

Iowa Department of Transportation

# JetFire User Guide and API Reference

Prepared by Matt Rohlf  
Last updated: 2/27/2013

## Table of Contents

Changelog .....	5
Introduction to JetFire Web Services.....	6
Important Terms .....	7
SOAP, POX, and JSON Web Services .....	8
Abbreviations in Function Names .....	9
Passing XML as an input parameter:.....	10
Handling XML Output.....	11
Date Handling in the LRS Architecture.....	12
Versioning .....	13
Calling JetFire in Oracle (POX):.....	14
Calling JetFire in Oracle (SOAP):.....	17
Handling long responses:.....	20
Migrating from earlier version of JetFire:.....	21
Important Addresses.....	22
Additional Resources .....	22
Available Services.....	23
CoordRouteToDatumAL.....	23
CoordRouteToDatumAP.....	25
CoordRouteToDatumSL .....	27
CoordRouteToDatumSP .....	29
CoordRouteToDatumUL.....	30
CoordRouteToDatumUP .....	32
GetBestRouteId.....	34
GetBestRouteIdByRouteSysId .....	35
GetBestRouteIdByRouteSysName .....	36
GetCardinalDirection .....	37
GetGeoExtentIdByName.....	38
GetGeoExtentListByRouteSysId .....	39
GetGeoExtentListByRouteSysName.....	40
GetGeographicExtentList .....	41
GetRefPostList.....	42

GetRefPostListByShape .....	43
GetRouteIdA .....	44
GetRouteIdU .....	45
GetRouteLength .....	46
GetRouteList .....	47
GetRouteListByShape .....	48
GetRouteNameByRouteId .....	50
GetRoutes .....	51
GetRouteSysIdByName .....	52
GetRouteSystemList .....	53
GetTransportSystemList .....	54
LatLongToLrs .....	55
LrsToLatLong .....	56
MilepointToDatumAL .....	57
MilepointToDatumAP .....	59
MilepointToDatumSL .....	61
MilepointToDatumSP .....	62
MilepointToDatumUL .....	63
MilepointToDatumUP .....	65
RefPostToDatumAL .....	67
RefPostToDatumAP .....	69
RefPostToDatumSL .....	71
RefPostToDatumSP .....	73
RefPostToDatumUL .....	74
RefPostToDatumUP .....	76
Reproject .....	77
SnapXYToDatum .....	78
XmlObjectToCoordRouteA .....	79
XmlObjectToCoordRouteS .....	81
XmlObjectToCoordRouteU .....	82
XmlObjectToLiteral .....	84
XmlObjectToMilepointA .....	85

XmlObjectToMilepointS .....	87
XmlObjectToMilepointU .....	89
XmlObjectToRefPostA.....	91
XmlObjectToRefPostS .....	93
XmlObjectToRefPostU.....	95
XmlObjectToWkt.....	97

## Changelog

3/13/2012 – Added GetRouteNameByRouteId function.

6/29/2012 – Added GetRoutes function

7/18/2012 – Fixed some missing parameter types in the API reference section, fixed broken formatting

2/27/2013 – Updated functions for version 1.2, added date and precision parameters to the appropriate functions

7/12/2013 – Updated some text to refer to version 1.2 instead of version 1.0.

## Introduction to JetFire Web Services

JetFire is a suite of web services that provides a web-based API for the DOT's Linear Referencing System. This allows users to access LRS functionality without needing to directly log into the LRS database. Because the services are based on common standards, they can be accessed through a wide variety of environments, including .NET applications, Oracle PL/SQL procedures, Javascript-, Flex-, and Silverlight-based web applications, and many more.

Most Jetfire services perform one of a few main functions:

- Transformation: Directly calls the LRS API to transform data between linear referencing methods, such as coordinate-route, reference post, literal description, geometry, etc. See the LRS User Guide (<http://portal/IT/GIS/Shared%20Documents/LRS/LRS%20UserGuide.pdf>) for more details.
- Data retrieval: Returns lists of commonly used LRS data such as route systems, geographic extents, routes, reference posts, etc.
- Reprojection: Converts point data from one coordinate system to another. This is helpful for shifting data into or out of the LRS Lambert coordinate system.
- Lookup: Queries the database to retrieve a specific piece of information, such as the length, cardinal direction, or ID of a route, the nearest datum to a coordinate, etc.

The newest version of JetFire is built on Microsoft's Windows Communication Foundation (WCF), which enables it to support POX and JSON services. These services will provide users with more options for accessing JetFire through different applications.

## Important Terms

- JSON – JavaScript Object Notation – a standard for representing data in a text file that can be easily parsed by JavaScript and other languages.
- LRS – Linear Referencing System – A system of procedures for determining and retaining a set of locations along linear features. The system includes linear referencing method(s) and the procedures and workflows for storing, maintaining, and retrieving location information about parts of the system.
- LRS Lambert – The coordinate system used by Iowa DOT’s LRS, JetFire, and most DOT spatial databases. Coordinates passed into and out of JetFire will need to be converted into/out of LRS Lambert if they are in a different coordinate system (such as the standard WGS84 lat/long used by GPS devices).
- POX – “Plain Old XML” – basic XML data, as opposed to a more complex specification such as SOAP or WSDL.
- SOAP – originally Simple Object Access Protocol – a common protocol specification for passing data to and from web services.
- WCF – Windows Communication Foundation – a .NET Framework API for building web services. It provides more flexibility than previous options. The current version of JetFire is built on the WCF API.
- Web Service – a software system that facilitates communication between two computers over a network. Web Services are described by WSDL files, and often adhere to standard specifications like SOAP.
- WSDL – Web Services Description Language – an XML-based language that describes a web service. Clients can read the WSDL to discover which operations are available, and how to interface with them.
- XML – Extensible Markup Language – a commonly used specification for storing data that can be easily read by computers and people.

## **SOAP, POX, and JSON Web Services**

SOAP is a protocol specification for passing information to and from web services. A client builds a specially-constructed XML message (called an envelope) for a data request, which is filled with the appropriate parameters and sent to the server. The server then processes the request and returns an XML-formatted document containing the response. The client can then parse the XML to retrieve the requested information. JetFire can be accessed through the SOAP protocol. For more information on SOAP, see <http://en.wikipedia.org/wiki/SOAP>

The other two service types available through JetFire are the POX (Plain Old XML) and JSON (JavaScript Object Notation) services. Rather than building a request envelope as with SOAP, the POX and JSON services are simply called through a URL containing the service name and (optionally) the service parameters. Like SOAP, a POX service returns an XML document that can be parsed by the client. The JSON service returns the same data in JSON form, which can then be used within JavaScript applications (or any other application that parses JSON).

**NOTE:** Because the POX and JSON services pass parameters through the URL, they are subject to common URL length limitations (which are usually around 2,000 characters). **If you need to pass more than 2,000 characters as a parameter, you must use the SOAP web services for that call.** (POX/JSON and SOAP calls can be mixed within the same application if desired.) This will primarily be an issue with calls to the XmlObjectTo\* services that pass in long datum strings.

## Abbreviations in Function Names

Many JetFire functions contain abbreviations which describe the type of input accepted (e.g. `CoordRouteToDatumAL`). These abbreviations fall into two categories:

Route parameters:

- S – System-level function. These identify the route using a route ID number.
- A – Application-level function. These are designed for systems that maintain route parameters (e.g. prefix, suffix, name, type, etc.) as separate columns in a database table.
- U – User-level function. These use a parser to extract route parameters from a standard format input string.

Feature type:

- P – Point feature. Accepts one coordinate representing a single point in the world.
- L – Linear feature. Accepts two coordinates, representing the start and end point of a feature that occurs along an LRS segment.

## Passing XML as an input parameter:

Some JetFire functions require XML data (for example, a datum reference) to be passed in as an input parameter. When using SOAP, the service will attempt to parse plain XML data as part of the SOAP envelope, so the request will fail. There are two workarounds for this:

- 1) Enclose XML data in CDATA tags (<![CDATA[ and ]]>):

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:XmlObjectToWkt>
      <lrs:xmlObject><![CDATA[
        <LinearReference> ... </LinearReference>
      ]]></lrs:xmlObject>
      <lrs:date></lrs:date>
    </lrs:XmlObjectToWkt>
  </soapenv:Body>
</soapenv:Envelope>
```

- 2) Convert all 'greater than' and 'less than' characters to &gt; and &lt; respectively:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:XmlObjectToWkt>
      <lrs:xmlObject>
        &lt;LinearReference&gt; ... &lt;/LinearReference&gt;
      </lrs:xmlObject>
      <lrs:date></lrs:date>
    </lrs:XmlObjectToWkt>
  </soapenv:Body>
</soapenv:Envelope>
```

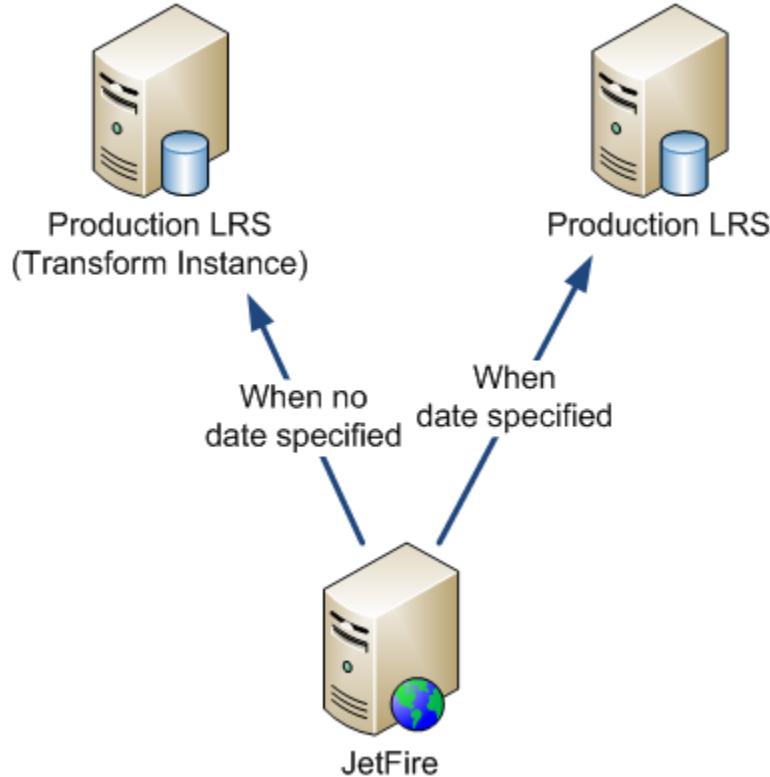
## Handling XML Output

The SOAP and POX services return their output in XML format, so it will need to be parsed before the data can be used. We recommend that you use an XPath-based parser.

