Building a REST Client

By Yvan Ayaay, Technical Support Engineer, 4D Inc.
TN 06-30

Introduction

Many sites such as Yahoo, Ebay, and Amazon.com offers REST developer interface for their Web Services. Usually, they provide this as another option besides the SOAP version of their web services. Representational State Transfer (REST) is a simple web-based interface that uses XML over HTTP. A REST-based web service interacts with a request based on the URL submitted. But unlike SOAP Web Services, it is not a standard. It is an architectural style for distributed computing like the World Wide Web. The Web Services Wizard of 4D cannot discover nor call with 4D web services commands these services. In this technote, it will be shown how to build a REST client which calls a REST web service and process its response within 4D. A REST client to query Yahoo Web Search will be constructed to illustrate this.

What is REST?

Representational State Transfer (REST) is an architectural approach to distributed computing such as the World Wide Web. REST came out of Roy Fielding's, one of the main authors of the HTTP specification, exposition of the architectural principle of the web. It is eminent that client requests on the web are comprised of resources (URLs). Each response to a client request can be considered a resource representation which sets the client application in a state. And as client traverses a link or make another request, another resource representation is accessed. This changes the state of the client application. The term REST was derived from this concept of state transition. Basically, a web application interacts with a resource by sending a response based on the identifier of the resource (URL) and the action required (GET, POST) in the client's request. REST web services are constructed and made available based on this principle.

REST-based web service is a simple web-based interface that uses HTTP and XML without the extra abstractions of standardized pattern such as the Web Service SOAP protocol. Even though REST is not a standard, it basically utilizes the following standards:

- HTTP
- URL
- XML

Based on the URL, a service returns a response in XML through HTTP. For example, an HTTP GET request on the URL http://www.myRESTExample.org/Employee?city=San+Jose

will return an XML with a list of employees living in the city of San Jose.
REST is designed differently from SOAP. One difference is the emphasis in identifying their services. REST emphasizes the resource while SOAP emphasizes the operation. Thus, most services in REST are nouns (e.g. city) while most services in SOAP are verbs (e.g. getCity).

A REST service has the following characteristics:

- Client-Server approach - a REST request from the client is made which is processed by the REST web service on the server.
- Stateless - each REST request contains all information necessary for it to be processed.
- Resources are universally identified - the service contains resources which are named using a URL. These resources can be accessed through HTTP methods such as GET, POST, etc.
- Traversal resource representation - the representations of the resources allow traversal to another resource.

In this technote, a REST client will be constructed. It will query Yahoo Web Search web service and process its result.

Note: For more information about REST, check this link:
http://en.wikipedia.org/wiki/REST

Constructing REST Client

REST-based web services cannot be discovered using the Web Services Wizard nor can they be called by 4D Web Services commands. REST requests are HTTP requests and the response of a REST service is in XML. In order to access a REST service, an HTTP request needs to be manually constructed within 4D. The 4D Internet Commands allows establishing a TCP connection to the server, and sending and retrieving data. Hence to build a REST client, an HTTP request (sent through GET or POST method) will be constructed and its response retrieved using 4D IC commands and the XML response will then be parsed using 4D XML commands. The sections below describe how to create a request to Yahoo Web Search service and how to parse its results.

Constructing a REST Request

Yahoo offers REST-based web services to developers such as Yahoo Web Search, Yahoo maps, Yahoo Traffic, Flickr, etc. In their website, the specifications of these APIs are listed (http://developer.yahoo.com/). To access these services in 4D, an HTTP request will have to be constructed in 4D. A request that queries the Yahoo Web Search service will be constructed to illustrate how to do this.

Below are the specifications of the **Yahoo Web Search API**:
(http://developer.yahoo.com/search/web/V1/webSearch.html)

- **Yahoo Web Service: Web Search**

The Web Search service allows you to search the Internet for web pages via REST.

