

English Heritage Software Development Centre

HERITAGE GATEWAY

Resource Provider Toolkit (v1.4)

Authors: Matt Colman & Rob Bowen

Addendum: Darren Collins

Contents

1. Background	5
1.1 Purpose of this document	5
1.2 Introduction	5
2.1 Overview	6
3. Search Message Schema	7
3.1 Overview	7
3.2 request/userInfo	7
3.3 Search information (request/entities & request/query)	8
3.3.1 Spatial location searching	8
3.3.2 Textual location searching	9
3.3.3 Thesaurus item lookup	9
3.3.3.1 Searching using Grade (Listed Building searches only)	11
3.3.4 Temporal Searching	11
3.3.5 Temporal Searching with Periods	11
3.3.5.1 Temporal Searching with Listing Date (Listed Building searches only) .	12
3.3.6 Person searching	12
3.3.7 County, District & Parish lookup	12
3.3.8 Other Information	13
4. Creating the Resource File	13
4.1 Overview	13
4.2 Resource File Schema	14
display	14
access	15
search_params	15
xslt	16
WMSAttributes	16
5. Web Service Responses	18
5.1 Overview	18
5.2 Result Meta Information	18
6.1 Examples provided	20
6. Web Service Implementation	21
6.1 Summary	21
6.2 Setting Up Toolkit	21
6.3 Configuring MapServer for Mapping	21
6.3.2 Monuments.map Configuration file	22

6.3.3 Layer.config Configuration File	22
6.3.3 Database Components.....	23
6.4 Setting up Web.Config file	26
connectionStrings.....	26
appSettings	26
6.5 Toolkit classes	28
Generic Data Access.....	28
IGatewayWebService.....	29
Search Message	30
Summary Result.....	32
CacheHandler	32
ToolKitConfiguration.....	33
StylesheetHelper	34
StyleSheetVersion.....	35
GatewayDate	35
MessageHandler	35
ResultsCollection	37
6.6 Filling in skeleton implementation	38
GatewaySearchHandler: SearchHandler.....	39
GatewaySummaryResult: SummaryResult.....	40
GatewaySingleResult.....	40
GatewayResultsCollection: ResultsCollection	41
GatewayService: IGatewayService	42
6.7 Default Program Logic	48
GetTopRecords/ GetSummaryResults	48
Diagram 1	48
Diagram 2	49
Diagram 3	50
Diagram 4	50
Diagram 5	51
GetFullSingle().....	51
Diagram 6	52
Diagram 7	53
Diagram 8	54
Appendix 1a	55

1. Background

1.1 Purpose of this document

This document is aimed at heritage dataset providers wishing to share their information on the Heritage Gateway. This guide provides a step by step guide for creating a compatible web service.

1.2 Introduction

The *Heritage Gateway* offers a search interface for querying multiple remote databases. After exploring a number of possibilities a Web Service based solution was chosen.

This solution provides a highly scalable and cost effective architecture that will be able to cope with current and future demands.

Search interfaces have been created aimed at both the general public and heritage professionals. The search results will give the user a clear picture of exactly what information is available on the queried subject from all of the attached databases. Users will be able to navigate the results through a consistent intuitive interface and will have the option to 'drill down' to single records.

2. Toolkit Overview

2.1 Overview

This toolkit should be utilised by heritage dataset providers wishing to share their information on the Heritage Gateway. All aspects of what is required to create a web service that conforms to the Heritage Gateway protocol have been covered.

The toolkit (distributed as a zip file has the following structure):

- Toolkit.doc (this document)
- **XMLSchemas** (directory)
 - base_resource.xsd
 - cdp.xsd
 - query.xsd
- **XML** (directory)
 - cdp.xml
 - com.xml
 - periods.xml
 - ranges.xml
- **WebService** (directory)
 - GatewayWebserviceTemplate.zip
- **Examples** (directory)
 - imagebased.xml
 - imagebased.xsl
 - textbased.xml
 - textbased.xsl
- **MapServer** (directory)
 - MapServer.zip

3. Search Message Schema

3.1 Overview

The search message created by the Heritage Gateway for use by the web services will always adhere to the schema provided as part of this toolkit (XMLSchemas/query.xsd).

A copy of the complete schema can be found in Appendix 1a (both a graphical and textual version is provided).

This schema is split into two major sections; user information (see 3.2) and search information (see 3.3).

3.2 request/userInfo

This section holds information about the user requesting the information from the web service including:

- **source_app** – The name of the application sending the request ('Heritage Gateway' in this instance). A global unique identifier is also provided as an attribute source_app_uid.
- **passport_uid** – English Heritage is currently implementing a passport system. If the user is registered then their passport UID will be passed here.
- **user_level** – Passes the level of the user creating the request, this can be used determine if content should be provided. Currently there are 2 valid values:
 - 0 – Unregistered user.
 - 1 – User is registered on the Heritage Gateway.
- **browserInfo** - Contains information relating to the users web browser
 - urlReferrer – The URL that create the request.
 - Browser – String containing the users browser details.
 - userLanguage – code describing the users Language settings.
- **visitInfo**
 - ip – IP address of the request originiator
 - timeRequested – The time the request was submitted.

Given below is a sample userInfo section as it would be received by the web service:

```
<userInfo>
  <source_app source_app_uid="1">Heritage Gateway</source_app>
  <passport_uid />
  <user_level>1</user_level>
```

```
<browserInfo>
  <urlReferrer>/Gateway/Homepage_Search.aspx</urlReferrer>
  <browser>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
1.1.4322; .NET CLR 2.0.50727)</browser>
  <userLanguages>en-gb</userLanguages>
</browserInfo>
<visitInfo timeRequested="24/11/2006 13:10:40">
  <ip>127.0.0.1</ip>
</visitInfo>
</userInfo>
```

This section is provided to allow content providers to see who is requesting their information (and log it if necessary). This information can also be used to determine whether or not information should be served.

3.3 Search information (request/entities & request/query)

An entities section has been included to allow for different types of entities to be searched for. Initially only monument information will be searched, however in the future multiple entity types may be passed here. The search message will contain the following structure:

```
<entities>
  <entity>Monument</entity>
</entities>
```

The query section contains the details of how the user has completed the search form. As with the forms themselves, this xml document is split into the 4 major sections of Where, What, When and Who.

3.3.1 Spatial location searching

Spatial search locations are currently obtained through an ESRI JavaScript API maps interface. The search message will contain a point specified in:

- Latitude & Longitude
- Literal Grid Reference (12 figure)
- Ordnance Survey grid reference (6 figure)

Any of the 3 options listed can be used to query your data. A range will also be specified by the user the options for which are:

- 100m
- 250m
- 500m
- 1km (1000m)
- 2km (2000m)
- 5km (5000m)
- 10km (10000m)

These can be found in the toolkit in the xml file named 'XML/ranges.xml'. Ranges will always be passed in metres in the search message. The following extract shows how a spatial search location would appear in a search message.

```
<where>
  <spatial>
    <gridref>414299185015</gridref>
    <osgridref>SU142850</osgridref>
    <latitude>51.5634</latitude>
    <longitude>-1.7937</longitude>
    <range>500</range>
  </spatial>
</where>
```

3.3.2 Textual location searching

Locations can also be specified in two textual forms. The first of these is County, District and Parish. For more information utilising this functionality please see the section titled 'County, District & Parish lookup' (3.3.7).

The second option is a free text place name search. This is included in the schema under where/freetext_where. Values passed here should be interpreted in the way that best relates to your data.

Ideally both spatial and textual searching should be implemented. However where datasets do not contain spatial information, textual searching is acceptable.

3.3.3 Thesaurus item lookup

On the basic search the user has the ability to select thesaurus terms from the top level of the MIDAS monuments types thesaurus only. On the advanced search a tree view style control is used to enable users to drill down to all levels of any of the thesauri on offer. Currently the following MIDAS thesauri are included on the advanced search form:

- Monument
- Object
- Evidence

The thesaurus items (request/query/what/thesaurus_items) section of the query xml schema is used to pass the selected terms:

```
<thesaurus_items>
  <term uid="90842" value="Lighthouse" />
```



```
</thesaurus_items>
```

Both the Basic and Advanced searches contain a free text box for specifying monument types. This can be found at 'request/query/what/freetext_what' in the query schema. Obviously a UID will not be forwarded with this term.

As part of the toolkit you have been provided with an xml document 'XML/com.xml', containing all the thesaurus terms available. If broad terms are sent in the search message then you should query all narrow terms beneath it. This file may be updated as new thesauri are added to the Gateway search.

The xml file has the following structure:

```
<?xml version="1.0" encoding="utf-8"?>
<thesauri>
  <monument_types>
    <term l="1" uid="134003" name="Agriculture And Subsistence">
      <term l="2" uid="68544" name="Agricultural Building">
        <term l="3" uid="68573" name="Apiary" />
      </term>
    </term>
  </monument_types>
</thesauri>
```

The term attribute 'l' specifies the level at which that term exists within the current thesaurus. This has been provided to speed access to different levels via XPath.

Both the name and the uid of the term will always be passed to avoid problems caused by duplicate terms at different locations in the thesauri.

The .net based GatewayWebService template included in the toolkit contains an implementation of the broad/narrow thesaurus term searching for guidance.

A 'specific type' free text option is also provided to allow the user to enter thesaurus terms directly n.b. items passed here will not have a uid. If completed the search message will contain the value in:

```
request/query/what/specific_type
```

3.3.3.1 Searching using Grade (Listed Building searches only)

If your resource provides information about listed buildings and you hold the listing grade of the buildings then you can search using the grade passed. The listing grade is an enumerated type and will be passed as one of the following:

- **Grades: I, II*, II, A, B, C, DC, NG**

Note that you will need to indicate that your resource is capable of dealing with listed building searches in your resource xml file (see 4.2 search_params section). The listing date is passed in the query schema as follows:

```
<what>  
  <grade>II</grade>  
</what>
```

3.3.4 Temporal Searching

The user has the ability to specify a 'from' date and a 'to' date. Both are defined in years and whether the years are a.d. or b.c. can be specified.

