Audit

On a mapper you want to audit, add the ReadableAudit.Audit extension. This is a platform extension.

The audit extension has some properties with which you can customize the behavior of the auditing process.

Audit Extension property	Description
AtomicFields	A semi-colon separated list of keys for fields whose values should be diffed in their entirety instead of character by character.
AuditPrefix	Mapper expression to be resolved and used as change description prefix styled using the audpre CSS class. You could use this property if you are auditing different mappers “as the same mapper” using the MapperKeyOverride property. You could then provide an indication of the exact source of audit record.
AuditSeparate	A true or false field that defines whether each changed field should be logged as a separate audit record. By default the audit process creates one field per insert/update/delete, regardless of how many fields were modified.
CustomAudit	A mapper expression evaluated against the audited mapper and written to the xot_audit_readable.custom_audit column.
MapperKeyOverride	Specified the mapper key to write to the xot_audit_readable.mapper_key column when auditing this mapper. This is used when you want a mapper to be audited as if it were from another mapper. Use if the mapper you are auditing is, for example, contains a subset of data from a main object mapper. If you present modifications based on the main object type, then this can ensure all your data changes are related to the correct object.

Audit Extension property	Description
ParentField	<p>The field in the mapper being audited that stores the foreign key to that record's parent record. In auditing, the audit record is associated (rel_id) with the primary key of the mapper that has changed. However, if the mapper has a foreign key field to a parent mapper, you can specify what that field is and then the value of that foreign key field is also associated with that audit record. Therefore you can "roll up" the audit history to the parent mapper of the audited mapper.</p> <p>E.g., A customer has a note. If you change a note, the note is audited and the audit rel_id value is the note_id. You can set the ParentField property to customer_id and the value of that field is also recorded with the audit record. Then when showing audits for a customer, the note audit will also appear.</p>
PKOverride	Specifies the field, in the mapper being audited, to use instead of the PK field when obtaining a value for the xot_audit_readable.rel_id column.

### Example of properties usage

This example shows the output of the audit process where AuditSeparate is false. Location zip is specified in the AtomicFields list.

Audit History (Filtered on *Parent*) [1 - 10 of 10] [Refresh](#) [More](#) [Filters](#)

	Changed Items	Change Description	Change Date	User
7	<input type="checkbox"/> Panels Replaced Detail	Panels Replaced Detail: <del>Left Nearside channel</del>	9/3/2009 9:50:41 AM	rabbey@netquarry
6	<input type="checkbox"/> Panels Replaced Detail	Panels Replaced Detail: Left Nearside <del>channel</del>	9/3/2009 9:49:30 AM	rabbey@netquarry
5	<input type="checkbox"/> Panels Replaced Detail	Panels Replaced Detail: Left Nearside <del>panel</del>	9/3/2009 9:49:00 AM	rabbey@netquarry
4	<input type="checkbox"/> Description, Region	Description: 2007 BMW 3-Series 2dr Cpe 335i RWD (Grey Poupon); Region: <del>North-West</del> North West	9/3/2009 9:48:00 AM	rabbey@netquarry
3	<input type="checkbox"/> Description, Location Zip, Region	Description: 2007 BMW 3-Series 2dr Cpe 335i RWD (Grey Poupon); Location Zip: <del>98901</del> 98902; Region: <del>North-West</del> North West	9/3/2009 9:47:08 AM	rabbey@netquarry
2	<input type="checkbox"/> Tires Front, Transmission	Tires Front: <del>Better than 60%</del> Worse than 60%; Transmission: <del>Stick</del> Automatic	9/3/2009 9:45:25 AM	rabbey@netquarry
1	<input type="checkbox"/> Color	Color: <del>Black</del>	9/3/2009 9:44:30 AM	rabbey@netquarry

Example Row	Description
1	New value added to a picklist field
2	New value entered into text field and modified value of picklist field
3	Description modified in text field (word added), Location Zip text field modified (but field is marked as "atomic" in AtomicFields property).
4	Description modified in text field (character changed in word)
5	New value entered into Panels Replaced text field