- **Request URL**

http://api.search.yahoo.com/WebSearchService/V1/webSearch

- **Request Parameters**

Parameter		Value	Description
appid	string	(required)	The application ID.
query	string	(required)	The query to search for (UTF-8 encoded). This query supports the full search language of Yahoo! Search, including meta keywords
region	string:	default *us*	The regional search engine on which the service performs the search.
type	*all* (default), *any*, or *phrase*		The kind of search to submit: • all returns results with all query terms. • any returns results with one or more of the query terms. • phrase returns results containing the query terms as a phrase.
results	integer:	default *10*, max *100*	The number of results to return.
start	integer:	default *1*	The starting result position to return (1-based). The finishing position (start + results - 1) cannot exceed 1000.
format	*any* (default), *html*, *msword*, *pdf*, *ppt*, *rss*, *txt*, *xls*		Specifies the kind of file to search for.
Adult_ok	*no value* (default), or *1*		Specifies whether to allow results with adult content. Enter a *1* to allow adult content.
similar_ok	*no value* (default), or *1*		Specifies whether to allow multiple results with similar content. Enter a *1* to allow similar content.
language	string:	default *no value* (all languages)	The language the results are written in.
country	string:	default *no value*	The country in which to restrict your search results. Only results on web sites within this country are returned.
site	string:	default *no value*	A domain to restrict your searches to (e.g. www.yahoo.com). You may submit up to 30 values (site=www.yahoo.com&site=www.cnn.com).
subscription	string:	default *no value*	Any subscriptions to premium content that should also be searched. You may submit multiple values.
license	*any* (default), *cc_any*, *cc_commercial*, *cc_modifiable*		The Creative Common Licenses that the contents are licensed under. You may submit multiple values (e.g. license=cc_commercial&license=cc_modifiable).

output	string: *xml* (default), *json, php*	The format for the output.
callback	string	The name of the callback function to wrap around the JSON data. The following characters are allowed: A-Z a-z 0-9 . [] and _. If output=json has not been requested, this parameter has no effect. More information on the callback can be found in the

- **Response fields of the XML returned:**

Field	Description
ResultSet	Contains all of the query responses. Has attributes: • totalResultsAvailable: The number of query matches in the database. • totalResultsReturned: The number of query matches returned. This may be lower than the number of results requested if there were fewer total results available. • firstResultPosition: The position of the first result in the overall search.
Result	Contains each individual response.
Title	The title of the web page.
Summary	Summary text associated with the web page.
Url	The URL for the web page.
ClickUrl	The URL for linking to the page..
MimeType	The MIME type of the page
ModificationDate	The date the page was last modified, in unix timestamp format.
Cache	The URL of the cached result, and its size in bytes.

For example, when performing a web search for **George W. Bush**, the URL request appears as shown below:
http://api.search.yahoo.com/WebSearchService/V1/webSearch?appid=Test4DREST&query=George+W.+Bush&results=2

This URL can easily be constructed and then sent to server within 4D using 4D Internet commands. Notice the application ID (appid) 'Test4DREST' inserted before the query in the request string. To access Yahoo Webservices, a user will need to get an application ID (http://login.yahoo.com/?.done=http://api.search.yahoo.com/webservices/register_application) that uniquely identifies a user's application. The method below shows how to construct the above URL request (with any other subject to search) and retrieve the result from the REST service:

`Method: GetResultYahoo
`Description: This method construct the http GET request and sends it to the server. It then retrieves the result of the web service and returns it.
` Parameters: The subject to be searched is passed as the first parameter and the number of results is the second parameter.

```
C_TEXT($Text;hostname)
C_LONGINT($Status;TCPID;vIPPort_Remote;$err)
C_TEXT($1;request_String)
C_BLOB($0;$result;$buffer)
C_LONGINT($2)

vIPPort_Remote:=80
vIP_Timeout:=10
vIP_Sessionfiltered:=1    `Asynchronous
hostname:="api.search.yahoo.com"    `API hostname

$err:=TCP_Open  (hostname;vIPPort_Remote;TCPID;vIP_SessionSettings)
$err:=TCP_State  (TCPID;$Status)
   `build http request
request_String:="GET
/WebSearchService/V1/webSearch?appid=Test4DREST&query="+$1+"&results="+String($2)
request_String:=request_String+"\n"+"HTTP/1.1"

If ($Status=8)    `connection was established
      $err:=TCP_Send  (TCPID;request_String)    `send request
End if


SET BLOB SIZE($result;0)
Repeat      ` retrieve result from server
      SET BLOB SIZE($buffer;0)
      $err:=TCP_ReceiveBLOB    (TCPID;$buffer)
      $err:=TCP_State  (TCPID;vState)
      $sizeBuffer:=BLOB size($buffer)
      $sizeResult:=BLOB size($result)
         `Append buffer BLOB to result BLOB
      INSERT IN BLOB($result;$sizeResult;$sizeBuffer)
      COPY BLOB($buffer;$result;0;$sizeResult;$sizeBuffer)
Until ((vState=0) | ($err#0))

$err:=TCP_State  (TCPID;$Status)
$err:=TCP_Close  (TCPID)
$0:=$result  ` return BLOB result
```