The dates are passed in the 'when' part of the schema:

```
<when>  
  <year_from adbc="ad" year="1854" />  
  <year_to adbc="ad" year="1879" />  
</when>
```

3.3.5 Temporal Searching with Periods

A simple period lookup has been implemented. This is designed mainly for use with English Heritage datasets however should you wish to implement period searching as well as date searching the files 'XML/periods.xml' (containing broad periods used on the basic search form) and 'XML/periods_full.xml' (a complete list of periods) have been included in the toolkit.

Please note that when the user specifies a period on search forms, the date fields are automatically completed so will always be passed.

As above, the period is passed in the 'when' part of the search schema:

```
<when>  
  <period>Iron Age</period>  
  <year_from adbc="bc" year="800" />  
  <year_to adbc="ad" year="43" />  
</when>
```

3.3.5.1 Temporal Searching with Listing Date (Listed Building searches only)

If your resource contains information about listed buildings and you hold the date of listing then you can use the listing date elements. Note that you will need to indicate that your resource is capable of dealing with listing date searches in your resource xml file (see 4.2 search_params section). The listing date is passed in the query schema as follows:

```
<when>
  <listing_date>
    <listing_year>1989</listing_year>
    <listing_month>10</listing_month>
  </listing_date>
</when>
```

3.3.6 Person searching

They 'who' section has been included to allow for people referenced in the data to be searched for.

Only one field is passed here and that is the surname. This may related to either the author/creator of the content or the subject of the content, or both. This decision should be made based on your specific data.

3.3.7 County, District & Parish lookup

County, District and Parish searching is implemented on the Advanced search page and contains a free text box accepting a County name, District name or Parish name. A file named 'cdp.xml' which contains all the CDP information used on the Gateway searches has been included in the toolkit. A Schema named 'cdp.xsd' is also provided. This was previously implemented with drop down boxes for County, District and Parish.

The CDP xml file has the following structure:

```
<?xml version="1.0" encoding="utf-8"?>
<cdp_lookup>
  <county uid="90842" name="Avon">
    <district uid="685386" county="Avon" name="Bath And North East Somerset">
      <parish district="Bath And North East Somerset" uid="90844" name="Bath" />
    </district>
  </county>
</cdp_lookup>
```

Given below are some useful XPath strings for use with this file:

To extract all counties:

```
//county[@name]
```

To extract all districts for a particular county:

```
//district[@county='countyName']
```

To extract all parish's for a particular district:

```
//parish[@district='districtName']
```

3.3.8 Other Information

Various other fields of information can be passed in the 'other' element of the query schema:

Homepage/Single Text Box Searches:

Homepage searches contain the string entered on the homepage search form along with the search type selected:

- **Exact:** match the exact phrase entered (exact).
- **Any Order:** match all the words entered but in no particular order (all).
- **Any Words:** match any of the words entered (any).

```
<other>  
  <homepage_search>  
    <search_text>SearchString</listing_year>  
    <search_type>exact|all|any</listing_month>  
  </homepage_search>  
</other>
```

Specific Resource UID Searches:

You may also choose to implement UID searching for your resource. The UID search is available on the 'Resources' tab of the advanced search form and allows the user to pass a uid string to a particular resource. Note that you will need to indicate that your resource supports uid searching by adding the required element in your resource xml file. The uid search item is passing in the following manner:

```
<other>  
  <uid_search>ABC1234</uid_search>  
</other>
```

4. Creating the Resource File

4.1 Overview

An XML based resource file is used to describe the details of your web service. This document holds all the information the Heritage Gateway requires to address your

web service and to correctly format results produced. Geographic information is included (optional) to avoid searching of datasets that do not relate to the area being searched. Temporal information can also be included (optional) to avoid searching datasets where information is only available for certain periods.

4.2 Resource File Schema

The schema for this document has been included as part of the toolkit ('XMLSchemas/base_resource.xsd'). A blank base resource xml has been included ('Examples/base_resource.xml') ready for you to populate with information relating to your resource.

The Schema can also be seen in both graphical and textual forms in appendix 1b.

The resource file contains the following sections:

display

This section contains all the information used to display your resource details on the Heritage Gateway:

- **title** – the title of your resource (e.g. PastScape).
- **provider_image** – resource providers image (e.g. englishheritage.gif).
n.b. this image should be no more than 30 pixels high.
- **provider_alt** – the alt text for the resource providers image.
- **image_url** – the resource logo image (e.g. pastscape.gif).
n.b. this image should be no more than 30 pixels high.
- **image_alt** – the alt text for the resource image.
- **description_short** - a short text based description of your resource (max 250 chars).
- **description_long** – a more concise description of your resource.
- **top_level_records** – the number of records to be displayed on the top level results screen.
- **application_records** – the number of records to display on the application screen.
- **css_url** – a link to either a local or remote style sheet to be used to style your results pages.

- Resource specific classes should be used (e.g. .pastscape_title) as to avoid conflicts with the Heritage Gateway css classes.
- **group** – An integer representing the resource group that your resource belongs to:
 - 1 - Statutory Data
 - 2 - Local Records
 - 3 - National Archaeological Data
 - 4 - National Image Collections
- **contact_link** – A link or email address which will be displayed as a ‘Feedback on Record’ link at the base of any single result screens.
 - **email:** mailto:abc.cba@acb.com
 - note that the mail message will automatically have a subject line appended containing the uid of the users current record.
 - **url:** http://www.mywebsite.com/feedbackform?recorduid={uid}
 - the text {uid} in the url string will be automatically replaced by the uid of the record currently being viewed.
- **isWMSLayer** – A Boolean value representing whether mapping has been implemented for the resource or not.

access

This section contains the information that allows the heritage gateway to communicate with your web service and online website (if available).

- **webservice_url** – the complete url to your web service.
- **application_url** – the complete url to your online application (if available).

search_params

This section contains geographic and temporal boundaries covered by your data. Searches performed by the user that fall outside the boundaries laid down here will not invoke your web service.

- **related_counties** – a text based list of the counties that relate to your dataset.
n.b. the toolkit contains a file named cdp.xml which contains all of the counties that may be used in a query by the Heritage Gateway.

- **spatial** – a bounding box that specifies the area covered by your dataset using literal grid references
- **temporal** – a date from and date to that set the bounds of the temporal information contained in your dataset.
- **listed_building_info** – Boolean indicating that the resource contains listed building information and has been implemented to use listing_date and grade elements of the query schema. If enabled, searches containing listed building fields will be sent to the resource – else the resource will be ignored for such searches.
- **uid_search_enabled** – Boolean indicating that the resource implements uid searching. Once enabled the resource will appear in the list of ‘Resources supporting Reference No searching’ on the advanced search form (resources tab). UID searches are only submitted to the selected resource.

xslt

This section is simply used to specify the name of the xsl file that should be used to style the results produced by your web service. Currently only one xslt file can be utilised but the xsd has been structured in such a way to allow multiple xsl files to be used if the future if needed – for this reason the id should always be set to 1 at the moment.

- **record_type** – contains 2 attributes one for the filename of your xsl file and one for the uid (should be set to 1).

WMSAttributes

This section contains the required information for allowing resource results to be displayed on a map on the results page

- **layerTitle** – The name of your resource. This will be the name that is shown in the map table of contents
- **path** – The virtual directory of your MapServer executable and the physical path of your configuration file for MapServer in the following format: -
<http://servername/WebServiceDirectory/Mapserver/scripts5.6/mapserver.exe?map=C:\\Inetpub\\wwwroot\\WebServiceDirectory\\Mapserver\\map\\yourfilename.map&sessionID=%sessionID%>
- **layerNamesSelection** – A comma delimited list containing your layer names as defined in the MapServer configuration file. These will be the layers that

will be displayed on the results map with all of the other resource results.

layerNamesSingle – A comma delimited list containing your layer names as defined in the MapServer configuration file. These will be the layers that will be displayed on the map within a single record (this is not currently implemented)

- **layerNamesAll** – A comma delimited list containing your layer names as defined in the MapServer configuration file. These will be the layers that will be displayed when viewing all results from your resource.
- **Version** – The version of WMS being used. E.g. 1.3.0
- **Wkid** – The well-known id being used for the spatial reference. E.g. 27700

5. Web Service Responses

5.1 Overview

In this section the requirements for the messages returned by web services providing content to the Heritage Gateway will be looked at in depth.

The message sent from the Heritage Gateway (formatted using the Search Schema) should be used to query your database in whichever way you deem to be the best. Once a result set is available, the Heritage Gateway requires the results be returned in one of 2 ways:

- overview format (used for the top level results and application level results)
- full single result format (used to transmit 1 MIDAS formatted record)

The following web methods require results to be returned in the overview format:

- GetTopRecords
- GetSummaryResults

And the full single result format should be returned when the following web methods are used:

- GetFullSingle

These are looked at in more depth in the section 6 (web service functionality).

5.2 Result Meta Information

All result information sent by your web service should contain a meta section. The purpose of this section is to hold information about the resource, general information about the results and details of which XSLT should be used to format the message.

The meta section should include the following:

- **ResourceID** – your unique resource ID passed in the search message
- **SessionID** – the sessionID passed in the search message

- **BBOX** – The bounding box containing all of the results that are being returned by the search in the format of xmin, ymin, xmax, ymax
- **TotalRecords** – the total number of records returned
- **NumberOfRecordsReturned** – the number of results returned in the current message
- **SourceSite** – the name of providing application
- **Stylesheet** – the section of the XSL stylesheet to apply to the results
 - TopLevel
 - Application
 - SingleResult
- **Version** – the version number of the stylesheet
- **UserType** – uid reflecting the type of user the message is being sent to
- **FirstRec** – the index of the first result being returned
- **LastRec** – the index of the last result being returned

An XML Schema results.xsd has been included in the Schemas folder of the toolkit. This shows how result messages should be returned to the Heritage Gateway for display.