Example Row	Description
6	Modified character in Panels Replaced text field
7	Deleted value from Panels Replace text field

This example is showing when AuditSeparate is set to true

Audit History (Filtered on *Parent*) [1 - 12 of 12] [Refresh](#) [More](#) [Filters](#)

Changed Items	Change Description	Change Date	User
<input type="checkbox"/> Panels Replaced Detail	field changed	9/3/2009 10:09:25 AM	rabbe@netquarry
<input type="checkbox"/> Damage Description	another field changed	9/3/2009 10:09:25 AM	rabbe@netquarry

## Auditing Fields

By default, no fields are automatically audited. To audit a field you must set the Audit field attribute.

When an insert or delete occurs, all of the audit fields are audited. During an update, a field is audited only when the field is dirty and when the old value and new value do not equal.

## Displaying Readable Audit Records

The platform readable audit is stored in the table `xot_audit_readable`. If you want to display audit information, you create a mapper from this table. There is another platform database object, `xov_audit_readable_rollup`. With this view you can create a mapper where you display rolled up audit information.

After you create the mapper from the `xot_audit_readable`, or `xov_audit_readable_rollup`, There are a couple of necessary meta data changes to the audit mapper fields.

Set the `change_text` field to HTML cell type and with the `HTMLContent` attribute. Since you typically display the audit information in a subform list, then set the `AllowListWrap` `FieldBehaviorAttribute`.

When you set up the subform to display audit information, whether you are showing regular audit, or `audit_rollup`, then your `ViewKey` property will be set to `rel_id`

## Auditing Changes with Simple Audit

On a mapper you want to audit, add the `SimpleAudit.Audit` extension. This is a platform extension.

There are some significant limitations in using `SimpleAudit`. There are no customizable properties to modify behavior and only updates are audited.

During an update, the `SimpleAudit` extension, interrogates the mapper to determine which fields are marked to be audited and which of those fields are dirty.

A record is then inserted into the `xot_audit_simple` table specifying the old and new value in separate columns.



### **Displaying Simple Audit Records**

The platform simple audit is stored in the table `xot_audit_simple`. If you want to display audit information, you create a mapper from this table.

When you set up the subform to display audit information, your ViewKey property will be set to `rel_id`.

## Using Console Pages

Console Pages provide the ability to display multi-faceted information about an object on a single page. The console page has the idea of a parent object with related child objects. Therefore, this concept is similar to the main page/subform relationships on standard list/detail pages.

All console pages must have a Main slot specified. The Main slot object doesn't necessarily have to be a parent of the child elements for the console page to be rendered correctly, but for this documentation, we'll consider the situation where the Main Slot is related to the data in the child elements.

## Page Properties

The properties you can specify at the main page level are limited.

Console Template Property	Description	
TemplateAttributes	These affect the behavior of the console template.	
	Attribute	Description
	NoElementMove	The console page elements cannot be moved around by the user.
	ShowToolbar	The main toolbar buttons should be displayed on the console page.
Columns	Here you can specify the number of columns in the console page matrix of Main element and ConsolePage page elements.	
ColumnWidths	Here you can specify a fixed width of one, or more columns in the console page matrix. If there are two columns, and you don't specify a width, the page elements a sized at 50% of the page width. You can only specify the widths of the left most columns. E.g., you have two columns and set the width to 100px. The first column will be 100px wide, the second column will size based on the size of the screen.	
PageCommands	You can set the page to have page commands, and these commands apply to the object in the Main slot. The commands will display on the Toolbar as either Actions, Links, or custom toolbar buttons.	
Caption	The caption to display on the page. Here you can add field references to the data associated with the main page element so that we can get contextual information about the object in the title of the page.	