It is not recommended to parse the XML output by using string functions to search for tag names. This can be imprecise and could lead to problems if slight changes are made to the output format. Future versions of JetFire may add new fields to the output, and an XPath-based parser will handle those changes correctly.

## Date Handling in the LRS Architecture

JetFire makes use of two LRS database instances: the production database, and the transform instance. Production LRS contains the full-featured LRS system, while the transform instance is optimized for performance, but does not support time-aware operations.



If no date is passed to JetFire, it runs against the faster transform instance of the LRS. If a date is passed, JetFire runs against the production instance, which can be significantly slower. It is recommended that no date should be specified unless it is completely necessary.

## Versioning

JetFire will evolve over time; however, it has been designed with a versioning scheme so that updates will not break existing code. Services will be accessed via a URL similar to the following:

<http://lrs/wcf/<version #>/JetFire.svc/HelloWorld>

Newer versions of the API will increment the version number whenever an update would break backward compatibility. Generally speaking, three types of updates are possible:

- 1) Additions of new functionality that don't affect the behavior of the existing API. For example, adding a new service or adding new fields to the output of a service. These changes will not break compatibility and will not require a version update.
- 2) Minor modifications of existing services that change the services' behavior, but not their interfaces. The services will continue to have the same inputs/outputs, so existing code will continue to be compatible, but they may provide unexpected results. These updates will be noted by a minor version update (e.g. 2.3 -> 2.4).
- 3) Major modifications that remove services or change their inputs and outputs. These will completely break compatibility for users of these services, and will be marked by a major version change (e.g. 2.4 -> 3.0). These updates should be rare, however.

New versions will be announced on the JetFire home page and the GIS SharePoint portal, and a changelog will be made available. Updates will generally be made available on the test service (<http://testlrs/>...) first.

Users will be free to continue using older versions of the API if necessary; however, new features will only be added to the latest version, so it is encouraged that you migrate your code to new versions if possible.

## Calling JetFire in Oracle (POX):

POX services are the easiest and most recommended way to use JetFire through Oracle PL/SQL. The basis of using a POX service is the `HttpUriType` function, which creates an Oracle object that can access a remote web page.

The most basic call to JetFire in Oracle looks like this:

```
SELECT httpuriType
('http://lrs/wcf/1.2/JetFire.svc/pox/GetRouteSystemList').getXML() FROM
DUAL;
```

- First, we start with the address of the POX service: <http://lrs/wcf/1.2/JetFire.svc/pox/>
- We add the name of the function we're calling, in this case `GetRouteSystemList`
- That string (<http://lrs/wcf/1.2/JetFire.svc/pox/GetRouteSystemList>) is the request to the web service. It can be pasted into a browser to see the output.
- The request string is used to create an `HttpUriType` object, and its `getXML()` function is called, which retrieves the XML object pointed at by that URL.

Most JetFire functions require one or more parameters:

```
SET ESCAPE ON;

SELECT httpuriType(httpuriType
('http://lrs/wcf/1.2/JetFire.svc/pox/CoordRouteToDatumAL?RouteSystemName=
INTERSTATE, US, STATE SIGNED&RoutePrefix=&RouteName=US
61&RouteType=&RouteSuffix=&RouteDirection=S&GeographicExtentName=STAT
E OF
IOWA&StartX=549466.5407904&StartY=183514.248724895&EndX=549274.3494552
&EndY=183069.078266497&Tolerance=&Date=').getExternalUrl()).getXML()
FROM DUAL;
```

- We start by building a request string like before:  
<http://lrs/wcf/1.2/JetFire.svc/pox/CoordRouteToDatumAL>
- Next, we add a question mark, followed by the parameters and their arguments separated by ampersands:  
<http://lrs/wcf/1.2/JetFire.svc/pox/CoordRouteToDatumAL?routeSystemName=INTERSTATE, US, STATE SIGNED&routePrefix=&routeName=US 61&routeType=&routeSuffix=&routeDirection=S&geographicExtentName=STATE OF IOWA&startX=549466.5407904&startY=183514.248724895&endX=549274.3494552&endY=183 069.078266497&tolerance=&date=>

- ‘&’ is a reserved character in Oracle and will cause problems. To get around this, we have to replace all ‘&’ with ‘\&’ and use the line ‘SET ESCAPE ON’ to tell Oracle to treat them as escape sequences.
- The service expects requests to be UTF-8 formatted (meaning that special characters are percent-encoded, e.g. spaces are converted to ‘%20’). In order to do this, we use `HttpUriType()` twice – the first to convert the string to UTF-8 by calling `getExternalUrl()`, and the second to make the actual request using `getXML()`.
- Note that in this example, we don’t pass in a route prefix, route suffix, tolerance or date. If a parameter is not passed, its name still needs to be included (followed by an equals sign).

## Parsing output

One way to work with the XML output is by using `XMLTable`. This parses the data into a virtual table, which you can then select from like any Oracle table.

The `GetGeographicExtentList` function returns an XML structure that looks like this:

```
<GeographicExtentList>
  <GeographicExtent>
    <GeographicExtentID>1562</GeographicExtentID>
    <Name>ADAIR WMA</Name>
  </GeographicExtent>
  <GeographicExtent>
    <GeographicExtentID>1630</GeographicExtentID>
    <Name>ALDO LEOPOLD WMA</Name>
  </GeographicExtent>
  <GeographicExtent>
    <GeographicExtentID>1666</GeographicExtentID>
    <Name>AMBROSE A. CALL STATE PARK</Name>
  </GeographicExtent>
  ...
</GeographicExtentList>
```

We want to parse this into a table, treating each `GeographicExtent` as a row, and each of their child tags (`GeographicExtentID` and `Name`) as columns.

```

SELECT *
FROM XMLTABLE (
    '/GeographicExtentList/GeographicExtent'
    PASSING httpuriType
    ('http://lrs/wcf/1.2/JetFire.svc/pox/GetGeographicExtentList').getXML()
    COLUMNS NAME VARCHAR2 (100) PATH 'Name',
        GEO_EXTENT_ID NUMBER (10) PATH 'GeographicExtentID') GE;

```

We call the XMLTable function, first passing it the path for the XML tag to treat as a row (/GeographicExtentList/GeographicExtent). Next, we use the PASSING keyword and pass the XML object that we retrieve using HttpUriType(). Finally, list the columns' names, data type, and path to the nodes they will be parsed from.

We can also filter the table using a where clause, or order it.

```

SELECT *
FROM XMLTABLE (
    '/GeographicExtentList/GeographicExtent'
    PASSING httpuriType
    ('http://lrs/wcf/1.2/JetFire.svc/pox/GetGeographicExtentList').getXML()
    COLUMNS NAME VARCHAR2 (100) PATH 'Name',
        GEO_EXTENT_ID NUMBER (10) PATH 'GeographicExtentID') GE
WHERE NAME LIKE 'COUNTY OF %'
ORDER BY GEO_EXTENT_ID;

```

Individual XML tags can be extracted using the XMLType.extract() function:

```

SET ESCAPE ON;

SELECT httpuriType(httpuriType(
    'http://lrs/wcf/1.2/JetFire.svc/pox/XmlObjectToWKT?xmlObject=<LinearReference>
<DatumReference><DatumPoint><AnchorSection>3837</AnchorSection><Offset>585.465
</Offset></DatumPoint></DatumReference></LinearReference>\&date=').getExternal
Url()).getXML().extract('/WKT/text()')
FROM DUAL ;

```

This code just uses the XMLType object returned by the getXML() function and calls extract(), using an XPath of '/WKT/text()', which extracts the text contents of the WKT tag.

## Calling JetFire in Oracle (SOAP):

Making SOAP calls in Oracle is a little more complicated, but they use the same basic template to build a SOAP envelope, send the request, and get the response:

```
DECLARE
    http_req      UTL_HTTP.req;
    http_resp     UTL_HTTP.resp;
    request_env   VARCHAR2(32767);
    response_env  VARCHAR2(32767);
BEGIN

    -- copy the SOAP envelope from the documentation
    request_env := '<soapenv:Envelope
        xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:lrs="http://lrs">
            <soapenv:Header/>
            <soapenv:Body>
                <lrs:GetRouteSystemList/>
            </soapenv:Body>
        </soapenv:Envelope>';

    http_req := UTL_HTTP.begin_request('http://lrs/wcf/1.2/JetFire.svc/soap',
                                         'POST',
                                         UTL_HTTP.HTTP_VERSION_1_1);

    -- building header
    UTL_HTTP.set_header(http_req, 'Content-Type', 'text/xml; charset=utf-8');
    UTL_HTTP.set_header(http_req, 'Content-Length', LENGTH(request_env));

    -- the URL here will change based on which service is being called
    UTL_HTTP.set_header(http_req,
                        'SOAPAction',
                        '"http://lrs/IJetFire/GetRouteSystemList"');

    -- uncomment to print request envelope
/*
DBMS_OUTPUT.put_line(' ');
DBMS_OUTPUT.put_line('Request envelope: ');
DBMS_OUTPUT.put_line(request_env);
DBMS_OUTPUT.put_line('Length: ' || LENGTH(request_env));
*/
    -- make the request, get the response, and read the response into a VARCHAR2
    UTL_HTTP.write_text(http_req, request_env);
    http_resp := UTL_HTTP.get_response(http_req);

    UTL_HTTP.read_text(http_resp, response_env);

    -- close http connection
    -- NOTE: it is very important to call end_response() even if the connection fails, so
make sure it is called in every case
    UTL_HTTP.end_response(http_resp);

    -- uncomment to print response envelope
/*
DBMS_OUTPUT.put_line(' ');
DBMS_OUTPUT.put_line('Response envelope: ');
DBMS_OUTPUT.put_line(response_env);
DBMS_OUTPUT.put_line(' ');
*/
END;
```

In this example, we are calling the GetRouteSystemList function.