6. Styling results with XSLT

6.1 Examples provided

The XML based results returned by your web service to the Heritage Gateway will be formatted using XSLT before being displayed to the user. The XSL file used will be specific to your resource, enabling you to display results to the user in the way that best suits your data.

A single XSL file contains the XSL transform code for the 3 levels of results returned by your web service:

- Top Level Results – TopLevel
- Application Level Results - Application
- Single Result (full record view) - SingleResult

The toolkit contains a base XSL file (named 'Examples/base.xsl') that contains all the essential sections required for results formatting. You will note that a variable is created named 'type' which reads the name of the style sheet from the meta information passed in the web service response (see 5.2). As the XSL is passed in a single file it is vital that you utilise this.

Two additional XSL files have been included as part of the toolkit, 'Examples/imagebased.xsl' and 'Examples/textbased.xsl'. The first is a simple example of an image based resource and the second of a mainly text based resource. These are included only as a guide and will need to be modified to work with the XML your web service produces.

The XSL file to be used by your web service is defined in the XSLT section of your XML based resource file. For more information on this please see the section entitled "Creating your resource file" (Section 4).

As the layout or content that your web service produces is likely to change over time, a process to allow updating of the XSLT has been implemented.

6. Web Service Implementation

6.1 Summary

The web service toolkit has been developed in order to speed up the creation of new .net web services allowing the Heritage Gateway to search remote datasets.

The toolkit has been created using C# 2005

6.2 Setting Up Toolkit

The best way of using the toolkit is to set it up in Visual Studio as a new Web Site Template. To set this up you should:

1. Copy the zipped GatewayWebserviceTemplate folder into your templates directory. This location can be identified by clicking Tool – Options, and then looking in the general section of the Projects and Solution node in the treeview. The default location of this directory is:
MyDocuments\Visual Studio 2005\Templates\Visual Web Developer\
2. Once the template is copied into the correct location it should appear under My Templates in the New Web Site dialog.

6.3 Configuring MapServer for Mapping

Included within GatewayWebserviceTemplate folder is a folder called Mapserver and this contains all of the required elements to generate a Web Map Service layer that will be loaded into the map within the Heritage Gateway site. The following detail is valid for Windows Server deployments and Microsoft SQL Server databases.

6.3.1 Internet Information Services (IIS) Configuration

The Mapserver folder that is included within the GatewayWebserviceTemplate folder will need to be configured as a Virtual Directory in IIS. It will need to be created with Read/Write properties and have Execution Permissions of Scripts and Executables. Authentication and access control should be set to Anonymous Access only.

Due to Mapserver being an executable, some rules need to be set to allow this executable to run.

1. Navigate to IIS Manager => Web Service Extensions.
2. From here, right click in the right hand pane and right click to 'Add a new Web Service extension...'.
3. Specify the Extension Name (mapserv.exe) and under Required Files navigate to 'your web service location\Mapserver\scripts5.6\mapserv.exe', then click add.
4. Tick 'Set extension status to Allowed' and then click OK.

This will then allow mapserv.exe to run when requested.

6.3.2 Monuments.map Configuration file

The monuments.map file is located within 'your web service location/Mapserver/map' and is one of two configuration files used by Mapserver. This file contains various details including: -

- Log file locations
- Symbology
- Image Qualities

There are a number of locations within this file that refer to 'your web service physical address'. This will need to be replaced with the location where the web service is configured in IIS. For example, c:\inetpub\wwwroot\HERWebService\Mapserver\... Where it refers to 'your web service virtual path', it is simply the <http://servername/HERWebService/MapServer/>... for example.

6.3.3 Layer.config Configuration File

Layer.config is referenced in Monuments.map as an included file. This is the configuration file where sensitive details such as database connection user names and passwords will be present and as such, it being included as a config file will mean that the details remain private.




The Layer.config file is where the 'layers' are created that will make up the WMS image that is loaded into the Heritage Gateway Results map. The Layer.config file that has been made available has three 'layers' added as examples which relate to the <layerNamesSelection>, <layerNamesAll> and <layerNamesSingle> parts of the resource configuration file as described in section 4.2. The Layer.config file contains instructions on what the various attributes need to be configured as.

An Open Database Connectivity connection (ODBC) will be required to access the database holding the data. This will need to be created by accessing Administrative Tools => Data Sources (ODBC). Once inside the ODBC Data Source Administrator, move to the System DSN tab and create a new ODBC connection. Please note the name of the connection along with the username and password used as this will be required to be added into the Layer.config file.

6.3.4 Database Components

There are a number of database components needed by Mapserver to allow the results returned by the Web service to be mapped. In the GatewayService.cs class, you will see an example of how the results returned will need to be stored in a new table on the database. The table will hold a copy of all of the matched results relating to a search. These are required to be stored as this information will need to be read many times by Mapserver. See 6.3.3.1 for an example table definition.

6.3.3.1 Table Definition (SelectedRecordsForSession)

	Column Name	Data Type	Allow Nulls
	SessionID	nvarchar(100)	<input type="checkbox"/>
	Entity	nvarchar(50)	<input type="checkbox"/>
	UID	int	<input type="checkbox"/>
	SessionDateTime	datetime	<input type="checkbox"/>

The SessionID relates to the Session ID that has been provided in the request from the Heritage Gateway. The SessionID will be a Globally Unique Identifier (GUID). Entity relates to a short textual description of the record type. UID relates to the unique ID of the record being sent back to the Heritage Gateway. The SessionDateTime is a timestamp of when the record was inserted into this table. The SessionDateTime column is something that can be used to maintain this table. Data can be removed when it becomes older than 2 hours for example.

6.3.3.2 Selected Records View Definition (dbo.GetSelectedResults)

As we have specified three types of 'layer' in our MapServer configuration file, three database views will need to be created that will be used by MapServer. The first is a view that will return the records that have been returned as part of this selection. An example of this view can be found below. MapServer requires that a column called

WKTPoint in the format of POINT(X Y), X and Y both being 6 digit integers and separated with a space, is returned by this view. The view will join the table described above to tables containing attribute data about the record. The column names of this view need to tally with the Mapserver 'Layer.config' file, as does the view name.

Example

```
CREATE VIEW [dbo].[GetSelectedRecords]
AS
SELECT    Name,
          ID,
          Description,
          'POINT (' + CAST(ROUND(X,0) AS VARCHAR(10)) + ' ' +
          CAST(ROUND(Y,0) AS VARCHAR(10)) + ')' AS WKTPoint,
          ROUND(X,0) AS XMIN,
          ROUND(Y,0) AS YMIN,
          ROUND(X,0) AS XMAX,
          ROUND(Y,0) AS YMAX,
          dbo.SelectedRecordsForSession.SessionID
FROM      dbo.RecordData INNER JOIN dbo.SelectedRecordsForSession ON
          dbo.RecordData.ID = dbo.SelectedRecordsForSession.UID

GO
```

The SessionUid column is important because this is the column that MapServer requires so that a 'where' clause definition can be added.

6.3.3.3 All Records View Definition (dbo.GetAllRecords)

The 'all records' view definition is very similar to the selected records view and will be used when viewing all records from a single resource rather than viewing results from all resources and, like the selected record view, will be required to present a column called WKTPoint in the same format as above.

Example

```
CREATE VIEW [dbo].[GetAllRecords]
AS
SELECT    Name,
          ID,
          Description,
          'POINT (' + CAST(ROUND(X,0) AS VARCHAR(10)) + ' ' +
          CAST(ROUND(Y,0) AS VARCHAR(10)) + ')' AS WKTPoint,
          ROUND(X,0) AS XMIN,
          ROUND(Y,0) AS YMIN,
          ROUND(X,0) AS XMAX,
          ROUND(Y,0) AS YMAX,
          dbo.SelectedRecordsForSession.SessionID
FROM      dbo.RecordData INNER JOIN dbo.SelectedRecordsForSession ON
          dbo.RecordData.ID = dbo.SelectedRecordsForSession.UID

GO
```

6.3.3.4 Single Record View Definition (dbo.GetSingleRecord)

The single record view is something that is not implemented at the minute, but will be used in the future when mapping of single records is implemented. The UID column is important here as this will be used by MapServer to restrict the data for just one record.

Example

```
CREATE VIEW [dbo].[GetSingleRecord]
AS
SELECT ID,
       Record_UID AS UID,
       'POINT (' + CAST(X AS VARCHAR(10)) + ' ' + CAST(Y AS
VARCHAR(10)) + ' )' AS WKTPoint,
       X AS XMIN,
       Y AS YMIN,
       X AS XMAX,
       Y AS YMAX
       substring(pastscape.WEBDATA.Description, 0, 50) + '...' as
Description
FROM   dbo.RecordData

GO
```

The UID column will need to be added as a filter within the Single Record section of the Layer.Config and it will look like this: -
FILTER "WHERE UID = '%UID%'"

6.4 Setting up Web.Config file

To set up the web service toolkit, you will need to make the following changes to the web.config file.

connectionStrings

1) Comment out one ServiceData connection strings depending on the database in use. Enter valid connection string details in the remaining connection string attribute e.g. if you are using a SQL server database, the OracleClient connection string should be removed.

```
<connectionStrings>
    <add name="ServiceData" connectionString="*** SET YOUR
CONNECTION STRING HERE" providerName="System.Data.OracleClient"/>
    <add name="ServiceData" connectionString="*** SET YOUR
CONNECTION STRING HERE" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

appSettings

The following things should also be set in the appSettings section of the web.config file. Failing to have these values set can cause errors to occur.

1) Ensure the Thesaurus Path points the location of the midas XML Thesaurus. Default setting is ok if you have not moved the documents position.