## Page Element Slot Descriptions

Slot	Description												
Main	The one and only main parent object for the console page. In the traditional scenario, this will use a MiniDetail component. It is from this single detail record that you specify the relationship to the child elements. In cases where there is no relationship between parent and child elements, you can specify any of the standard console page components.												
Side	Optionally specify the one and only control to display in the side slot. This is typically going to be a navigator page using the NavBar component. In the properties of the NavBar page element component, set the Navigator property to the name of the navigator object you want to display on the side slot.												
ConsolePage	<p>Here you can specify as many ConsolePage elements as you need to present the data. There is no physical limit to the number of ConsolePage elements you can specify on the console page, other than you should think about the use of real estate and readability for the user.</p> <p>The types of console page that can be used</p> <table> <tr> <th>Component</th><th>Description</th></tr> <tr> <td>MiniDetail</td><td>A detail view of data that requires an html fragment to layout the fields.</td></tr> <tr> <td>MiniList</td><td>A list view of the data</td></tr> <tr> <td>MiniNav</td><td>A set of navigation items from a navigator object</td></tr> <tr> <td>GMap</td><td>A google map object displaying locations from a set of latitudes and longitudes.</td></tr> <tr> <td>Graph</td><td>A google graph object displaying data into a chart format.</td></tr> </table>	Component	Description	MiniDetail	A detail view of data that requires an html fragment to layout the fields.	MiniList	A list view of the data	MiniNav	A set of navigation items from a navigator object	GMap	A google map object displaying locations from a set of latitudes and longitudes.	Graph	A google graph object displaying data into a chart format.
Component	Description												
MiniDetail	A detail view of data that requires an html fragment to layout the fields.												
MiniList	A list view of the data												
MiniNav	A set of navigation items from a navigator object												
GMap	A google map object displaying locations from a set of latitudes and longitudes.												
Graph	A google graph object displaying data into a chart format.												

Each of the page elements has a set of properties specific to the component type. In some cases the property is common to all page types. Below is a description of the properties used by each component.

## MiniDetail Properties

Property	Description
EnableRule	A JavaScript expression that will contain field references to data on the main console page element. This expression evaluates to true or false to show or hide the ConsolePage element. Typically you would only use this property for child ConsolePage elements of the console page.

Property	Description												
PaneAttributes	<p>The attributes that modify the behavior of the console page element.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>FixedAtBottom</td><td>The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.</td></tr> <tr> <td>FixedAtTop</td><td>The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.</td></tr> <tr> <td>FixedHeight</td><td>The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.</td></tr> <tr> <td>HideHeader</td><td>Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.</td></tr> <tr> <td>NoExpandCollapse</td><td>Specifies whether the user can expand or collapse a page element.</td></tr> </table>	Attribute	Description	FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.	FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.	FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.	HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.	NoExpandCollapse	Specifies whether the user can expand or collapse a page element.
Attribute	Description												
FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.												
FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.												
FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.												
HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.												
NoExpandCollapse	Specifies whether the user can expand or collapse a page element.												
Filter	Specifies any additional filtering to be applied to the mapper on this element.												
Mapper	The mapper used to retrieve data for the element.												
View	Override the view for the mapper if necessary												
ParentViewKeySource	If this MiniDetail element is on the Main slot, then you wouldn't specify the ParentViewKeySource. As a ConsolPage element, it specifies a different parent view key from the primary key of the main detail element.												
ViewKey	If this MiniDetail element is on the Main slot, then you wouldn't specify the ViewKey . As a ConsolePage element, it specifies the foreign key on the child element to the Parents primary key (or ParentViewKeySource).												
Column	Defines which column of the console pane that this page element is located. The column index is zero based.												
PaneVisibility	<p>Controls initial visibility of the page element.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>DefaultCollapsed</td><td>The pane is collapsed by default when the page is initially opened.</td></tr> <tr> <td>NoDataCollapsed</td><td>If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.</td></tr> <tr> <td>NoDataHidden</td><td>When set, the pane will be hidden when the page is initially opened if there is no data.</td></tr> </table>	Attribute	Description	DefaultCollapsed	The pane is collapsed by default when the page is initially opened.	NoDataCollapsed	If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.	NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.				
Attribute	Description												
DefaultCollapsed	The pane is collapsed by default when the page is initially opened.												
NoDataCollapsed	If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.												
NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.												
Template	Specify the name of a template that contains a fragment of html that contains layout information of the detail. This property is required for MiniDetail page elements.												
LinkKey	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.												
LinkMOP	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.												