- Start by making a VARCHAR2 with the SOAP request envelope (examples for each function can be found in the reference section at the end of this document).
- Set the header. This will always be the same, except the “<http://lrs/IJetFire/GetRouteSystemList>” will need to be changed based on the name of the function being called.
- Make the HTTP request (this part will always be the same).
- Finally, close the HTTP connection. The connection needs to be closed every time it is opened, otherwise Oracle will hit its max connection limit and further requests will fail.

To pass parameters into a service, they will need to be inserted into the SOAP envelop like so:

```
DECLARE
    route_id      NUMBER(10);
    x              NUMBER(17, 10);
    y              NUMBER(17, 10);

    http_req      UTL_HTTP.req;
    http_resp     UTL_HTTP.resp;
    request_env   VARCHAR2(32767);
    response_env  VARCHAR2(32767);
BEGIN
    route_id := 12095;
    x := 294610;
    y := 224612.26;

    -- copy the SOAP envelope from the documentation
    request_env := '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"'
    xmlns:lrs="http://lrs">
        <soapenv:Header/>
        <soapenv:Body>
            <lrs:CoordRouteToDatumSP>
                <lrs:routeId>' || route_id || '</lrs:routeId>
                <lrs:x>' || x || '</lrs:x>
                <lrs:y>' || y || '</lrs:y>
                <lrs:tolerance></lrs:tolerance>
                <lrs:date></lrs:date>
            </lrs:CoordRouteToDatumSP>
        </soapenv:Body>
    </soapenv:Envelope>';

    http_req :=
        UTL_HTTP.begin_request('http://lrs/wcf/1.2/JetFire.svc/soap',
                               'POST',
                               UTL_HTTP.HTTP_VERSION_1_1);

    -- building header
    UTL_HTTP.set_header(http_req, 'Content-Type', 'text/xml; charset=utf-8');
    UTL_HTTP.set_header(http_req, 'Content-Length', LENGTH(request_env));

    -- the URL here will change based on which service is being called
    UTL_HTTP.set_header(http_req,
                        'SOAPAction',
                        '"http://lrs/IJetFire/CoordRouteToDatumSP"');

    -- uncomment to print request envelope

    DBMS_OUTPUT.put_line(' ');
    DBMS_OUTPUT.put_line('Request envelope: ');
    DBMS_OUTPUT.put_line(request_env);
    DBMS_OUTPUT.put_line('Length: ' || LENGTH(request_env));

    -- make the request, get the response, and read the response into a VARCHAR2
    UTL_HTTP.write_text(http_req, request_env);
    http_resp := UTL_HTTP.get_response(http_req);

    UTL_HTTP.read_text(http_resp, response_env);

    -- close http connection
    -- NOTE: it is very important to call end_response() even if the connection fails, so make
    sure it is called in every case
    UTL_HTTP.end_response(http_resp);

    -- uncomment to print response envelope

    DBMS_OUTPUT.put_line(' ');
    DBMS_OUTPUT.put_line('Response envelope: ');
    DBMS_OUTPUT.put_line(response_env);
    DBMS_OUTPUT.put_line(' ');

END;
```

## Handling long responses:

There is a limitation with the UTL\_HTTP.read\_text() function that limits it to reading 32767 characters. If a response is longer than that, the data returned from read\_text() will be truncated. The workaround for this is to retrieve the data in 32767 character increments and append them into a CLOB.

Instead of calling read\_text() like this:

```
UTL_HTTP.read_text(http_resp, response_env);
```

...we create a loop that retrieves the 32767 character sections and appends them together:

```
BEGIN
  LOOP
    UTL_HTTP.read_text(http_resp, response_env, 32767);
    DBMS_LOB.writeappend (l_clob, LENGTH(response_env), response_env);
  END LOOP;
EXCEPTION
  -- when we hit the end of the text, close the connection
  WHEN UTL_HTTP.end_of_body THEN
    UTL_HTTP.end_response(http_resp);
END;
```

Note that this loop calls UTL\_HTTP.end\_response() when it reaches the end of the response text, so you will not need to call it again. Calling UTL\_HTTP.end\_response() twice on the same HTTP response variable will cause an Oracle error.

For more information, see this article:

<http://www.oracle-base.com/articles/misc/RetrievingHTMLAndBinariesIntoTablesOverHTTP.php>

## **Migrating from earlier version of JetFire:**

SOAP calls to the WCF version of JetFire are very similar to calls to previous versions; however, there are a few notable changes. If you are converting existing code to use the newest version of JetFire, please note the following:

- Because many parameters have been renamed to follow a more consistent naming convention, the SOAP envelopes have been changed. Examples of the new envelopes can be found in the last section of this document.
- The address called in UTL\_HTTP.begin\_request() has been changed from 'http://lrs/Service.asmx' to 'http://lrs/wcf/1.2/JetFire.svc/soap'
- The SOAPAction value placed in the header by UTL\_HTTP.set\_header() has been changed from "http://lrs/<name of function>" to <http://lrs/JetFire/<name of function>>

## Important Addresses

### Prod:

<a href="http://lrs/wcf/1.2/JetFire.svc?wsdl">http://lrs/wcf/1.2/JetFire.svc?wsdl</a>	- the Web Services Description Language file
<a href="http://lrs/wcf/1.2/JetFire.svc/soap/">http://lrs/wcf/1.2/JetFire.svc/soap/</a>	- SOAP endpoint
<a href="http://lrs/wcf/1.2/JetFire.svc/pox/">http://lrs/wcf/1.2/JetFire.svc/pox/</a>	- POX (Plain Old XML) endpoint
<a href="http://lrs/wcf/1.2/JetFire.svc/json/">http://lrs/wcf/1.2/JetFire.svc/json/</a>	- JSON (JavaScript Object Notation) endpoint

### Test:

<a href="http://testlrs/wcf/1.2/JetFire.svc?wsdl">http://testlrs/wcf/1.2/JetFire.svc?wsdl</a>	- the Web Services Description Language file
<a href="http://testlrs/wcf/1.2/JetFire.svc/soap/">http://testlrs/wcf/1.2/JetFire.svc/soap/</a>	- SOAP endpoint
<a href="http://testlrs/wcf/1.2/JetFire.svc/pox/">http://testlrs/wcf/1.2/JetFire.svc/pox/</a>	- POX (Plain Old XML) endpoint
<a href="http://testlrs/wcf/1.2/JetFire.svc/json/">http://testlrs/wcf/1.2/JetFire.svc/json/</a>	- JSON (JavaScript Object Notation) endpoint

## Additional Resources

LRS User Guide – <http://portal/IT/GIS/Shared%20Documents/LRS/LRS%20UserGuide.pdf>

This guide contains more detailed information about LRS concepts and features.

Oracle XML DB Reference Guide – <http://portal/IT/GIS/Shared%20Documents/Oracle/10g/b10790.pdf>

This document covers how to work with XML data in Oracle, including how to use the `HttpUriType()` function to retrieve data. Chapter 4 covers `XMLType` functions, and Chapter 17 covers `HttpUriType`.

W3Schools XPath Tutorial - <http://www.w3schools.com/xpath/default.asp>

This is a good guide for learning the concepts and syntax of XPath.

## Available Services

### CoordRouteToDatumAL

Transforms an application-level linear coordinate route reference to a datum reference.

#### Parameters:

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

string routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

string routeDirection – The direction of the route, e.g., N, W, S, E.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double startX – The x coordinate of the start of the linear event on the route (in LRS Lambert).

double startY – The y coordinate of the start of the linear event on the route (in LRS Lambert).

double endX – The x coordinate of the end of the linear event on the route (in LRS Lambert).

double endY – The y coordinate of the end of the linear event on the route (in LRS Lambert).

string tolerance – Optional. The snap tolerance for the transform, in meters. Leave blank to use default tolerance.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

**Returns:**

```
<LinearReference>
  <DatumReference>
    <DatumSequence>
      <Size>1</Size>
      <DatumExtent>
        <AnchorSection>1525</AnchorSection>
        <StartOffset>6009.291</StartOffset>
        <EndOffset>6010.288</EndOffset>
      </DatumExtent>
    </DatumSequence>
  </DatumReference>
</LinearReference>
```