```
<add key="ThesaurusPath" value="~/Thesaurus/com.xml"/>
```

2) Set the name of the source site or DataSet here e.g. Viewfinder

```
<add key="SourceSiteName" value="*** Set the name of the source site
here ***"/>
```

3) Set version number of the XsltTemplate you have available e.g. for the first version use 1.

```
<add key="StylesheetVersion" value="1"/>
```

4) Set path of the XSLT template that should be used to format the results. Default location is fine if using the pre-created files.

```
<add key="StylesheetPath" value="~/XSLT/XsltTemplate.xsl" />
```

5) Number of minutes that results should be cached for.

```
<add key="CacheDuration" value="5" />
```

```
<!-- Names of the three XSLT templates -->
```

```
<add key="SingleResult" value="SingleResult" />
```

```
<add key="SummaryResult" value="TopLevel" />
```

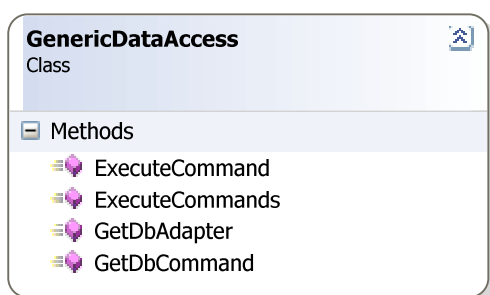
```
<add key="ApplicationResult" value="Application" />
```

6.5 Toolkit classes

The toolkit is comprised of several classes that simplify the common tasks required by the web service.

Generic Data Access

This static class provides database independent methods to assist searches.



```
public static DataTable ExecuteCommand(DbCommand command)
```

Executes the passed in command object, and returns a DataTable object containing the results.

```
public static DataSet ExecuteCommands(List<DbCommand> commands)
```

Executes the passed in command objects, and returns a DataSet object containing the results tables.

```
public static DbDataAdapter GetDbAdapter()
```

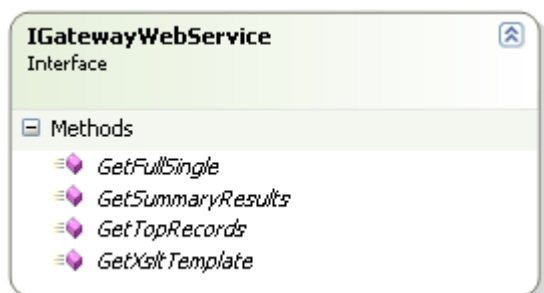
Returns a DataAdapter suitable for use with you chosen database implementation

```
public static DbCommand GetDbCommand()
```

Returns a pre-initialised command object suitable for use with your chosen database.

IGatewayWebService

An interface declaring methods that a 'Gateway Web Service' must provide.



```
public string GetFullSingle(int usertype, string searchItem, int resourceID);
```

Performs a search for the record with the specified UID

```
public string GetSummaryResults(string sessionID, int usertype, string searchItem, int firstRecord, int lastRecord, int resourceID)
```

Performs a search for records matching the search criteria

```
public string GetTopRecords(string sessionID, int userType, string searchString, int numberOfRecords, int resourceID);
```

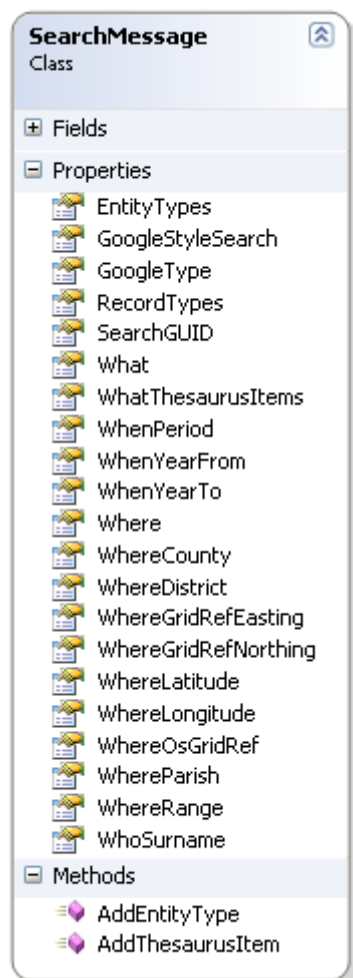
Performs a search for the top X records matching the search criteria

```
public string GetXsltTemplate();
```

Returns a string containing the contents of the latest version of the Xslt template used to format the Results of a query.

Search Message

This class stores the search criteria, for a search being made. It contains properties that set and get the various search fields held. These properties are populated by the MessageHandler class when a search message is parsed.



GoogleStyleSearch – Get/ Set text to be searched for using a google style search.

GoogleType - Get/ Set a GoogleSearchType enumeration that can be used to store the type of a google search being carried out. This can be exact, all or any.

SearchGuid – Get/ Set a id for the search being carried out.

What - Get/ Set free text specifying what is to be searched for.

WhatThesaurusItems – Get a List<ListItem> holding the Midas thesaurus items to be searched for. Each ListItem in this collection holds both the uid and name of the thesaurus terms, using the ListItems Text and Value properties.

WhenPeriod - Get/ Set a PeriodType object specifying a period that should be searched for

WhenYearFrom – Get/ Set a GatewayDate object specifying the earliest date that results should be return from.

WhenYearTo - Get/ Set a GatewayDate object specifying the earliest date that results should be retruedn from.

Where – Get/ Set a place name that should be searched for.

WhereCounty - Get/ Set text identifying the county to be searched for.

WhereDistrict - Get/ Set text identifying the district to be searched for.

WhereGridRefEasting - Get/ Set text specifying the easting element of a literal grid reference relating to the area which should be searched.

WhereGridRefNorthing - Get/ Set text specifying the northing element of a literal grid reference relating to the area which should be searched.

WhereLatitude – Get/ Set text holding the latitude of where results should be extracted from.

WhereLongitude - Get/ Set text holding the longitude of where results should be extracted from.

WhereOsGridRef – Get/ Set text specifying a ordnance survey grid reference relating to the area which should be searched.

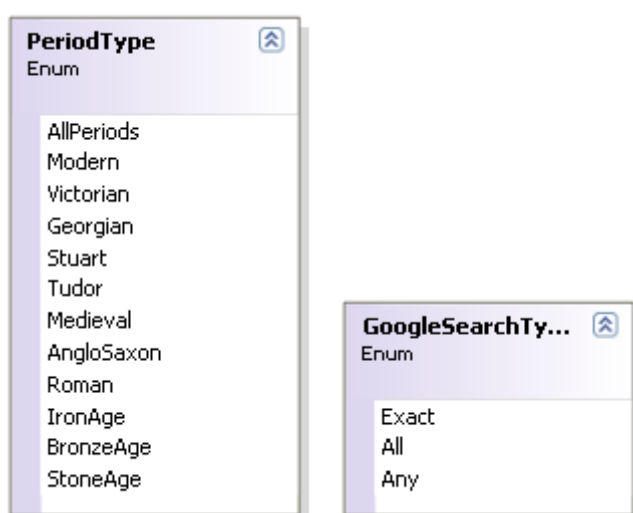
WhereRange – Get/ set text specifying the range, or distance around the spatial point specified that should be searched.

WhereParish – Get/ Set text identifying the parish to be searched for.

WhoSurname – Get/ Set text specifying the surname of a person who should be searched for.

EntityTypes - Get a List<String> holding the entity types to be included in search results. Examples of entity type would include monument, and event.

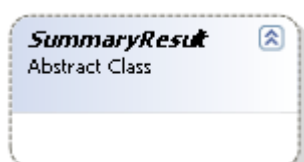
Other enumerations used by the SearchMessage object



Summary Result

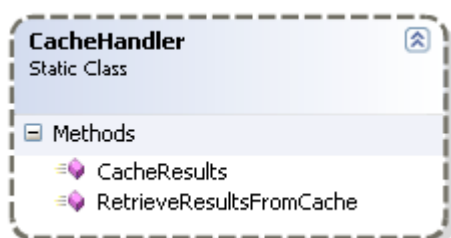
An abstract class acting as a placeholder for being stored in a ResultsCollection.

Contains no properties or methods.



CacheHandler

A static class that assists in handling the caching required.



```
public static void CacheResults(DataTable results, string searchGUID)
```

Static method that caches the results table using the specified searchGUID, as its address

```
public static DataTable RetrieveResultsFromCache(string sessionID)
```

Static method that attempts to retrieve a DataTable from the cache using the specified address.

ToolKitConfiguration

A Static class used to simplify access to the configuration properties specified in the web.config file.

CacheDuration - Number of minutes that results should be cached for

ConnectionString – The connection string for the database being used.

DbProviderName – The name of the database provider in use.

SiteName - The name of the source site here e.g. Viewfinder

StylesheetPath – The path of the XSLT template that should be used to format the results

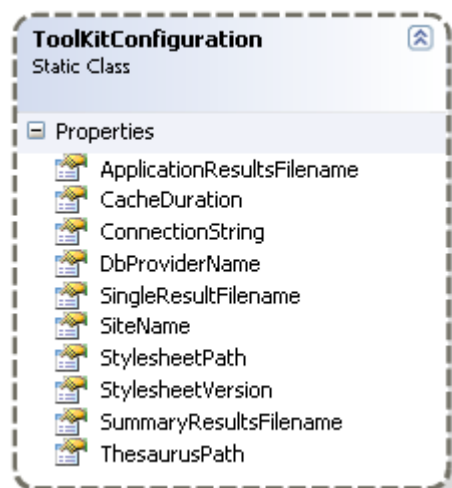
StylesheetVersion – The version number of the XsltTemplate you have available

SummaryResultsFilename/

SingleResultFilename/

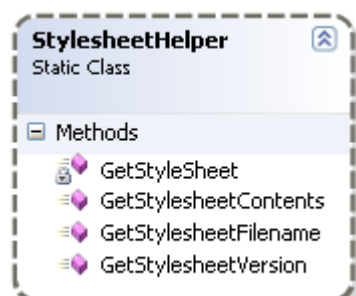
ApplicationResultsFilename - Names of the three XSLT templates

ThesaurusPath – The location of the midas XML Thesaurus.



StylesheetHelper

A static class providing methods to assist with performing stylesheet related operations.