Property	Description
NoRecordHtml	If there are no records found for this page element, then by default the string “There are no records to display” is displayed. You can set an alternative string to display with this property.

## MiniList Properties

Property	Description												
EnableRule	A JavaScript expression that will contain field references to data on the main console page element. This expression evaluates to true or false to show or hide the ConsolePage element. Typically you would only use this property for child ConsolePage elements of the console page.												
PaneAttributes	<p>The attributes that modify the behavior of the console page element.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>FixedAtBottom</td><td>The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.</td></tr> <tr> <td>FixedAtTop</td><td>The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.</td></tr> <tr> <td>FixedHeight</td><td>The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.</td></tr> <tr> <td>HideHeader</td><td>Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.</td></tr> <tr> <td>NoExpandCollapse</td><td>Specifies whether the user can expand or collapse a page element.</td></tr> </table>	Attribute	Description	FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.	FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.	FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.	HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.	NoExpandCollapse	Specifies whether the user can expand or collapse a page element.
Attribute	Description												
FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.												
FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.												
FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.												
HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.												
NoExpandCollapse	Specifies whether the user can expand or collapse a page element.												
Filter	Specifies any additional filtering to be applied to the mapper on this element.												
Sort	Specifies a custom sort on the elements mapper in addition to the current sort.												
Mapper	The mapper used to retrieve data for the element.												
View	Override the view for the mapper if necessary												
ParentViewKeySource	If this MiniDetail element is on the Main slot, then you wouldn't specify the ParentViewKeySource. As a ConsolePage element, it specifies a different parent view key from the primary key of the main detail element.												
ViewKey	If this MiniDetail element is on the Main slot, then you wouldn't specify the ViewKey. As a ConsolePage element, it specifies the foreign key on the child element to the Parents primary key (or ParentViewKeySource).												
Column	Defines which column of the console pane that this page element is located. The column index is zero based.												
PaneVisibility	<p>Controls initial visibility of the page element.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>DefaultCollapsed</td><td>The pane is collapsed by default when the page is initially opened.</td></tr> <tr> <td>NoDataCollapsed</td><td>If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.</td></tr> <tr> <td>NoDataHidden</td><td>When set, the pane will be hidden when the page is initially opened if there is no data.</td></tr> </table>	Attribute	Description	DefaultCollapsed	The pane is collapsed by default when the page is initially opened.	NoDataCollapsed	If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.	NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.				
Attribute	Description												
DefaultCollapsed	The pane is collapsed by default when the page is initially opened.												
NoDataCollapsed	If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.												
NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.												
RowsPerPage	Specifies the number of rows to be displayed in the MiniList. It's better to have a fewer number of rows than larger.												



Property	Description
LinkKey	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.
LinkMOP	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.
NewTarget	You can specify a mop with which you can navigate to a page that creates a new item on the MiniList. If this property is specified, the UI has a ">> New" link added to the element header.
NoRecordHtml	If there are no records found for this page element, then by default the string "There are no records to display" is displayed. You can set an alternative string to display with this property.