In the method above, a TCP connection is first established. Then, the URL request via GET method is sent. The parameters, the subject to be searched and the number of results, are appended to the request string. The request is then sent to the server. Next, the method waits to receive the entire response. Usually, web servers automatically close their connections once they have performed their action. Thus, it goes to a repeat loop until the connection is closed. The retrieved XML result stored in a BLOB is returned.
The returned result in a BLOB of this method can then be saved into an XML file just like the method below to be parsed later:

```
   `Method: P_XMLSave
   ` Description: Save BLOB to an XML file.

C_TIME(vhDoc)
```

```
C_BLOB($resultBLOB)
$resultBLOB:=GetResultYahoo("George W. Bush";2)
vhDoc:=Create document("ResultREST.xml")  `  save result in xml file.
If (OK=1)
        CLOSE DOCUMENT(vhDoc)  `  Close the document
        BLOB TO DOCUMENT("ResultREST.xml";$resultBLOB)
End if
```

A sample XML response from the request above returned by the Yahoo Web Search web service is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn:yahoo:srch"
xsi:schemaLocation="urn:yahoo:srch
http://api.search.yahoo.com/WebSearchService/V1/WebSearchResponse.xsd" type="web"
totalResultsAvailable="44700000" totalResultsReturned="2" firstResultPosition="1"
moreSearch="/WebSearchService/V1/webSearch?query=George+W.+Bush&amp;appid=ivandterrible&amp;regi
on=us">
<Result><Title>President of the United States - George W. Bush</Title><Summary>Takes a look inside the
Oval Office and includes a biography of President Bush, a video tour, information on the latest policy
initiatives. From the official White House
site.</Summary><Url>http://www.whitehouse.gov/president</Url><ClickUrl>http://uk.wrs.yahoo.com/_ylt=
A0Je5VuxjMZE3j4AElDdmMwF;_ylu=X3oDMTB2cXVjNTM5BGNvbG8DdwRsA1dTMQRwb3MDMQRzZWMDc3IEdnR
pZAM-
/SIG=11nmlojnp/EXP=1153949233/**http%3a//www.whitehouse.gov/president</ClickUrl><DisplayUrl>www.
whitehouse.gov/president</DisplayUrl><ModificationDate>1153810800</ModificationDate><MimeType>text
/html</MimeType>
<Cache><Url>http://uk.wrs.yahoo.com/_ylt=A0Je5VuxjMZE3j4AFVDdmMwF;_ylu=X3oDMTBwOHA5a2tvBGNv
bG8DdwRwb3MDMQRzZWMDc3IEdnRpZAM-
/SIG=16n48lrqp/EXP=1153949233/**http%3a//66.218.69.11/search/cache%3fei=UTF-
8%26appid=ivandterrible%26query=George%2bW.%2bBush%26results=2%26u=www.whitehouse.gov/presid
ent%26w=george%2bw%2bbush%26d=RAS9xmP9NKug%26icp=1%26.intl=us</Url><Size>30244</Size></
Cache>
</Result>