```
public static string GetStylesheetContents()
```

Returns a string holding the contents of the xslt template.

```
public static int GetStylesheetVersion()
```

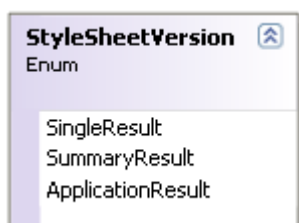
Returns an integer specifying the version number of the latest xslt template being used.

```
public static string GetStylesheetFilename(StyleSheetVersion version)
```

Returns a string containing the name of the part of the xslt template in use, e.g. for the application level results, the returned value would be Application. These values are extracted from the web.config file.

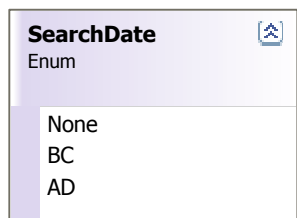
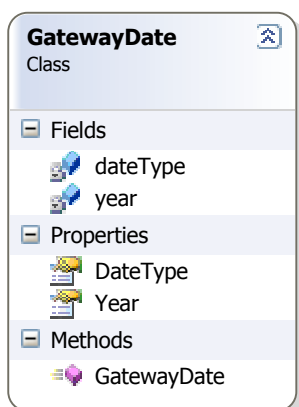
StyleSheetVersion

An enumeration used to distinguish between the three types of XSLT template.



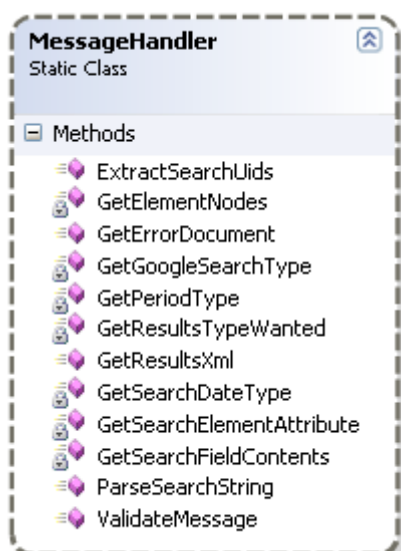
GatewayDate

A class used to represent a search date. The DateType property relates to the SearchDate enumeration, which specifies where the date is AD, BC, or unspecified. The year property relates to the year to be searched for.



MessageHandler

A class used to assist in the reading and creation of Gateway search and results messages.



```
public static List<string> ExtractSearchUids(string searchItem)
```

Returns a List<string> containing the uids of records that should be located. These uids are extracted from the passed in searchItem xml message.

```
public static XmlDataDocument GetErrorDocument(string errorMessage)
```

Return an xml formatted string, containing the required error message.

```
public static string GetResultsXml(IXmlSerializable
resultsToSerialize, Type type)
```

Uses the IxmlSerializable interface's WriteXml method to obtain the xml mark-up that represents the results of a search.

```
public static SearchMessage ParseSearchString(string strXmlMessage)
```

Parses a gateway search message into a SearchMessage object, allowing search criteria to be easily accessed.

ResultsCollection

An abstract class used as an inheritance base for the more specialised GatewayResultsCollection objects. It implements the IXmlSerializable interface to ensure that the GatewayResultsCollection object will know how to display when serialised to XML. Contains several properties storing information about the collection.

FirstRecord - The index of the first matching record included in this collection.

LastRecord - The index of the last matching record included in this collection.

RecordCount - The the number of records returned as part of this collection.

ResourceId - The id of the resource that this collection has been returned from.

SessionId - The session id that was assigned to this results collection.

Stylesheet - The name of stylesheet section to be used to format the results. Values could be TopLevel, or Application. TopLevel is for the first level of results, and Application is for when drilled down into and individual applications results.

TotalRecords - The total number of records that matched the search criteria.

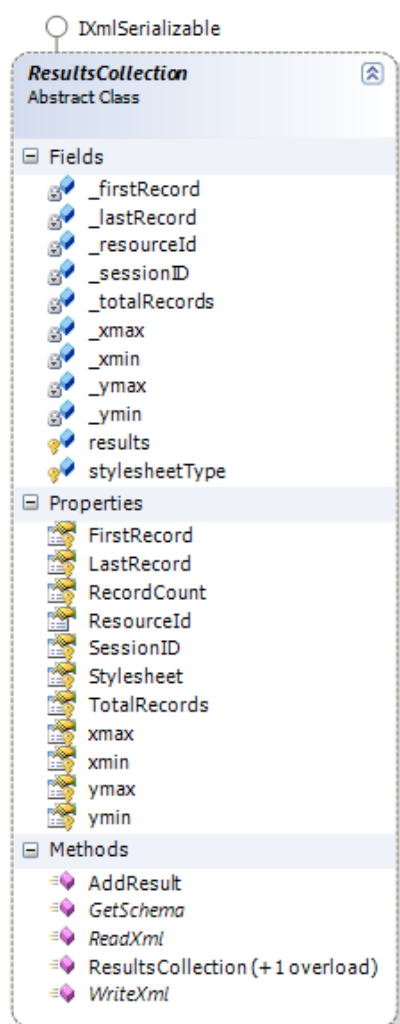
XMIN - The minimum x coordinate for this set of results in the format of a 6 digit integer

XMAX - The maximum x coordinate for this set of results in the format of a 6 digit integer

YMIN - The minimum y coordinate for this set of results in the format of a 6 digit integer

YMAX - The maximum y coordinate for this set of results in the format of a 6 digit integer

You can also access two protected fields which are *results* and *stylesheetType*. *Results* allows you to access the collection of contain results. These are stored in the format of a *List<SummaryResults>*. *StylesheetType* can be used to access the enumeration storing the type of results that are being returned, and hence the part of the stylesheet that should be used to format them.



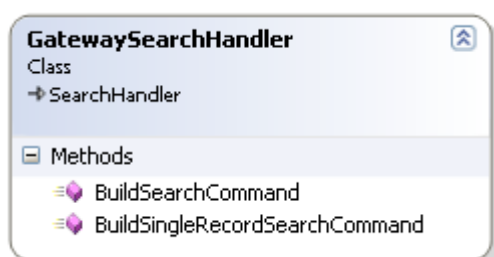
6.6 Filling in skeleton implementation

Various parts of the template have sections that require implementation. The following is a list of these sections. All of the Items are marked with a marker in the form of IR(number). To locate these areas in the source code, use control F, or select them from the user task list in visual studio.

(View – Tasks list, and select comments in the drop down list)

GatewaySearchHandler: SearchHandler

The GatewaySearchHandler class is an implementation of the abstract class SearchHandler. The main purpose of this class is to build the search commands that are used when queries are performed. There are two abstract methods of the SearchHandler class that must be implemented, in order to provide the required functionality.



Override Two SearchHandler methods.

```
1) public override List<System.Data.Common.DbCommand>
    BuildSearchCommand(SearchMessage searchMessage) {}
```

The purpose of this method is to convert the passed in SearchMessage into the DbCommands required to extract the matching records from the database.

This method is called when performing a search for multiple records (As opposed to searching for a single full record). The searchMessage object contains properties that allow you to access the search criteria for the search.

To locate this section Ctrl-F and search project for **IR1**, or select from task list (Visual studio)