## MiniNav Properties

Property	Description												
EnableRule	A JavaScript expression that will contain field references to data on the main console page element. This expression evaluates to true or false to show or hide the ConsolePage element. Typically you would only use this property for child ConsolePage elements of the console page.												
PaneAttributes	<p>The attributes that modify the behavior of the console page element.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>FixedAtBottom</td><td>The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.</td></tr> <tr> <td>FixedAtTop</td><td>The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.</td></tr> <tr> <td>FixedHeight</td><td>The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.</td></tr> <tr> <td>HideHeader</td><td>Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.</td></tr> <tr> <td>NoExpandCollapse</td><td>Specifies whether the user can expand or collapse a page element.</td></tr> </table>	Attribute	Description	FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.	FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.	FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.	HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.	NoExpandCollapse	Specifies whether the user can expand or collapse a page element.
Attribute	Description												
FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.												
FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.												
FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.												
HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.												
NoExpandCollapse	Specifies whether the user can expand or collapse a page element.												
Column	Defines which column of the console pane that this page element is located. The column index is zero based.												
PaneVisibility	<p>Controls initial visibility of the page element.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>DefaultCollapsed</td><td>The pane is collapsed by default when the page is initially opened.</td></tr> <tr> <td>NoDataCollapsed</td><td>If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.</td></tr> <tr> <td>NoDataHidden</td><td>When set, the pane will be hidden when the page is initially opened if there is no data.</td></tr> </table>	Attribute	Description	DefaultCollapsed	The pane is collapsed by default when the page is initially opened.	NoDataCollapsed	If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.	NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.				
Attribute	Description												
DefaultCollapsed	The pane is collapsed by default when the page is initially opened.												
NoDataCollapsed	If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.												
NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.												
LinkKey	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.												
LinkMOP	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.												
Navigator	This is where you specify the navigator to display in the console pane. The navigator links automatically pick up the parent key information from the main object, so they navigate to the correct location with the appropriate parent filtering.												

## GMap Properties

The GMap handling has some special rules. The mapper that provides the location information, must have two fields that provide both latitude and longitude data. The GMap is expecting to find fields call “latitude” and “longitude” in the underlying mapper. If the fields are not called this, then you can go to the mapper object, and the User Defined Text subform. You can add two text items Latitude and Longitude and then for each text item, set the value to a field reference in the mapper that represents the latitude and longitude data.

Property	Description												
EnableRule	A JavaScript expression that will contain field references to data on the main console page element. This expression evaluates to true or false to show or hide the ConsolePage element. Typically you would only use this property for child ConsolePage elements of the console page.												
PaneAttributes	<div> <div>The attributes that modify the behavior of the console page element.</div> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>FixedAtBottom</td><td>The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.</td></tr> <tr> <td>FixedAtTop</td><td>The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.</td></tr> <tr> <td>FixedHeight</td><td>The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.</td></tr> <tr> <td>HideHeader</td><td>Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.</td></tr> <tr> <td>NoExpandCollapse</td><td>Specifies whether the user can expand or collapse a page element.</td></tr> </table> </div>	Attribute	Description	FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.	FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.	FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.	HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.	NoExpandCollapse	Specifies whether the user can expand or collapse a page element.
Attribute	Description												
FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.												
FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.												
FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.												
HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.												
NoExpandCollapse	Specifies whether the user can expand or collapse a page element.												
Filter	Specifies any additional filtering to be applied to the mapper on this element.												
Sort	Specifies a custom sort on the elements mapper in addition to the current sort.												
Mapper	The mapper used to retrieve data for the element.												
View	Override the view for the mapper if necessary												
ParentViewKeySource	If this MiniDetail element is on the Main slot, then you wouldn't specify the ParentViewKeySource. As a ConsolePage element, it specifies a different parent view key from the primary key of the main detail element.												
ViewKey	If this MiniDetail element is on the Main slot, then you wouldn't specify the ViewKey. As a ConsolePage element, it specifies the foreign key on the child element to the Parents primary key (or ParentViewKeySource).												
Column	Defines which column of the console pane that this page element is located. The column index is zero based.												