**SOAP Envelope:**

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:CoordRouteToDatumAL>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routePrefix>?</lrs:routePrefix>
      <lrs:routeName>?</lrs:routeName>
      <lrs:routeType>?</lrs:routeType>
      <lrs:routeSuffix>?</lrs:routeSuffix>
      <lrs:routeDirection>?</lrs:routeDirection>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:startX>?</lrs:startX>
      <lrs:startY>?</lrs:startY>
      <lrs:endX>?</lrs:endX>
      <lrs:endY>?</lrs:endY>
      <lrs:tolerance>?</lrs:tolerance>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:CoordRouteToDatumAL>
  </soapenv:Body>
</soapenv:Envelope>
```

## **CoordRouteToDatumAP**

Transforms an application-level point coordinate route reference to a datum reference.

### **Parameters:**

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

string routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

string routeDirection – The direction of the route, e.g., N, W, S, E.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double x – The x coordinate of the point event on the route (in LRS Lambert).

double y – The y coordinate of the point event on the route (in LRS Lambert).

string tolerance – Optional. The snap tolerance for the transform, in meters. Leave blank to use default tolerance.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:CoordRouteToDatumAP>  
            <lrs:routeSystemName>?</lrs:routeSystemName>  
            <lrs:routePrefix>?</lrs:routePrefix>  
            <lrs:routeName>?</lrs:routeName>  
            <lrs:routeType>?</lrs:routeType>  
            <lrs:routeSuffix>?</lrs:routeSuffix>  
            <lrs:routeDirection>?</lrs:routeDirection>  
            <lrs:geographicExtentName>?</lrs:geographicExtentName>  
            <lrs:x>?</lrs:x>  
            <lrs:y>?</lrs:y>  
            <lrs:tolerance>?</lrs:tolerance>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</?lrs:precision>  
        </lrs:CoordRouteToDatumAP>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## **CoordRouteToDatumSL**

Transforms a system-level linear coordinate route reference to a datum reference.

### **Parameters:**

int routId – The ID of the reference route.

double startX – The x coordinate of the start of the linear event on the route (in LRS Lambert).

double startY – The y coordinate of the start of the linear event on the route (in LRS Lambert).

double endX – The x coordinate of the end of the linear event on the route (in LRS Lambert).

double endY – The y coordinate of the end of the linear event on the route (in LRS Lambert).

string tolerance – Optional. The snap tolerance for the transform, in meters. Leave blank to use default tolerance.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.

Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <DatumReference>
    <DatumSequence>
      <Size>1</Size>
      <DatumExtent>
        <AnchorSection>1525</AnchorSection>
        <StartOffset>6009.291</StartOffset>
        <EndOffset>6010.288</EndOffset>
      </DatumExtent>
    </DatumSequence>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:CoordRouteToDatumSL>  
            <lrs:routeId>?</lrs:routeId>  
            <lrs:startX>?</lrs:startX>  
            <lrs:startY>?</lrs:startY>  
            <lrs:endX>?</lrs:endX>  
            <lrs:endY>?</lrs:endY>  
            <lrs:tolerance>?</lrs:tolerance>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:CoordRouteToDatumSL>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## **CoordRouteToDatumSP**

Transforms a system-level point coordinate route reference to a datum reference.

### **Parameters:**

string routId – The ID of the reference route.

double x – The x coordinate of the point event on the route (in LRS Lambert).

double y – The y coordinate of the point event on the route (in LRS Lambert).

string tolerance – Optional. The snap tolerance for the transform, in meters. Leave blank to use default tolerance.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.

Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

### **SOAP Envelope:**

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:CoordRouteToDatumSP>
      <lrs:routId>?</lrs:routId>
      <lrs:x>?</lrs:x>
      <lrs:y>?</lrs:y>
      <lrs:tolerance>?</lrs:tolerance>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:CoordRouteToDatumSP>
  </soapenv:Body>
</soapenv:Envelope>
```

## **CoordRouteToDatumUL**

Transforms a user-level linear coordinate route reference to a datum reference.

### **Parameters:**

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double startX – The x coordinate of the start of the linear event on the route (in LRS Lambert).

double startY – The y coordinate of the start of the linear event on the route (in LRS Lambert).

double endX – The x coordinate of the end of the linear event on the route (in LRS Lambert).

double endY – The y coordinate of the end of the linear event on the route (in LRS Lambert).

string tolerance – Optional. The snap tolerance for the transform, in meters. Leave blank to use default tolerance.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.

Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <DatumReference>
    <DatumSequence>
      <Size>1</Size>
      <DatumExtent>
        <AnchorSection>1525</AnchorSection>
        <StartOffset>6009.291</StartOffset>
        <EndOffset>6010.288</EndOffset>
      </DatumExtent>
    </DatumSequence>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:CoordRouteToDatumUL>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routeDesignator>?</lrs:routeDesignator>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:startX>?</lrs:startX>
      <lrs:startY>?</lrs:startY>
      <lrs:endX>?</lrs:endX>
      <lrs:endY>?</lrs:endY>
      <lrs:tolerance>?</lrs:tolerance>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:CoordRouteToDatumUL>
  </soapenv:Body>
</soapenv:Envelope>
```

## **CoordRouteToDatumUP**

Transforms a user-level point coordinate route reference to a datum reference.

### **Parameters:**

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double x – The x coordinate of the point event on the route (in LRS Lambert).

double y – The y coordinate of the point event on the route (in LRS Lambert).

string tolerance – Optional. The snap tolerance for the transform, in meters. Leave blank to use default tolerance.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.

Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:CoordRouteToDatumUP>  
            <lrs:routeSystemName>?</lrs:routeSystemName>  
            <lrs:routeDesignator>?</lrs:routeDesignator>  
            <lrs:geographicExtentName>?</lrs:geographicExtentName>  
            <lrs:x>?</lrs:x>  
            <lrs:y>?</lrs:y>  
            <lrs:tolerance>?</lrs:tolerance>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:CoordRouteToDatumUP>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## GetBestRouteId

Returns the best route ID for a datum reference represented as an XML object.

### Parameters:

string xmlObject – The datum reference XML object.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<RouteID>1</RouteID>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetBestRouteId>
      <lrs:xmlObject>?</lrs:xmlObject>
      <lrs:date>?</lrs:date>
    </lrs:GetBestRouteId>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetBestRouteIdByRouteSysId

Returns the best route ID for a datum reference represented as an XML object.

### Parameters:

string xmlObject – The datum reference XML object.

int systemId – The route system ID.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<RouteID>1</RouteID>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetBestRouteIdByRouteSysId>
      <lrs:xmlObject>?</lrs:xmlObject>
      <lrs:systemId>?</lrs:systemId>
      <lrs:date>?</lrs:date>
    </lrs:GetBestRouteIdByRouteSysId>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetBestRouteIdByRouteSysName

Returns the best route ID for a datum reference represented as an XML object.

### Parameters:

string xmlObject – The datum reference XML object.

string systemName – Name of the route system.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<RouteID>1</RouteID>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:GetBestRouteIdByRouteSysName>  
            <lrs:xmlObject>?</lrs:xmlObject>  
            <lrs:systemName>?</lrs:systemName>  
            <lrs:date>?</lrs:date>  
        </lrs:GetBestRouteIdByRouteSysName>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## GetCardinalDirection

Gets the cardinal direction of a route.

### Parameters:

int systemId – The route system ID of the route.

int extentId – The geographic extent ID of the route.

string routeName – Name of the route.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<CardinalDirection>N</CardinalDirection>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetCardinalDirection>
      <lrs:systemId?></lrs:systemId>
      <lrs:extentId?></lrs:extentId>
      <lrs:routeName?></lrs:routeName>
      <lrs:date?></lrs:date>
    </lrs:GetCardinalDirection>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetGeoExtentIdByName

Gets the geographic extent ID of a geographic extent.

### Parameters:

string extentName – Name of the geographic extent.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<GeographicExtentId>718</GeographicExtentId>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetGeoExtentIdByName>
      <lrs:extentName>?</lrs:extentName>
      <lrs:date>?</lrs:date>
    </lrs:GetGeoExtentIdByName>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetGeoExtentListByRouteSysId

Returns a list of geographic extents in the LRS filtered by a route system name.

### Parameters:

int systemId – The route system ID.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<GeographicExtentList>
    ...
    <GeographicExtent>
        <GeographicExtentID>82</GeographicExtentID>
        <Name>COUNTY OF SCOTT</Name>
    </GeographicExtent>
    <GeographicExtent>
        <GeographicExtentID>83</GeographicExtentID>
        <Name>COUNTY OF SHELBY</Name>
    </GeographicExtent>
    ...
</GeographicExtentList>
```

### SOAP Envelope:

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:lrs="http://lrs">
    <soapenv:Header/>
    <soapenv:Body>
        <lrs:GetGeoExtentListByRouteSysId>
            <lrs:systemId?></lrs:systemId>
            <lrs:date?></lrs:date>
        </lrs:GetGeoExtentListByRouteSysId>
    </soapenv:Body>
</soapenv:Envelope>
```

## GetGeoExtentListByRouteSysName

Returns a list of geographic extents in the LRS filtered by a route system ID.

### Parameters:

string systemName – Name of the route system.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<GeographicExtentList>
    ...
    <GeographicExtent>
        <GeographicExtentID>82</GeographicExtentID>
        <Name>COUNTY OF SCOTT</Name>
    </GeographicExtent>
    <GeographicExtent>
        <GeographicExtentID>83</GeographicExtentID>
        <Name>COUNTY OF SHELBY</Name>
    </GeographicExtent>
    ...
</GeographicExtentList>
```

### SOAP Envelope:

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:lrs="http://lrs">
    <soapenv:Header/>
    <soapenv:Body>
        <lrs:GetGeoExtentListByRouteSysName>
            <lrs:systemName?></lrs:systemName>
            <lrs:date?></lrs:date>
        </lrs:GetGeoExtentListByRouteSysName>
    </soapenv:Body>
</soapenv:Envelope>
```

## GetGeographicExtentList

Returns a list of geographic extents in the LRS.

### Parameters:

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<GeographicExtentList>
    ...
    <GeographicExtent>
        <GeographicExtentID>82</GeographicExtentID>
        <Name>COUNTY OF SCOTT</Name>
    </GeographicExtent>
    <GeographicExtent>
        <GeographicExtentID>83</GeographicExtentID>
        <Name>COUNTY OF SHELBY</Name>
    </GeographicExtent>
    ...
</GeographicExtentList>
```

### SOAP Envelope:

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:lrs="http://lrs">
    <soapenv:Header/>
    <soapenv:Body>
        <lrs:GetGeographicExtentList>
            <lrs:date>?</lrs:date>
        </lrs:GetGeographicExtentList>
    </soapenv:Body>
</soapenv:Envelope>
```

## GetRefPostList

Returns the list of reference posts associated with a given route number.

### Parameters:

int routId – The route ID.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<ReferencePostList>
  <ReferencePost>
    <PostValue>0</PostValue>
  </ReferencePost>
  <ReferencePost>
    <PostValue>1</PostValue>
  </ReferencePost>
  <ReferencePost>
    <PostValue>2</PostValue>
  </ReferencePost>
  <ReferencePost>
    <PostValue>3</PostValue>
  </ReferencePost>
  <ReferencePost>
    <PostValue>4</PostValue>
  </ReferencePost>
</ReferencePostList>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetRefPostList>
      <lrs:routId>?</lrs:routId>
      <lrs:date>?</lrs:date>
    </lrs:GetRefPostList>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetRefPostListByShape

Returns the list of reference posts within buffer range of a given WKT geometry.

### Parameters:

string wkt – Well-known Text representing the input geometry.

string interaction – Oracle Spatial interaction type (see:

[http://docs.oracle.com/cd/E16338\\_01/appdev.112/e11830/sdo\\_intro.htm#i880253](http://docs.oracle.com/cd/E16338_01/appdev.112/e11830/sdo_intro.htm#i880253))

Examples include ANYINTERACT, TOUCH, EQUAL, CONTAINS, COVERS, etc.

double buffer – The buffer distance, in meters.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<ReferencePostList>
  <ReferencePost>
    <PostValue>0</PostValue>
  </ReferencePost>
  <ReferencePost>
    <PostValue>1</PostValue>
  </ReferencePost>
  <ReferencePost>
    <PostValue>2</PostValue>
  </ReferencePost>
  <ReferencePost>
    <PostValue>3</PostValue>
  </ReferencePost>
  <ReferencePost>
    <PostValue>4</PostValue>
  </ReferencePost>
</ReferencePostList>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetRefPostListByShape>
      <lrs:wkt>?</lrs:wkt>
      <lrs:interaction>?</lrs:interaction>
      <lrs:buffer>?</lrs:buffer>
      <lrs:date>?</lrs:date>
    </lrs:GetRefPostListByShape>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetRouteIdA

Gets the route ID of a route based on its application-level description.

### Parameters:

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

string routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

string routeDirection – The direction of the route, e.g., N, W, S, E.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<RouteID>1</RouteID>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:GetRouteIdA>  
            <lrs:routeSystemName>?</lrs:routeSystemName>  
            <lrs:routePrefix>?</lrs:routePrefix>  
            <lrs:routeName>?</lrs:routeName>  
            <lrs:routeType>?</lrs:routeType>  
            <lrs:routeSuffix>?</lrs:routeSuffix>  
            <lrs:routeDirection>?</lrs:routeDirection>  
            <lrs:geographicExtentName>?</lrs:geographicExtentName>  
            <lrs:date>?</lrs:date>  
        </lrs:GetRouteIdA>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## GetRouteIdU

Gets the route ID of a route based on its user-level description.

### Parameters:

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<RouteID>1</RouteID>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:GetRouteIdU>  
            <lrs:routeSystemName?></lrs:routeSystemName>  
            <lrs:routeDesignator?></lrs:routeDesignator>  
            <lrs:geographicExtentName?></lrs:geographicExtentName>  
            <lrs:date?></lrs:date>  
        </lrs:GetRouteIdU>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## GetRouteLength

Calculates the length of a route.

### Parameters:

int routeId – The route ID.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<Length>31249.23</Length>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetRouteLength>
      <lrs:routeId>?</lrs:routeId>
      <lrs:date>?</lrs:date>
    </lrs:GetRouteLength>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetRouteList

Returns the list of routes associated with a given route system and geographic extent.

### Parameters:

string systemId – Optional. The route system ID to filter with. Leave blank to not filter by route system ID.

string extentId - Optional. The geographic extent ID to filter with. Leave blank to not filter by geographic extent ID.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<RouteList>
  <Route>
    <RouteID>1000146</RouteID>
    <RouteSystemID>1</RouteSystemID>
    <AssignedRoutesID>146</AssignedRoutesID>
    <GeographicExtentID>718</GeographicExtentID>
    <RouteName>183</RouteName>
    <RouteDirection>S</RouteDirection>
    <FullName>IA 183 S</FullName>
    <OfficialName>STATE OF IOWA, IA 183 S</OfficialName>
  </Route>
  <Route>
    <RouteID>1000023</RouteID>
    <RouteSystemID>1</RouteSystemID>
    <AssignedRoutesID>23</AssignedRoutesID>
    <GeographicExtentID>718</GeographicExtentID>
    <RouteName>67</RouteName>
    <RouteDirection>S</RouteDirection>
    <FullName>US 67 S</FullName>
    <OfficialName>STATE OF IOWA, US 67 S</OfficialName>
  </Route>
</RouteList>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetRouteList>
      <lrs:systemId?></lrs:systemId>
      <lrs:extentId?></lrs:extentId>
      <lrs:date?></lrs:date>
    </lrs:GetRouteList>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetRouteListByShape

Returns the list of routes associated with a given route system and geographic extent, filtered by a given WKT geometry.

### Parameters:

string wkt

string systemId – Optional. The route system ID to filter with. Leave blank to not filter by route system ID.

string extentId - Optional. The geographic extent ID to filter with. Leave blank to not filter by geographic extent ID.

string interaction – Oracle Spatial interaction type (see:

[http://docs.oracle.com/cd/E16338\\_01/appdev.112/e11830/sdo\\_intro.htm#i880253](http://docs.oracle.com/cd/E16338_01/appdev.112/e11830/sdo_intro.htm#i880253)

Examples include ANYINTERACT, TOUCH, EQUAL, CONTAINS, COVERS, etc.

double buffer – The buffer distance, in meters.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<RouteList>
  <Route>
    <RouteID>1000146</RouteID>
    <RouteSystemID>1</RouteSystemID>
    <AssignedRoutesID>146</AssignedRoutesID>
    <GeographicExtentID>718</GeographicExtentID>
    <RouteName>183</RouteName>
    <RouteDirection>S</RouteDirection>
    <FullName>IA 183 S</FullName>
    <OfficialName>STATE OF IOWA, IA 183 S</OfficialName>
  </Route>
  <Route>
    <RouteID>1000023</RouteID>
    <RouteSystemID>1</RouteSystemID>
    <AssignedRoutesID>23</AssignedRoutesID>
    <GeographicExtentID>718</GeographicExtentID>
    <RouteName>67</RouteName>
    <RouteDirection>S</RouteDirection>
    <FullName>US 67 S</FullName>
    <OfficialName>STATE OF IOWA, US 67 S</OfficialName>
  </Route>
</RouteList>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetRouteListByShape>
      <lrs:wkt>?</lrs:wkt>
      <lrs:systemId>?</lrs:systemId>
      <lrs:extentId>?</lrs:extentId>
      <lrs:interaction>?</lrs:interaction>
      <lrs:buffer>?</lrs:buffer>
      <lrs:date>?</lrs:date>
    </lrs:GetRouteListByShape>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetRouteNameByRouteId

Returns the name, full name, and official name of a route based on its route ID.

### Parameters:

int routeId – The route ID of the route.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<Route>
  <RouteName>29</RouteName>
  <FullName>I 29 N</FullName>
  <OfficialName>STATE OF IOWA, I 29 N</OfficialName>
</Route>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetRouteNameByRouteId>
      <lrs:routeId>1</lrs:routeId>
      <lrs:date>01/01/2010</lrs:date>
    </lrs:GetRouteNameByRouteId>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetRoutes

Gets a list of routes ids related to an anchor section and offset.

### Parameters:

anchorSectionID – The unique ID of the anchor section, e.g. 21550.

offset – The offset measure from the anchor section for the route(s), e.g. 601.1.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<Routes>
<Route>
  <RouteId>12</RouteId>
  <RouteSystemId>1</RouteSystemId>
</Route>
...
</Routes>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetRoutes>
      <lrs:anchorSectionID>?</lrs:anchorSectionID>
      <lrs:offset>?</lrs:offset>
      <lrs:date>?</lrs:date>
    </lrs:GetRoutes>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetRouteSysIdByName

Returns the route system ID of a route system.

### Parameters:

string systemName – Name of the route system.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<RouteSystemId>1</RouteSystemId>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:GetRouteSysIdByName>  
            <lrs:systemName>?</lrs:systemName>  
            <lrs:date>?</lrs:date>  
        </lrs:GetRouteSysIdByName>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## GetRouteSystemList

Returns a list of the route systems in the LRS.

### Parameters:

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<RouteSystemList>
  <RouteSystem>
    <RouteSystemID>1</RouteSystemID>
    <Name>INTERSTATE, US, STATE SIGNED</Name>
  </RouteSystem>
  <RouteSystem>
    <RouteSystemID>2</RouteSystemID>
    <Name>COUNTY SIGNED ROUTES</Name>
  </RouteSystem>
  <RouteSystem>
    <RouteSystemID>3</RouteSystemID>
    <Name>MUNICIPAL SIGNED ROUTES</Name>
  </RouteSystem>
</RouteSystemList>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetRouteSystemList>
      <lrs:date>?</lrs:date>
    </lrs:GetRouteSystemList>
  </soapenv:Body>
</soapenv:Envelope>
```

## GetTransportSystemList

Returns a list of the transport systems (used for milepointing) in the LRS.