```
2) public override List<System.Data.Common.DbCommand>
    BuildSingleRecordSearchCommand(string recordID)
```

The purpose of this method is to convert the passed in records uid into the DbCommands required to extract the matching records

details from the database. It takes a string representing the id of the record to be located and should return a list of DbCommand objects.

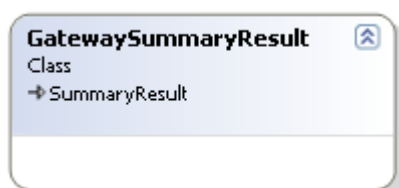
This method is called when performing a single record search.

To locate this section Cntl-F and search project for **IR2**, or select from task list (Visual studio)

GatewaySummaryResult: SummaryResult

This class is used to represent a summary of an individual result. This will be used as part of a collection of multiple search results in the GatewayResultsCollection.

This class is derived from the abstract SummaryResult class.

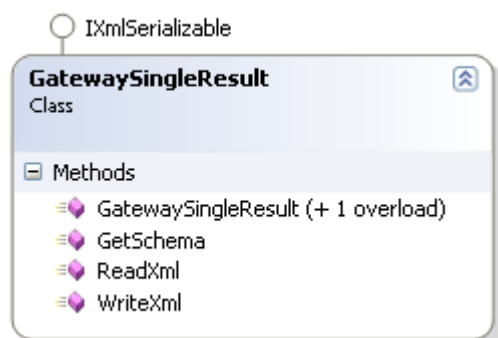


You will need to add fields to store the information that you wish to be returned for each search result. Also add Properties if required in order to access the added fields.

*To locate this section Cntl-F and search project for **IR3**, or select from task list (Visual studio)*

GatewaySingleResult

This class is used to represent a single in depth search result. It implements the XmlSerializable interface,



You will need to add fields and properties that can be used to store and access the information for a single result.

*To locate this section Cntl-F and search project for **IR4**, or select from task list (Visual studio)*

You will also need to add constructor logic to set up a new GatewaySingleResult from the passed in datatable.

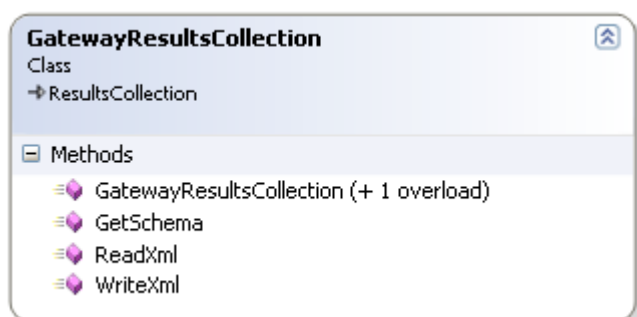
*To locate this section Cntl-F and search project for **IR5**, or select from task list (Visual studio)*

The last required task will be to implement the overridden public WriteXml(System.Xml.XmlWriter writer) method. This is called when obtaining the XML markup that represent this search result. Use the XmlWriter to manually build the required XML into the required format, using the objects added fields where required. The other IxmlSerializable methods can be left untouched (ReadXml and GetSchema)

*To locate this section Cntl-F and search project for **IR6**, or select from task list (Visual studio)*

GatewayResultsCollection: ResultsCollection

This class is used to represent a collection of search results. It is also used to generate the required XML mark-up to return for multiple search results. This class holds a collection of GatewaySummaryResults in order to store the information for each of the individual results stored within the collection. This class is derived from the ResultsCollection class.



*You will need to add field and properties in order to store and access the information for a the results collection. To locate this section Cntl-F and search project for **IR7**, or select from task list (Visual studio)*

You will also need to add consructor logic to set up a new GatewayResultsCollection object from the passed in datatable.

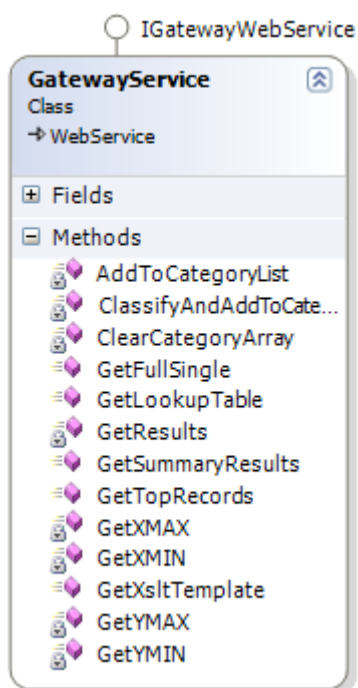
*To locate this section Cntl-F and search project for **IR8**, or select from task list (Visual studio)*

The last required task will be to implement the overridden public WriteXml(System.Xml.XmlWriter writer) method. This is called when obtaining the XML markup that represents this collection of results. Use the XmlWriter to manually build the required XML into the required format, using the objects fields where required. The other IxmlSerializable methods can be left untouched (ReadXml and GetSchema)

*To locate this section Cntl-F and search project for **IR9**, or select from task list (Visual studio)*

GatewayService: IGatewayService

This class implements the search methods defined in IGatewayWebService, and is where the main flow of the search methods is coded. A default implementation of these methods is provided, but this can be adjusted as required.



```
public string GetFullSingle(int usertype, string searchItem, int resourceID)
```

This method must be implemented to provide a search for the record with the specified UID

Purpose

To return a single result with the passed in uid.

Returns

A string containing an xml formatted results message representing the required record

Parameters

int usertype - An integer specifying the type of the user performing the search.

string searchItem - An xml formatted string containing the id of the result to be returned, and also information about the user

int resourceID - The resourceID that has been allocated to the web service being called

```
public string GetSummaryResults(string sessionID, int usertype,
string searchItem, int firstRecord, int lastRecord, int resourceID)
```

This must be implemented to provide a search for records matching the search criteria.

Purpose

To search for multiple records matching the passed in search criteria

Returns

This purpose of this method is to search for multiple records matching the passed in search criteria..

Parameters

string sessionID - A string containing the id of the result to be returned.

int userType - An integer storing the type of the user calling the search method.

string searchItem - An string holding a Xml formatted message containing the search criteria for the search.

int firstRecord/ int lastRecord - The indexes of the records to be retrieved e.g. if these values were 1 & 10, the first ten matching results should be returned.

int resourceID - The id that has been allocated to the web service being called.

```
public string GetTopRecords(string sessionID,int userType, string
```

```
searchString, int numberOfRecords, int resourceID)
```

This must be implemented to perform a search for the top X records matching the search criteria

Purpose

This purpose of this method is to search for multiple records matching the passed in search criteria. Unlike the previous method this always returns the top x results.

Returns

A string containing an xml formatted results message representing any matching search results.

Parameters

string sessionID - A string containing the id of the result to be returned.

int userType - An integer storing the type of the user calling the search method.

string searchItem - An string holding a Xml formatted message containing the search criteria for the search.

int numberOfRecords - The number of records to be returned e.g. if this value was 12, the top twelve matching results should be returned.

int resourceID - The id that has been allocated to the web service being called.

```
public string GetXsltTemplate(
```

This method must be implemented to return a string containing the contents of the latest version of the Xslt template used to format the Results of a query.

```
public int GetXMIN(DataRowCollection results)
```

This method must be implemented to return an integer containing the minimum x coordinate. This can be found by iterating the results collection.

Purpose

This purpose of this method is to calculate the minimum x coordinate for the results set that is being returned to the Heritage Gateway

Returns

An integer containing the minimum x coordinate as a 6 digit integer

Parameters

DataRowCollection results - The rows that are contained in the dataset generated by calling PerformSearch.

```
public int GetYMIN(DataRowCollection results)
```

This method must be implemented to return an integer containing the minimum y coordinate. This can be found by iterating the results collection.

Purpose

This purpose of this method is to calculate the minimum y coordinate for the results set that is being returned to the Heritage Gateway

Returns

An integer containing the minimum y coordinate as a 6 digit integer

Parameters

DataRowCollection results - The rows that are contained in the dataset generated by calling PerformSearch.

```
public int GetXMAX(DataRowCollection results)
```

This method must be implemented to return an integer containing the maximum x coordinate. This can be found by iterating the results collection.

Purpose

This purpose of this method is to calculate the maximum x coordinate for the results set that is being returned to the Heritage Gateway

Returns

An integer containing the maximum x coordinate as a 6 digit integer

Parameters

DataRowCollection results - The rows that are contained in the dataset generated by calling PerformSearch.