Property	Description								
PaneVisibility	Controls initial visibility of the page element.								
	<table><tr><th>Attribute</th><th>Description</th></tr><tr><td>DefaultCollapsed</td><td>The pane is collapsed by default when the page is initially opened.</td></tr><tr><td>NoDataCollapsed</td><td>If there is no data to display in the console pane, you would see a message “No Data to display”, or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.</td></tr><tr><td>NoDataHidden</td><td>When set, the pane will be hidden when the page is initially opened if there is no data.</td></tr></table>	Attribute	Description	DefaultCollapsed	The pane is collapsed by default when the page is initially opened.	NoDataCollapsed	If there is no data to display in the console pane, you would see a message “No Data to display”, or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.	NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.
	Attribute	Description							
	DefaultCollapsed	The pane is collapsed by default when the page is initially opened.							
	NoDataCollapsed	If there is no data to display in the console pane, you would see a message “No Data to display”, or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.							
NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.								
LinkKey	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.								
LinkMOP	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.								
DetailTarget	The map displays the location defined by the record. If you drill down into that location, you are taken to the MOP specified by this property.								
ZoomLevel	The default zoom level for the map at which to display the locations. The zoom level range is between 1 and 18, with 18 being the most zoomed in.								

## Graph Properties

Property	Description												
EnableRule	A JavaScript expression that will contain field references to data on the main console page element. This expression evaluates to true or false to show or hide the ConsolePage element. Typically you would only use this property for child ConsolePage elements of the console page.												
PaneAttributes	<p>The attributes that modify the behavior of the console page element.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>FixedAtBottom</td><td>The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.</td></tr> <tr> <td>FixedAtTop</td><td>The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.</td></tr> <tr> <td>FixedHeight</td><td>The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.</td></tr> <tr> <td>HideHeader</td><td>Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.</td></tr> <tr> <td>NoExpandCollapse</td><td>Specifies whether the user can expand or collapse a page element.</td></tr> </table>	Attribute	Description	FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.	FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.	FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.	HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.	NoExpandCollapse	Specifies whether the user can expand or collapse a page element.
Attribute	Description												
FixedAtBottom	The pane should be positioned at the bottom of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the bottom in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page.												
FixedAtTop	The pane should be positioned at the top of the console and fixed in place. If there is more than one pane specified with this attributes, then the panes will be fixed at the top in order specified by the order property of the page element. Additionally, when specified as fixed at bottom, the page element spans the width of the page. Note that when FixedAtTop, the page elements are placed above the position of the main page element.												
FixedHeight	The pane should have a fixed height, rather than resizing per the pane contents. The specific height is set in the MaxHeight property.												
HideHeader	Determines whether the console pane header should be hidden, or not. When hidden, the console pane element is displayed, automatically expanded. The UI element that provides expand and collapse capability is not available. Neither are any page element links.												
NoExpandCollapse	Specifies whether the user can expand or collapse a page element.												
Filter	Specifies any additional filtering to be applied to the mapper on this element.												
Sort	Specifies a custom sort on the elements mapper in addition to the current sort.												
Mapper	The mapper used to retrieve data for the element.												
View	Override the view for the mapper if necessary												
ParentViewKeySource	If this MiniDetail element is on the Main slot, then you wouldn't specify the ParentViewKeySource. As a ConsolePage element, it specifies a different parent view key from the primary key of the main detail element.												
ViewKey	If this MiniDetail element is on the Main slot, then you wouldn't specify the ViewKey. As a ConsolePage element, it specifies the foreign key on the child element to the Parents primary key (or ParentViewKeySource).												
Column	Defines which column of the console pane that this page element is located. The column index is zero based.												
PaneVisibility	<p>Controls initial visibility of the page element.</p> <table> <tr> <th>Attribute</th><th>Description</th></tr> <tr> <td>DefaultCollapsed</td><td>The pane is collapsed by default when the page is initially opened.</td></tr> <tr> <td>NoDataCollapsed</td><td>If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.</td></tr> <tr> <td>NoDataHidden</td><td>When set, the pane will be hidden when the page is initially opened if there is no data.</td></tr> </table>	Attribute	Description	DefaultCollapsed	The pane is collapsed by default when the page is initially opened.	NoDataCollapsed	If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.	NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.				
Attribute	Description												
DefaultCollapsed	The pane is collapsed by default when the page is initially opened.												
NoDataCollapsed	If there is no data to display in the console pane, you would see a message "No Data to display", or the text specified in the NoRecordHtml property. When set, the pane will be collapsed closed when the page is initially opened if there is no data.												
NoDataHidden	When set, the pane will be hidden when the page is initially opened if there is no data.												
LinkKey	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.												

Property	Description
LinkMOP	Used with the LinkMOP property. If the page element is displaying a header, then the caption of the header will be a link to the page specified in the LinkMOP property. Specify the LinkKey to determine the pk for the navigation.
NoRecordHtml	If there are no records found for this page element, then by default the string "There are no records to display" is displayed. You can set an alternative string to display with this property.
BarWidth	Specify the width of bars in a vertical or horizontal bar chart. The format of the data follows the google chart specification. <a href="http://code.google.com/apis/chart/docs/gallery/bar_charts.html#chbh">http://code.google.com/apis/chart/docs/gallery/bar_charts.html#chbh</a>
ChartSize	Specify the dimensions of the chart. If not specified, the chart is sized to 200x125 (px). The size is specified as a string in the format "widthxheight" <a href="http://code.google.com/apis/chart/docs/chart_params.html#gcharts_chs">http://code.google.com/apis/chart/docs/chart_params.html#gcharts_chs</a>
ChartType	The type of chart required. Only a small subset of the available charts are available. We support Line, Pie, 3D Pie, Vertical Bar, Horizontal Bar <a href="http://code.google.com/apis/chart/docs/chart_params.html#gcharts cht">http://code.google.com/apis/chart/docs/chart_params.html#gcharts cht</a>
Colors	The color values for the chart series data. <a href="http://code.google.com/apis/chart/docs/chart_params.html#gcharts_series_color">http://code.google.com/apis/chart/docs/chart_params.html#gcharts_series_color</a> Each entry is a string in RGB format <a href="http://code.google.com/apis/chart/docs/chart_params.html#gcharts_rgb">http://code.google.com/apis/chart/docs/chart_params.html#gcharts_rgb</a> If the chart displays a legend, then the legend also matches the color.
FormatY	Specifies the number format for the y-axis data. The format uses the standard .Net number format specifications. Internally, the platform converts this data into the appropriate Google chart label format specification combined with the ScaleY format to generate a Y-axis labels
Gridlines	Specifies whether gridlines should be displayed on the chart. The options are true and false. The platform generates a Google chart compliant format string for the chg parameter with the following settings, "20,20,1,5" <a href="http://code.google.com/apis/chart/docs/chart_params.html#gcharts_grid_lines">http://code.google.com/apis/chart/docs/chart_params.html#gcharts_grid_lines</a>
Legend	Specifies whether a legend should be displayed on the chart. You specify a semi-colon delimited list of legend text items. The platform converts this list into a Google chart compliant pipe separated list . <a href="http://code.google.com/apis/chart/docs/chart_params.html#gcharts_legend">http://code.google.com/apis/chart/docs/chart_params.html#gcharts_legend</a>
ScaleY	Specifies the scale of the y-axis data. The value expects an integer value and has a default value of 1. Internally, the platform converts this data into the appropriate Google chart label format specification combined with the FormatY format to generate a Y-axis labels
CategoryAxis	Specifies which field in the mapper provides the data specifying the chart categories. This field is required for a chart specification.
ValueAxis	Specifies which field(s) in the mapper provides the data specifying the chart values. At least one field is required to generate a valid chart. You can specify multiple value fields in a semi-colon separated list. Internally, the values are converted into the SimpleEncoding mechanism, limiting the data to one of 61 encoding values. <a href="http://code.google.com/apis/chart/docs/data_formats.html#simple">http://code.google.com/apis/chart/docs/data_formats.html#simple</a> Simple formatting is used to minimize the length of the posted querystring for large datasets.
ValueRoundTo	Specifies the rounding applied to value data. The default value is 1