### Parameters:

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### Returns:

```
<TransportSystemList>
  <TransportSystem>
    <TransportSystemID>1</TransportSystemID>
    <Name>STATEWIDE</Name>
    <Description>Statewide
      milepointing</Description>
      <Extent>STATE OF IOWA</Extent>
    </TransportSystem>
    <TransportSystem>
      <TransportSystemID>3</TransportSystemID>
      <Name>ADAIR COUNTY PRIMARY</Name>
      <Description>Used for County Based primary
      milepointing</Description>
      <Extent>COUNTY OF ADAIR</Extent>
    </TransportSystem>
    <TransportSystem>
      <TransportSystemID>4</TransportSystemID>
      <Name>ADAMS COUNTY PRIMARY</Name>
      <Description>Used for County Based primary
      milepointing</Description>
      <Extent>COUNTY OF ADAMS</Extent>
    </TransportSystem>
  </TransportSystemList>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:GetTransportSystemList>
      <lrs:date>?</lrs:date>
    </lrs:GetTransportSystemList>
  </soapenv:Body>
</soapenv:Envelope>
```

## LatLongToLrs

Transforms a lat/long pair into the LRS Lambert coordinate system.

### Parameters:

x – The longitude of the point to be transformed.

y – The latitude of the point to be transformed.

### Returns:

```
<Coordinate>
  <X>276549.797612075</X>
  <Y>87571.9496136308</Y>
</Coordinate>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:LatLongToLrs>
      <lrs:x>?</lrs:x>
      <lrs:y>?</lrs:y>
    </lrs:LatLongToLrs>
  </soapenv:Body>
</soapenv:Envelope>
```

## LrsToLatLong

Transforms an LRS Lambert coordinate pair into a lat/long pair.

### Parameters:

x – The X coordinate of the Lambert coordinate pair.

y – The Y coordinate of the Lambert coordinate pair.

### Returns:

```
<Coordinate>
  <X>-93.8347621588629</X>
  <Y>40.7881404177634</Y>
</Coordinate>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:LrsToLatLong>
      <lrs:x>?</lrs:x>
      <lrs:y>?</lrs:y>
    </lrs:LrsToLatLong>
  </soapenv:Body>
</soapenv:Envelope>
```

## MilepointToDatumAL

Transforms an application-level linear milepoint reference to a datum reference.

### Parameters:

string transportSystemName – The name of the transport system to use for milepointing, e.g., STATEWIDE, STORY COUNTY PRIMARY, etc.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

string routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

string routeDirection – The direction of the route, e.g., N, W, S, E.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double startMilepoint – The milepoint for start of linear event.

double endMilepoint – The milepoint for end of linear event.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumSequence>
      <Size>1</Size>
      <DatumExtent>
        <AnchorSection>18853</AnchorSection>
        <StartOffset>5540.837</StartOffset>
        <EndOffset>5621.304</EndOffset>
      </DatumExtent>
    </DatumSequence>
  </DatumReference>
</LinearReference>
```

## SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:MilepointToDatumAL>
      <lrs:transportSystemName>?</lrs:transportSystemName>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routePrefix>?</lrs:routePrefix>
      <lrs:routeName>?</lrs:routeName>
      <lrs:routeType>?</lrs:routeType>
      <lrs:routeSuffix>?</lrs:routeSuffix>
      <lrs:routeDirection>?</lrs:routeDirection>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:startMilepoint>?</lrs:startMilepoint>
      <lrs:endMilepoint>?</lrs:endMilepoint>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:MilepointToDatumAL>
  </soapenv:Body>
</soapenv:Envelope>
```

## MilepointToDatumAP

Transforms an application-level point milepoint reference to a datum reference.

### Parameters:

string transportSystemName – The name of the transport system to use for milepointing, e.g., STATEWIDE, STORY COUNTY PRIMARY, etc.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

string routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

string routeDirection – The direction of the route, e.g., N, W, S, E.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double milepoint – The milepoint for the event.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:MilepointToDatumAP>
      <lrs:transportSystemName>?</lrs:transportSystemName>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routePrefix>?</lrs:routePrefix>
      <lrs:routeName>?</lrs:routeName>
      <lrs:routeType>?</lrs:routeType>
      <lrs:routeSuffix>?</lrs:routeSuffix>
      <lrs:routeDirection>?</lrs:routeDirection>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:milepoint>?</lrs:milepoint>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:MilepointToDatumAP>
  </soapenv:Body>
</soapenv:Envelope>
```

## MilepointToDatumSL

Transforms a system-level linear milepoint reference to a datum reference.

### Parameters:

int transportSystemId – The ID of the transport system to use for milepointing.  
int routeId – The route ID.  
double startMilepoint – The milepoint for start of linear event.  
double endMilepoint – The milepoint for end of linear event.  
string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.  
string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumSequence>
      <Size>1</Size>
      <DatumExtent>
        <AnchorSection>38118</AnchorSection>
        <StartOffset>3376.68</StartOffset>
        <EndOffset>1767.336</EndOffset>
      </DatumExtent>
    </DatumSequence>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:MilepointToDatumSL>
      <lrs:transportSystemId?></lrs:transportSystemId>
      <lrs:routeId?></lrs:routeId>
      <lrs:startMilepoint?></lrs:startMilepoint>
      <lrs:endMilepoint?></lrs:endMilepoint>
      <lrs:date?></lrs:date>
      <lrs:precision?></lrs:precision>
    </lrs:MilepointToDatumSL>
  </soapenv:Body>
</soapenv:Envelope>
```

## MilepointToDatumSP

Transforms a system-level point milepoint reference to a datum reference.

### Parameters:

int transportSystemId – The ID of the transport system to use for milepointing.  
int routeId – The route ID.  
double milepoint – The milepoint for the event.  
string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.  
string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:MilepointToDatumSP>
      <lrs:transportSystemId>?</lrs:transportSystemId>
      <lrs:routeId>?</lrs:routeId>
      <lrs:milepoint>?</lrs:milepoint>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:MilepointToDatumSP>
  </soapenv:Body>
</soapenv:Envelope>
```

## MilepointToDatumUL

Transforms a user-level linear milepoint reference to a datum reference.

### Parameters:

string transportSystemName – The name of the transport system to use for milepointing, e.g., STATEWIDE, STORY COUNTY PRIMARY, etc.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double startMilepoint – The milepoint for start of linear event.

double endMilepoint – The milepoint for end of linear event.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumSequence>
      <Size>1</Size>
      <DatumExtent>
        <AnchorSection>18853</AnchorSection>
        <StartOffset>5540.837</StartOffset>
        <EndOffset>5621.304</EndOffset>
      </DatumExtent>
    </DatumSequence>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:MilepointToDatumUL>
      <lrs:transportSystemName>?</lrs:transportSystemName>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routeDesignator>?</lrs:routeDesignator>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:startMilepoint>?</lrs:startMilepoint>
      <lrs:endMilepoint>?</lrs:endMilepoint>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:MilepointToDatumUL>
  </soapenv:Body>
</soapenv:Envelope>
```

## MilepointToDatumUP

Transforms a user-level point milepoint reference to a datum reference.

### Parameters:

string transportSystemName – The name of the transport system to use for milepointing, e.g., STATEWIDE, STORY COUNTY PRIMARY, etc.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double milepoint – The milepoint for the event.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.

Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:MilepointToDatumUP>  
            <lrs:transportSystemName>?</lrs:transportSystemName>  
            <lrs:routeSystemName>?</lrs:routeSystemName>  
            <lrs:routeDesignator>?</lrs:routeDesignator>  
            <lrs:geographicExtentName>?</lrs:geographicExtentName>  
            <lrs:milepoint>?</lrs:milepoint>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:MilepointToDatumUP>  
    </soapenv:Body>
```

## RefPostToDatumAL

Transforms an application-level linear reference post reference to a datum reference.

### Parameters:

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

string routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

string routeDirection – The direction of the route, e.g., N, W, S, E.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double startPostName – The reference post name for start of linear event, e.g., 4, 8, 15A, etc.

double startOffset – The offset measure from startPostName.

double endPostName – The reference post name for end of linear event, e.g., 16, 23, 42, etc.

double endOffset – The offset measure from endPostName.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumSequence>
      <Size>1</Size>
      <DatumExtent>
        <AnchorSection>18853</AnchorSection>
        <StartOffset>5540.837</StartOffset>
        <EndOffset>5621.304</EndOffset>
      </DatumExtent>
    </DatumSequence>
  </DatumReference>
</LinearReference>
```

## SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:RefPostToDatumAL>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routePrefix>?</lrs:routePrefix>
      <lrs:routeName>?</lrs:routeName>
      <lrs:routeType>?</lrs:routeType>
      <lrs:routeSuffix>?</lrs:routeSuffix>
      <lrs:routeDirection>?</lrs:routeDirection>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:startPostName>?</lrs:startPostName>
      <lrs:startOffset>?</lrs:startOffset>
      <lrs:endPostName>?</lrs:endPostName>
      <lrs:endOffset>?</lrs:endOffset>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:RefPostToDatumAL>
  </soapenv:Body>
</soapenv:Envelope>
```

## RefPostToDatumAP

Transforms an application-level point reference post reference to a datum reference.

### Parameters:

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

string routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

string routeDirection – The direction of the route, e.g., N, W, S, E.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double postName – The reference post name for start of linear event, e.g., 4, 8, 15A, etc.

double offset – The offset measure from postName.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

## SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:RefPostToDatumAP>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routePrefix>?</lrs:routePrefix>
      <lrs:routeName>?</lrs:routeName>
      <lrs:routeType>?</lrs:routeType>
      <lrs:routeSuffix>?</lrs:routeSuffix>
      <lrs:routeDirection>?</lrs:routeDirection>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:postName>?</lrs:postName>
      <lrs:offset>?</lrs:offset>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:RefPostToDatumAP>
  </soapenv:Body>
</soapenv:Envelope>
```

## RefPostToDatumSL

Transforms a system-level linear reference post reference to a datum reference.