```
public int GetYMAX(DataRowCollection results)
```

This method must be implemented to return an integer containing the maximum y coordinate. This can be found by iterating the results collection.

Purpose

This purpose of this method is to calculate the maximum y coordinate for the results set that is being returned to the Heritage Gateway

Returns

An integer containing the maximum y coordinate as a 6 digit integer

Parameters

DataRowCollection results - The rows that are contained in the dataset generated by calling PerformSearch.

Default implementations of these methods are provided.

You can adjust/ add your own logic for these methods. The following section demonstrates the default logic of the search methods

6.7 Default Program Logic

The following sequence diagrams show the default program flow as currently set in the toolkit implementation.

GetTopRecords/ GetSummaryResults

The following diagrams show the sequence of interactions that occur when the gateway calls either the GetTopRecords, or GetSummaryResults web methods. (The only difference between these methods are the parameters that they pass into the GetResults() method, so only GetTopRecords() is shown)

Diagram1

This diagram show the basic flow for theses methods.

1. HeritageGateway calls the required web method belonging to the GatewayService Class. This methods calls the private GetResults method, passing in the desired parameters.
2. The CacheHandler class' RetrieveResultsFromCache() method is then called to try and retrieve previously cached results. This diagram shows the situation where no cached results are found. For the situation where cached results are founds please see [diagram 2](#).
3. As no cached results are found the MessageHandler's ParseSearchString() method is called. This method creates a new SearchMessage object and attempts to populate it with the searh criteria. In the case where the MessageHandler class fails to parse the search message please see [diagram 3](#).
4. The GatewaySearchHandler's PerformSearch() method is called. This method calls the private BuildSearchCommand, where the command required to perform the search is created from the SearchMessage created in step 3. The search is then carried out. Two alternate flows can results at this stage. The first is that an error occurs when the search is being carried out (see [diagram 4](#)) The second is that no results are found to match the search. (see [diagram 5](#))
5. The CacheHandler's CacheResults() method is then called to save the results for future calls.
6. The GatewaySearchHandler's LimitResults method is then used to limit the results to be returned to the Heritage Gateway.
7. A call is made to the database to store the results returned in PerformSearch into a session table (See section 6.3.4) that will be read by MapServer to generate mapping for these results.

8. A new GatewayResultsCollection object is then created using the limited results.
9. The MessageHandler's GetResultsXML() method is then called to get the Xml mark up that represents the collection.

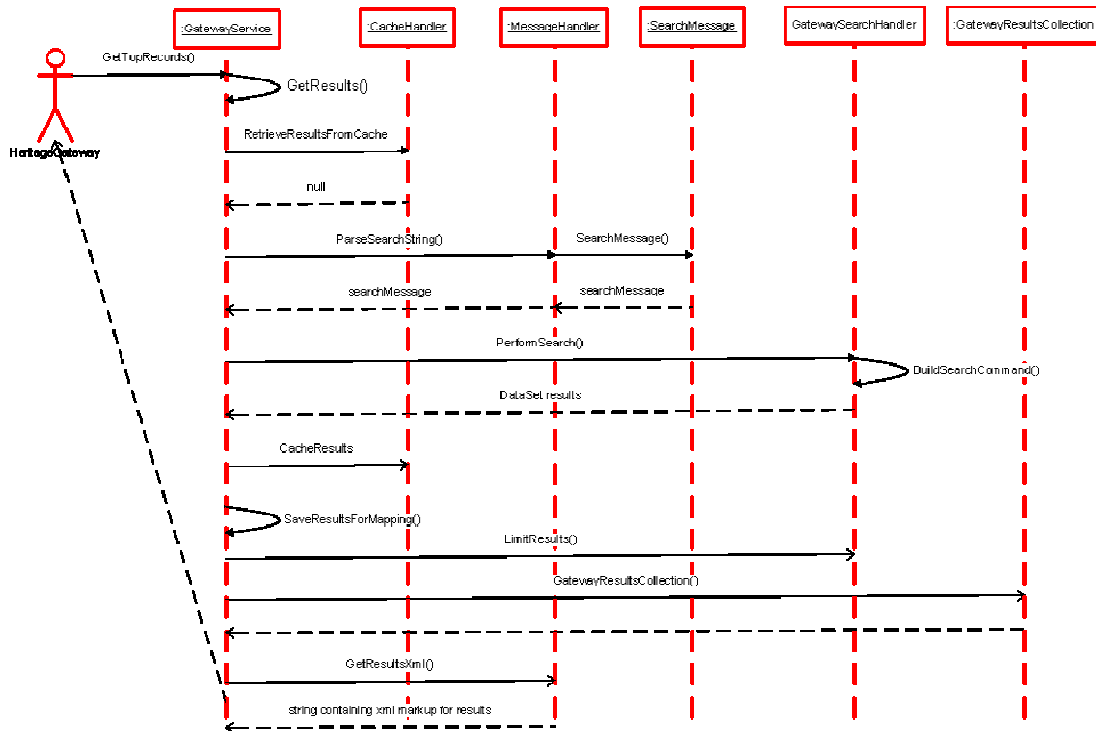


Diagram 2

This diagram show the flow when previously cached results are located.