<Result><Title>George W. Bush - Wikipedia, the free encyclopedia</Title><Summary>User-created profile of
the 43rd U.S. President, George W. Bush. Includes biographical information, notes on his career, policies, and
legislation.</Summary><Url>http://en.wikipedia.org/wiki/George_W._Bush</Url><ClickUrl>http://uk.wrs.yah
oo.com/_ylt=A0Je5VuxjMZE3j4AF1DdmMwF;_ylu=X3oDMTB2ZjQ4dDExBGNvbG8DdwRsA1dTMQRwb3MDMgRz
ZWMDc3IEdnRpZAM-
/SIG=11v5cqe3m/EXP=1153949233/**http%3a//en.wikipedia.org/wiki/George_W._Bush</ClickUrl><Display
Url>en.wikipedia.org/wiki/George_W._Bush</DisplayUrl><ModificationDate>1153810800</ModificationDate>
<MimeType>text/html</MimeType>
<Cache><Url>http://uk.wrs.yahoo.com/_ylt=A0Je5VuxjMZE3j4AGlDdmMwF;_ylu=X3oDMTBwZG5hOWwzBGNv
bG8DdwRwb3MDMgRzZWMDc3IEdnRpZAM-
/SIG=16vdtetlv/EXP=1153949233/**http%3a//66.218.69.11/search/cache%3fei=UTF-
8%26appid=ivandterrible%26query=George%2bW.%2bBush%26results=2%26u=en.wikipedia.org/wiki/George
_W._Bush%26w=george%2bw%2bbush%26d=UpH_T2P9NKvh%26icp=1%26.intl=us</Url><Size>182348</
Size></Cache>
</Result>
```

</ResultSet>

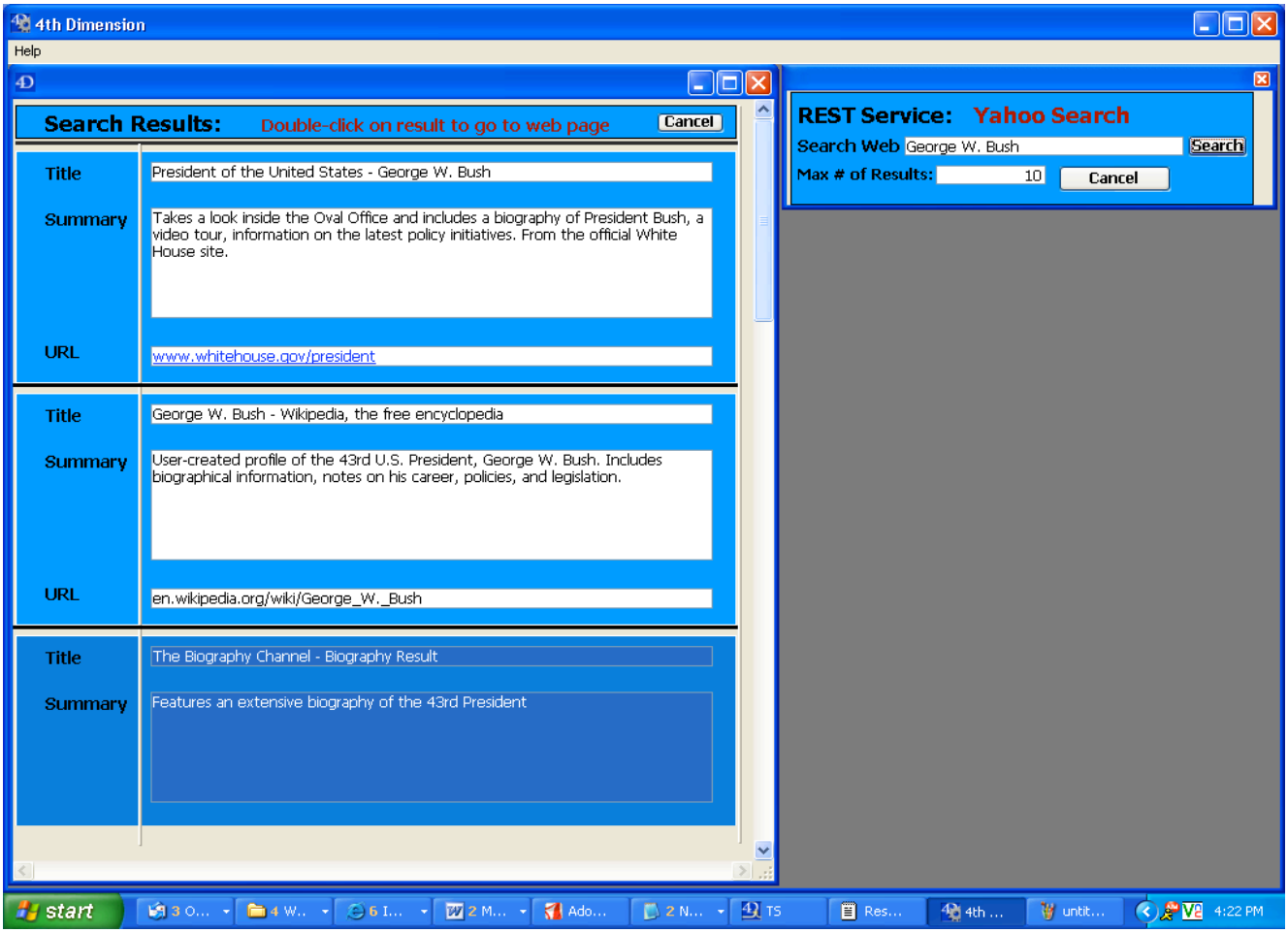
Parsing of the XML response

The sample XML response above can be parsed using 4D DOM XML commands. The method below parses and retrieves all results in the XML response. The results are saved into a table wherein they can be retrieved and listed. Only the current results of the search request are saved in the table. Using DOM XML commands, it traverses through the XML to retrieve the element values of each result. Each result retrieved is saved into a record created.

```
   `Method: P_XMLParse
   `Description: This method parses the XML and get all results and then save the current records in the SearchResult table.

C_STRING(16;$xml_Element_Ref;$xref1)
C_STRING(16;$xTitle;$xSummary;$xUrl;$xClickUrl;$xDisplayUrl)

ALL RECORDS([SearchResult])
LAST RECORD([SearchResult])
   `Delete all current records
For (i;1;Records in selection([SearchResult]))
      DELETE RECORD([SearchResult])
      PREVIOUS RECORD([SearchResult])
End for

$xml_Ref:=DOM Parse XML source("ResultREST.xml")   ` Parse result xml
$xref1:=DOM Find XML element($xml_Ref;"/Result/")   ` Find result starting node
While (OK=1)   ` Loop until all results are retrieved. The Title, Summary, Url, ClickUrl, and DisplayUrl for each result are saved in the table.
      CREATE RECORD([SearchResult])   ` create a record to save result
      $xTitle:=DOM Find XML element($xref1;"/Result/Title")
      DOM GET XML ELEMENT VALUE($xTitle;$mTitle)
      [SearchResult]Title:=$mTitle
      $xSummary:=DOM Find XML element($xref1;"/Result/Summary")
      DOM GET XML ELEMENT VALUE($xSummary;$mSummary)
      [SearchResult]Summary:=$mSummary
      $xUrl:=DOM Find XML element($xref1;"/Result/Url")
      DOM GET XML ELEMENT VALUE($xUrl;$mUrl)
      [SearchResult]URL:=$mUrl
      $xClickUrl:=DOM Find XML element($xref1;"/Result/ClickUrl")
      DOM GET XML ELEMENT VALUE($xClickUrl;$mClickUrl)
      [SearchResult]ClickUrl:=$mClickUrl
      $xDisplayUrl:=DOM Find XML element($xref1;"/Result/DisplayUrl")
      DOM GET XML ELEMENT VALUE($xDisplayUrl;$mDisplayUrl)
      [SearchResult]DisplayUrl:=$mDisplayUrl
      SAVE RECORD([SearchResult])
      $xref1:=DOM Get Next sibling XML element($xref1)   ` Got to the next result
End while

DOM CLOSE XML($xml_Ref)
```

Sample Database

The sample database that comes with this technote is the implementation of the REST Client for Yahoo Web Search. The interface is shown below. The subject is entered in the Yahoo Search dialog and, then, the search results are displayed. If a listed search result is double-clicked, the default browser displays the web page of the selected search result.



Conclusion

REST-based web services can be called within 4D. Since they cannot be discovered nor can it be called by web services command, REST queries need to be manually constructed and its response parsed within 4D to be able to build a REST Client. REST queries are simple HTTP requests that contain enough information that the REST-based web service server can interact with. An HTTP request can be built using 4D Internet Commands. With 4D IC, a TCP connection can be established, a request constructed can be sent, and the result from the REST service response can be retrieved. The response of REST is usually in XML. Parsing can be done using 4D XML commands to retrieve the results. This entire approach to build a REST client was shown in this technote. The Yahoo Web Search service, a REST-based web service, was queried and its response was parsed.