### Parameters:

int routId – The route ID.

double startPostName – The reference post name for start of linear event, e.g., 4, 8, 15A, etc.

double startOffset – The offset measure from startPostName.

double endPostName – The reference post name for end of linear event, e.g., 16, 23, 42, etc.

double endOffset – The offset measure from endPostName.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.

Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumSequence>
      <Size>1</Size>
      <DatumExtent>
        <AnchorSection>18853</AnchorSection>
        <StartOffset>5540.837</StartOffset>
        <EndOffset>5621.304</EndOffset>
      </DatumExtent>
    </DatumSequence>
  </DatumReference>
</LinearReference>
```

### **SOAP Envelope:**

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:RefPostToDatumSL>  
            <lrs:routeId>?</lrs:routeId>  
            <lrs:startPostName>?</lrs:startPostName>  
            <lrs:startOffset>?</lrs:startOffset>  
            <lrs:endPostName>?</lrs:endPostName>  
            <lrs:endOffset>?</lrs:endOffset>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:RefPostToDatumSL>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## RefPostToDatumSP

Transforms a system-level point reference post reference to a datum reference.

### Parameters:

int routeId – The route ID.

double postName – The reference post name for start of linear event, e.g., 4, 8, 15A, etc.

double offset – The offset measure from postName.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.

Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:RefPostToDatumSP>
      <lrs:routeId>?</lrs:routeId>
      <lrs:postName>?</lrs:postName>
      <lrs:offset>?</lrs:offset>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:RefPostToDatumSP>
  </soapenv:Body>
</soapenv:Envelope>
```

## RefPostToDatumUL

Transforms a user-level linear reference post reference to a datum reference.

### Parameters:

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double startPostName – The reference post name for start of linear event, e.g., 4, 8, 15A, etc.

double startOffset – The offset measure from startPostName.

double endPostName – The reference post name for end of linear event, e.g., 16, 23, 42, etc.

double endOffset – The offset measure from endPostName.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumSequence>
      <Size>1</Size>
      <DatumExtent>
        <AnchorSection>18853</AnchorSection>
        <StartOffset>5540.837</StartOffset>
        <EndOffset>5621.304</EndOffset>
      </DatumExtent>
    </DatumSequence>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:RefPostToDatumUL>  
            <lrs:routeSystemName>?</lrs:routeSystemName>  
            <lrs:routeDesignator>?</lrs:routeDesignator>  
            <lrs:geographicExtentName>?</lrs:geographicExtentName>  
            <lrs:startPostName>?</lrs:startPostName>  
            <lrs:startOffset>?</lrs:startOffset>  
            <lrs:endPostName>?</lrs:endPostName>  
            <lrs:endOffset>?</lrs:endOffset>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:RefPostToDatumUL>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## RefPostToDatumUP

Transforms a user-level point reference post reference to a datum reference.

### Parameters:

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

double postName – The reference post name for start of linear event, e.g., 4, 8, 15A, etc.

double offset – The offset measure from postName.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:RefPostToDatumUP>
      <lrs:routeSystemName></lrs:routeSystemName>
      <lrs:routeDesignator></lrs:routeDesignator>
      <lrs:geographicExtentName></lrs:geographicExtentName>
      <lrs:postName></lrs:postName>
      <lrs:offset></lrs:offset>
      <lrs:date></lrs:date>
      <lrs:precision></lrs:precision>
    </lrs:RefPostToDatumUP>
  </soapenv:Body>
</soapenv:Envelope>
```

## Reproject

Reprojects the specified point.

### Parameters:

double x – The X coordinate of the point.

double y – The Y coordinate of the point.

int inSrid – The SRID of the input point.

int outSrid – The SRID of the desired output point.

### Notes:

Common SRIDs:

4326 = WGS84 (standard latitude/longitude)

1050010 = LRS Lambert (used by LRS and many other DOT systems)

### Returns:

```
<Coordinate>
  <X>-93.8347621588629</X>
  <Y>40.7881404177634</Y>
</Coordinate>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:Reproject>
      <lrs:x>?</lrs:x>
      <lrs:y>?</lrs:y>
      <lrs:inSRID>?</lrs:inSRID>
      <lrs:outSRID>?</lrs:outSRID>
    </lrs:Reproject>
  </soapenv:Body>
</soapenv:Envelope>
```

## SnapXYToDatum

Snaps a point specified by LRS Lambert (x,y) coordinates to the nearest datum feature and returns the datum reference in XML format.

### Parameters:

double x – The X coordinate (in LRS Lambert).

double y – The Y coordinate (in LRS Lambert).

string tolerance – Optional. The snap tolerance for the transform, in meters. Leave blank to use default tolerance.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the datum.  
Leave blank to use the LRS default value.

### Returns:

```
<LinearReference>
  <DatumReference>
    <DatumPoint>
      <AnchorSection>18853</AnchorSection>
      <Offset>5540.824</Offset>
    </DatumPoint>
  </DatumReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:SnapXYToDatum>
      <lrs:x>?</lrs:x>
      <lrs:y>?</lrs:y>
      <lrs:tolerance>?</lrs:tolerance>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:SnapXYToDatum>
  </soapenv:Body>
</soapenv:Envelope>
```

## **XmlObjectToCoordRouteA**

Transforms a datum XML object to a coordinate route reference.

### **Parameters:**

string xmlObject – The datum XML object.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

routeDirection – The direction of the route, e.g., N, W, S, E.

geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the coordinate route reference. Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <CoordinateRouteReference>
    <RouteDefinition>
      <RouteSystem>STATE</RouteSystem>
      <GeographicExtent>STATE OF IOWA</GeographicExtent>
      <RouteDescription>
        <RouteName>35</RouteName>
        <RouteType>I</RouteType>
        <RouteDirection>N</RouteDirection>
      </RouteDescription>
    </RouteDefinition>
    <Coordinate>
      <X>276549.797274761</X>
      <Y>87571.9436199561</Y>
    </Coordinate>
  </CoordinateRouteReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:XmlObjectToCoordRouteA>  
            <lrs:xmlObject>?</lrs:xmlObject>  
            <lrs:routeSystemName>?</lrs:routeSystemName>  
            <lrs:routePrefix>?</lrs:routePrefix>  
            <lrs:routeName>?</lrs:routeName>  
            <lrs:routeType>?</lrs:routeType>  
            <lrs:routeSuffix>?</lrs:routeSuffix>  
            <lrs:routeDirection>?</lrs:routeDirection>  
            <lrs:geographicExtentName>?</lrs:geographicExtentName>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:XmlObjectToCoordRouteA>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## **XmlObjectToCoordRouteS**

Transforms a datum XML object to a coordinate route reference.

### **Parameters:**

string xmlObject – The datum XML object.

string routId – The route ID.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the coordinate route reference. Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <CoordinateRouteReference>
    <RouteDefinition>
      <RouteSystem>STATE</RouteSystem>
      <GeographicExtent>STATE OF IOWA</GeographicExtent>
      <RouteDescription>
        <RouteName>35</RouteName>
        <RouteType>I</RouteType>
        <RouteDirection>N</RouteDirection>
      </RouteDescription>
    </RouteDefinition>
    <Coordinate>
      <X>276549.797274761</X>
      <Y>87571.9436199561</Y>
    </Coordinate>
  </CoordinateRouteReference>
</LinearReference>
```

### **SOAP Envelope:**

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:XmlObjectToCoordRouteS>
      <lrs:xmlObject>?</lrs:xmlObject>
      <lrs:routId>?</lrs:routId>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:XmlObjectToCoordRouteS>
  </soapenv:Body>
</soapenv:Envelope>
```

## **XmlObjectToCoordRouteU**

Transforms a datum XML object to a coordinate route reference.

### **Parameters:**

string xmlObject – The datum XML object.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.  
Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the coordinate route reference. Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <CoordinateRouteReference>
    <RouteDefinition>
      <RouteSystem>STATE</RouteSystem>
      <GeographicExtent>STATE OF IOWA</GeographicExtent>
      <RouteDescription>
        <RouteName>35</RouteName>
        <RouteType>I</RouteType>
        <RouteDirection>N</RouteDirection>
      </RouteDescription>
    </RouteDefinition>
    <Coordinate>
      <X>276549.797274761</X>
      <Y>87571.9436199561</Y>
    </Coordinate>
  </CoordinateRouteReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:XmlObjectToCoordRouteU>
      <lrs:xmlObject>?</lrs:xmlObject>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routeDesignator>?</lrs:routeDesignator>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:date>?</lrs:date>
      <lrs:precision>?<lrs:precision>
    </lrs:XmlObjectToCoordRouteU>
  </soapenv:Body>
</soapenv:Envelope>
```

## **XmlObjectToLiteral**

Transforms a datum XML object to a literal description reference.

### **Parameters:**

string xmlObject – The datum XML object.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

### **Returns:**

```
<LiteralDescription>
    ON STATE OF IOWA, US 69 N AT CITY OF AMES, CHESNUT ST, E AT OFFSET
    405.02 M TOWARDS CITY OF AMES, SE 5TH ST, W
</LiteralDescription>
```

### **SOAP Envelope:**

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:lrs="http://lrs">
    <soapenv:Header/>
    <soapenv:Body>
        <lrs:XmlObjectToLiteral>
            <lrs:xmlObject>?</lrs:xmlObject>
            <lrs:date>?</lrs:date>
        </lrs:XmlObjectToLiteral>
    </soapenv:Body>
</soapenv:Envelope>
```

## **XmlObjectToMilepointA**

Transforms a datum XML object to a milepoint reference.

### **Parameters:**

string xmlObject – The datum XML object.

string transportSystemName – The name of the transport system to use for milepointing, e.g., STATEWIDE, STORY COUNTY PRIMARY, etc.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

string routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

string routeDirection – The direction of the route, e.g., N, W, S, E.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the milepoint reference. Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <MilepointReference>
    <TransportationSystem>STATEWIDE</TransportationSystem>
    <RouteDefinition>
      <RouteSystem>STATE</RouteSystem>
      <GeographicExtent>STATE OF IOWA</GeographicExtent>
      <RouteDescription>
        <RouteName>35</RouteName>
        <RouteType>I</RouteType>
        <RouteDirection>N</RouteDirection>
      </RouteDescription>
    </RouteDefinition>
    <Milepoint>
      <Offset>15</Offset>
    </Milepoint>
  </MilepointReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:XmlObjectToMilepointA>
      <lrs:xmlObject>?</lrs:xmlObject>
      <lrs:transportSystemName>?</lrs:transportSystemName>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routePrefix>?</lrs:routePrefix>
      <lrs:routeName>?</lrs:routeName>
      <lrs:routeType>?</lrs:routeType>
      <lrs:routeSuffix>?</lrs:routeSuffix>
      <lrs:routeDirection>?</lrs:routeDirection>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:XmlObjectToMilepointA>
  </soapenv:Body>
</soapenv:Envelope>
```

## **XmlObjectToMilepoints**

Transforms a datum XML object to a milepoint reference.

### **Parameters:**

string xmlObject – The datum XML object.

int transportSystemId – The ID of the transport system to use for milepointing.

string routeld – The route ID.

int offsetType – Sets how offsets should be returned. Use 1 for offsets to be all positive, -1 for offsets to be all negative, and 0 for mixed.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the milepoint reference. Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <MilepointReference>

    <TransportationSystem>STATEWIDE</TransportationSystem>
      <RouteDefinition>
        <RouteSystem>STATE</RouteSystem>
        <GeographicExtent>STATE OF IOWA</GeographicExtent>
        <RouteDescription>
          <RouteName>29</RouteName>
          <RouteType>I</RouteType>
          <RouteDirection>N</RouteDirection>
        </RouteDescription>
      </RouteDefinition>
      <Milepoint>
        <Offset>1.5</Offset>
      </Milepoint>
    </MilepointReference>
  </LinearReference>
```

### **SOAP Envelope:**

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:XmlObjectToMilepointS>  
            <lrs:xmlObject>?</lrs:xmlObject>  
            <lrs:transportSystemId>?</lrs:transportSystemId>  
            <lrs:routeId>?</lrs:routeId>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:XmlObjectToMilepointS>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## **XmlObjectToMilepointU**

Transforms a datum XML object to a milepoint reference.

### **Parameters:**

string xmlObject – The datum XML object.

string transportSystemName – The name of the transport system to use for milepointing, e.g., STATEWIDE, STORY COUNTY PRIMARY, etc.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the milepoint reference. Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <MilepointReference>

    <TransportationSystem>STATEWIDE</TransportationSystem>
      <RouteDefinition>
        <RouteSystem>STATE</RouteSystem>
        <GeographicExtent>STATE OF IOWA</GeographicExtent>
        <RouteDescription>
          <RouteName>35</RouteName>
          <RouteType>I</RouteType>
          <RouteDirection>N</RouteDirection>
        </RouteDescription>
      </RouteDefinition>
      <Milepoint>
        <Offset>15</Offset>
      </Milepoint>
    </MilepointReference>
  </LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:lrs="http://lrs">
  <soapenv:Header/>
  <soapenv:Body>
    <lrs:XmlObjectToMilepointU>
      <lrs:xmlObject>?</lrs:xmlObject>
      <lrs:transportSystemName>?</lrs:transportSystemName>
      <lrs:routeSystemName>?</lrs:routeSystemName>
      <lrs:routeDesignator>?</lrs:routeDesignator>
      <lrs:geographicExtentName>?</lrs:geographicExtentName>
      <lrs:date>?</lrs:date>
      <lrs:precision>?</lrs:precision>
    </lrs:XmlObjectToMilepointU>
  </soapenv:Body>
</soapenv:Envelope>
```

## **XmlObjectToRefPostA**

Transforms a datum XML object to a reference post reference.

### **Parameters:**

string xmlObject – The datum XML object.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routePrefix – The prefix for the route name, e.g., OLD, W, N, etc.

string routeName – The name of the route, e.g., 80, Main, etc.

string routeType – The route type, e.g., US, RD, ST, etc.

string routeSuffix – The suffix for the route name, e.g., BUS, HWY, S, etc.

string routeDirection – The direction of the route, e.g., N, W, S, E.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

int offsetType – Sets how offsets should be returned. Use 1 for offsets to be all positive, -1 for offsets to be all negative, and 0 for mixed.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the reference post reference. Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <ReferencePostReference>
    <RouteDefinition>
      <RouteSystem>STATE</RouteSystem>
      <GeographicExtent>STATE OF IOWA</GeographicExtent>
      <RouteDescription>
        <RouteName>35</RouteName>
        <RouteType>I</RouteType>
        <RouteDirection>N</RouteDirection>
      </RouteDescription>
    </RouteDefinition>
    <ReferencePostPoint>
      <ReferencePost>16</ReferencePost>
      <Offset>0.25</Offset>
    </ReferencePostPoint>
  </ReferencePostReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:XmlObjectToRefPostA>  
            <lrs:xmlObject>?</lrs:xmlObject>  
            <lrs:routeSystemName>?</lrs:routeSystemName>  
            <lrs:routePrefix>?</lrs:routePrefix>  
            <lrs:routeName>?</lrs:routeName>  
            <lrs:routeType>?</lrs:routeType>  
            <lrs:routeSuffix>?</lrs:routeSuffix>  
            <lrs:routeDirection>?</lrs:routeDirection>  
            <lrs:geographicExtentName>?</lrs:geographicExtentName>  
            <lrs:offsetType>?</lrs:offsetType>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:XmlObjectToRefPostA>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## **XmlObjectToRefPostS**

Transforms a datum XML object to a reference post reference.

### **Parameters:**

string xmlObject – The datum XML object.

string routeld – The route ID.

int offsetType – Sets how offsets should be returned. Use 1 for offsets to be all positive, -1 for offsets to be all negative, and 0 for mixed.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the reference post reference. Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <ReferencePostReference>
    <RouteDefinition>
      <RouteSystem>STATE</RouteSystem>
      <GeographicExtent>STATE OF IOWA</GeographicExtent>
      <RouteDescription>
        <RouteName>35</RouteName>
        <RouteType>I</RouteType>
        <RouteDirection>N</RouteDirection>
      </RouteDescription>
    </RouteDefinition>
    <ReferencePostPoint>
      <ReferencePost>16</ReferencePost>
      <Offset>0.25</Offset>
    </ReferencePostPoint>
  </ReferencePostReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:XmlObjectToRefPostS>  
            <lrs:xmlObject>?</lrs:xmlObject>  
            <lrs:routeId>?</lrs:routeId>  
            <lrs:offsetType>?</lrs:offsetType>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:XmlObjectToRefPostS>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## **XmlObjectToRefPostU**

Transforms a datum XML object to a reference post reference.

### **Parameters:**

string xmlObject – The datum XML object.

string routeSystemName – The name of the route system to which the route belongs, e.g., "INTERSTATE, US, STATE SIGNED".

string routeDesignator – A text string denoting the route, e.g., STATE OF IOWA, US 80 E, etc.

string geographicExtentName – The geographic extent for the route, e.g., STATE OF IOWA, COUNTY OF STORY, etc.

int offsetType – Sets how offsets should be returned. Use 1 for offsets to be all positive, -1 for offsets to be all negative, and 0 for mixed.

string date – Optional. The date for the LRS operation to run against. Format: MM/DD/YYYY.

Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the reference post reference. Leave blank to use the LRS default value.

### **Returns:**

```
<LinearReference>
  <ReferencePostReference>
    <RouteDefinition>
      <RouteSystem>STATE</RouteSystem>
      <GeographicExtent>STATE OF IOWA</GeographicExtent>
      <RouteDescription>
        <RouteName>35</RouteName>
        <RouteType>I</RouteType>
        <RouteDirection>N</RouteDirection>
      </RouteDescription>
    </RouteDefinition>
    <ReferencePostPoint>
      <ReferencePost>16</ReferencePost>
      <Offset>0.25</Offset>
    </ReferencePostPoint>
  </ReferencePostReference>
</LinearReference>
```

### SOAP Envelope:

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:XmlObjectToRefPostU>  
            <lrs:xmlObject>?</lrs:xmlObject>  
            <lrs:routeSystemName>?</lrs:routeSystemName>  
            <lrs:routeDesignator>?</lrs:routeDesignator>  
            <lrs:geographicExtentName>?</lrs:geographicExtentName>  
            <lrs:offsetType>?</lrs:offsetType>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:XmlObjectToRefPostU>  
    </soapenv:Body>  
</soapenv:Envelope>
```

## **XmlObjectToWkt**

Transforms a datum XML object to a WKT geometry string.

### **Parameters:**

string xmlObject – The datum XML object.

string date – The date for the LRS operation to run against. Format: MM/DD/YYYY. Leave blank to use most recent data. Leaving the date blank also tends to result in a faster response time.

string precision – Optional. The number of decimal places for JetFire to return in the WKT coordinates. Leave blank to use the LRS default value.

### **Returns:**

```
<WKT>POINT (-95.6691264772602 40.5929557065894)</WKT>
```

### **SOAP Envelope:**

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:lrs="http://lrs">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <lrs:XmlObjectToWkt>  
            <lrs:xmlObject>?</lrs:xmlObject>  
            <lrs:date>?</lrs:date>  
            <lrs:precision>?</lrs:precision>  
        </lrs:XmlObjectToWkt>  
    </soapenv:Body>  
</soapenv:Envelope>
```