As for Diagram 1 although steps 3, 4, 5 are missed.

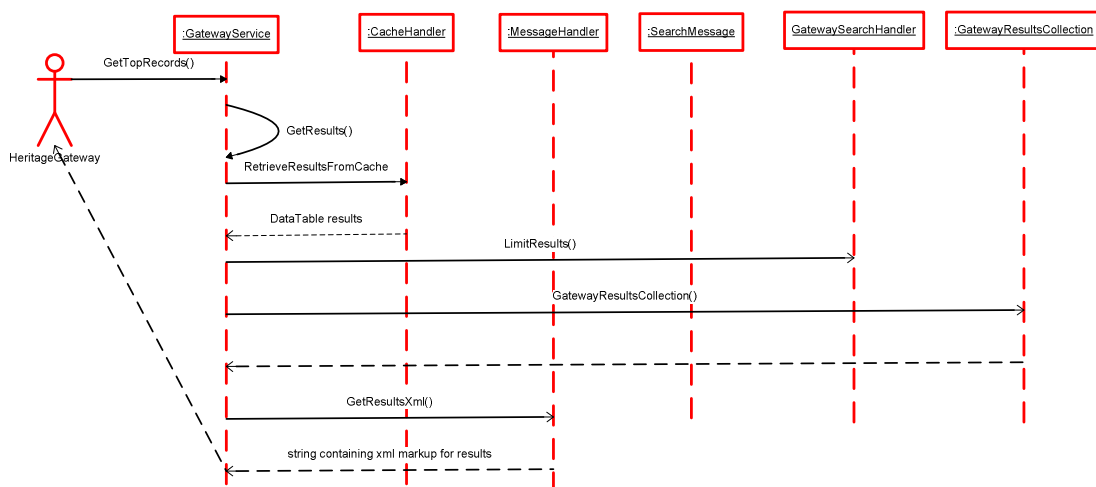


Diagram 3

This diagram shows the flow that occurs when the MessageHandler object is unable to Parse the passed in search message. In this situation steps 4 to 8 are replaced by:

4. The MessageHandler's GetErrorDocument method is called to create the required error message.
5. This message is returned to the Heritage Gateway.

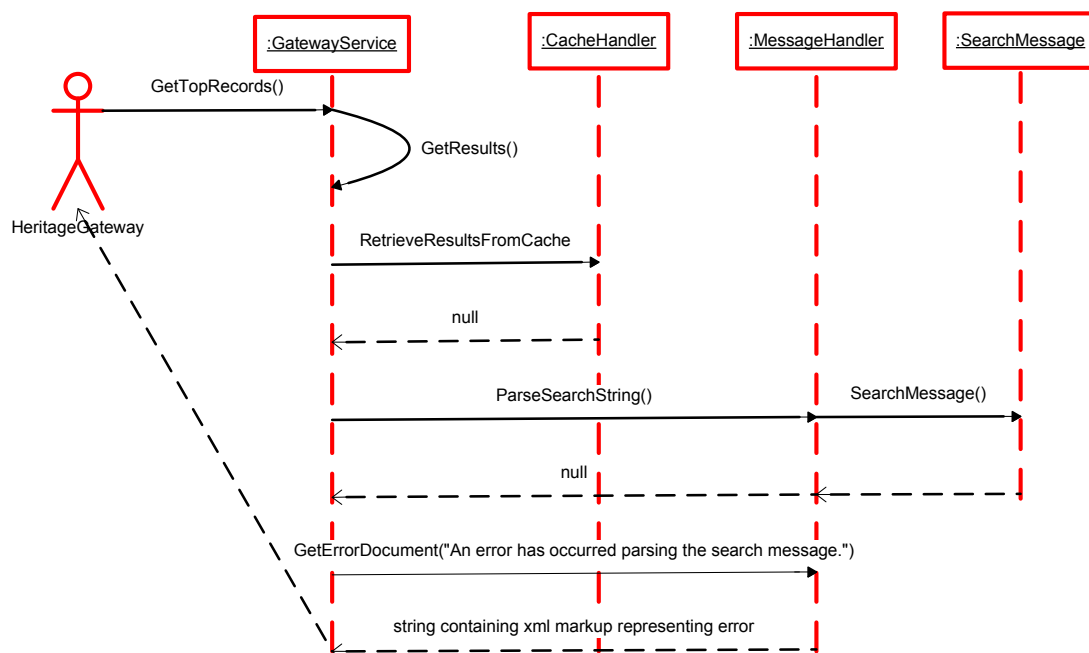


Diagram 4

This diagram shows the flow that occurs when an error occurs performing a search against the database. In this situation steps 5 to 8 are replaced by:

5. The MessageHandler's GetErrorDocument method is called to create the required error message.
6. This message is returned to the Heritage Gateway.

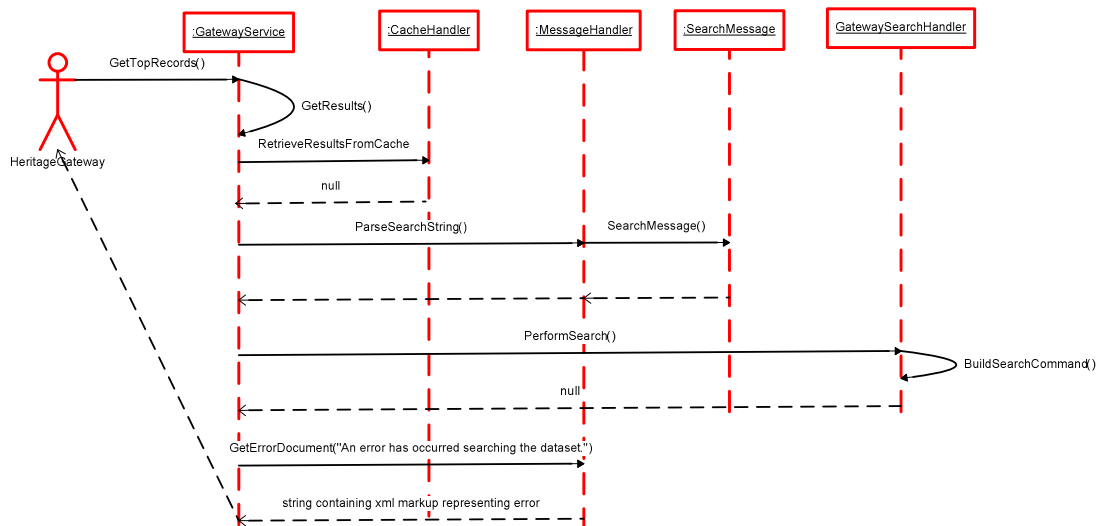
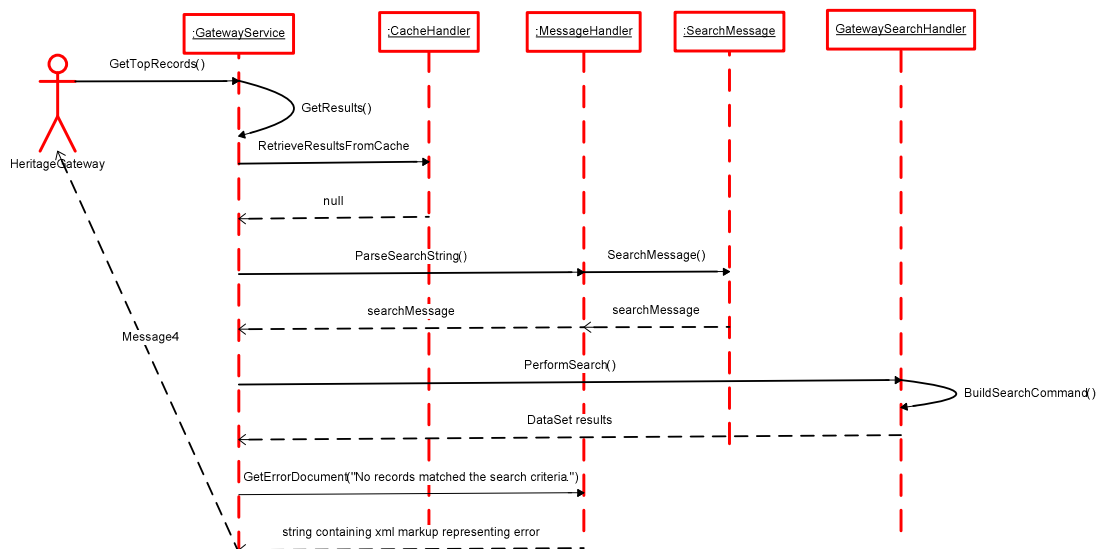


Diagram 5

This diagram shows the flow that occurs when no results are found that match the search criteria. In this situation steps 5 to 8 are replaced by:

5. The MessageHandler's GetErrorDocument method is called to create the required error message.
6. This message is returned to the Heritage Gateway.



GetFullSingle()

The following diagrams show the sequence of interactions that occur when the gateway calls the GetFullSingle() web method.

Diagram 6

This diagram show the basic flow for this method.

1. The Heritage Gateway calls the GetFullSingle web method belonging to the GatewayService Class.
2. The MessageHandler's ExtractSearchUids command is called to get the required search uids from the searchItem parameter's xml.
3. The GatewaySearchHandler's PerformSingleRecordSearch method is called. This method makes a call to the private BuildSingleRecordSearchCommand method which is used to create the search command that will be used to retrieve the required fields. During this search two alternate flows can be created. The first is that an error can occur whilst performing the search (see [diagram 7](#)). The second is that no matching results are located (see [diagram 8](#)).
4. A new GatewaySingleResult object is created and its fields populated from the results DataSet.
5. The MessageHandler's GetResultsXml method is then called to obtain the Xml mark-up that represents the results.
6. The xml is then returned to the HeritageGateway.



Diagram 7

This diagram shows the flow that occurs when an error occurs during a single record search being carried out. Steps 4, 5 and 6 are replaced by:

4. MessageHandler's GetErrorDocument method is called to create the required error message.
5. The xml marking representing the error is the returned to the Heritage Gateway.

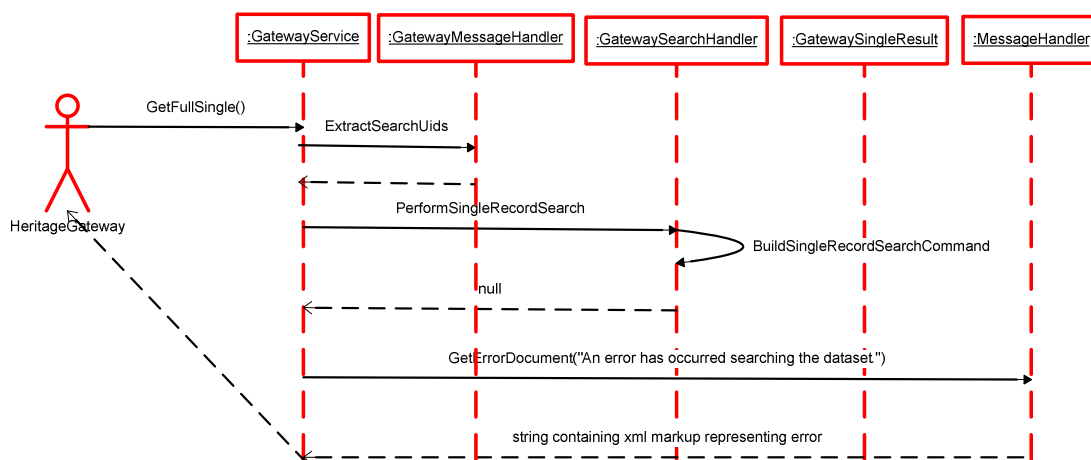
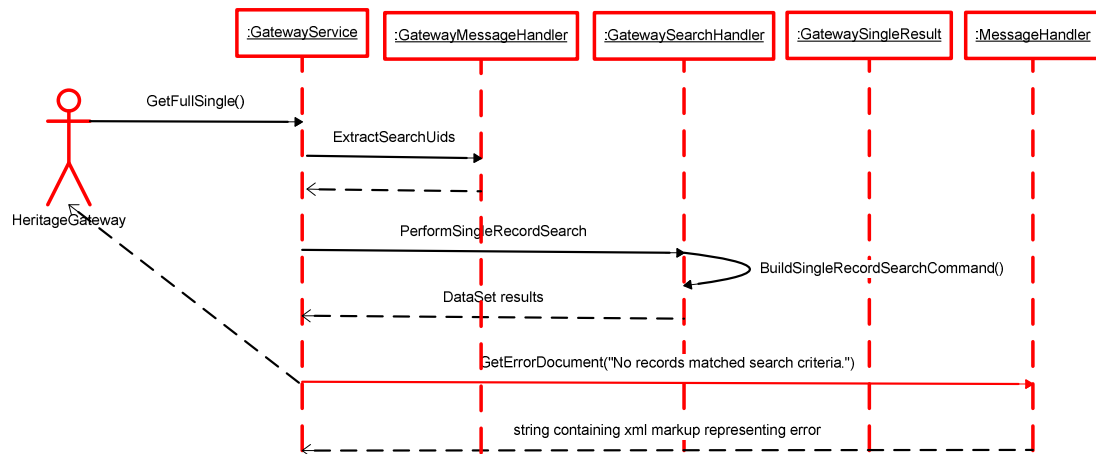


Diagram 8

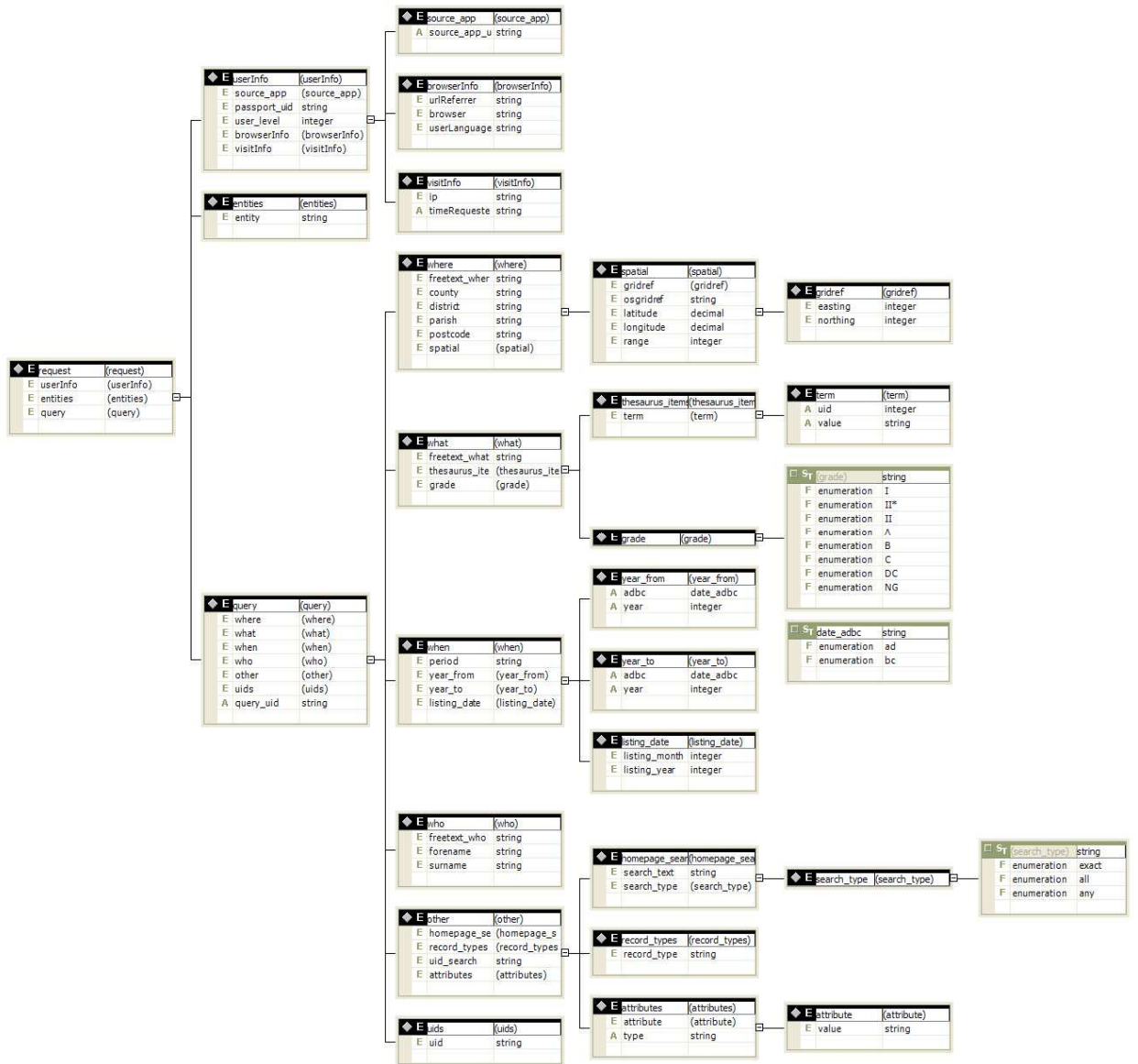
This diagram shows the flow that occurs when no match is found for the required record ID. Steps 4, 5 and 6 are replaced by:

4. MessageHandler's GetErrorDocument method is called to create the required error message.
5. The xml marking representing the error is the returned to the Heritage Gateway.



Appendix 1a

query.xsd:



Appendix 1b

base_resource.xsd:

