Information Builders

DB2 Web Query Web Services

IBM

Contents

Preface

This documentation describes how to create Web Services for DB2 Web Query features by accessing supplied Web pages that provide the basic development interface for Web Services support. It also describes the function calls required to implement the capability. It is intended for experienced developers who will use this capability to expose Web Query functionality as callable services from a Microsoft Visual Studio .NET or J2EE development platform. Developers should have a knowledge of Web Service technology and object-oriented programming.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	**Contents**
1	Web Query Web Services	Describes Web Services, components, and how to use Web Services with Web Query features.
2	Using the Web Query WSDL Utility	Describes the Web Query WSDL utility used by .NET and Java programmers to create WSDL files that call the set of Web Query Web Services functions.
3	Web Query Web Services Structures	Describes the structures used for Web Query Web Service functions.
4	Web Query Web Services Functions	Describes the functions used by Web Query Web Services.
5	Troubleshooting Web Query Web Services	Provides information about troubleshooting Web Services.

Documentation Conventions

The following table lists and describes the conventions that apply in this manual.

Convention	**Description**

`THIS TYPEFACE`
or
`this typeface`
	Denotes syntax that you must enter exactly as shown.

`this typeface`	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
`underscore`	Indicates a default setting.
this typeface	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option you can click or select.
this typeface	Highlights a file name or command.
Key + Key	Indicates keys that you must press simultaneously.
`{ }`	Indicates two or three choices; type one of them, not the braces.
`[ ]`	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
`|`	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
`...`	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

IBM

1 Web Query Web Services

You can interact with a variety of Web Query features using Web Services.

Topics:

- ❑ DB2 Web Query Software Development Kit
- ❑ What Is a Web Service?
- ❑ Components of a Web Service
- ❑ Using Web Query Web Services
- ❑ Web Query Web Services Architecture

DB2 Web Query Software Development Kit

DB2 Web Query provides a Software Development Kit for ISVs use to develop Web applications. The Software Development Kit is comprised of a set of Web Services functions that enables a developer to deliver DB2 Web Query content. For example, an ISV could use DB2 Web Query to develop an executive dashboard that interacts with DB2 Web Query content. Interaction could include parameterized reports, graphs, charts, and drill-down capabilities. A Software Development Kit application leverages the standard security defined in the DB2 Web Query environment.

What Is a Web Service?

A Web Service is a self-contained application that performs certain functions that you can publish on the Web based on open standards. A Web Service facilitates communication between applications because it is not platform-specific or language-based.

A Web Service functions like a black box, because you can provide input and expect certain output without knowing exactly how the results are achieved. Since Web Services are platform-neutral and comply with open standards, more businesses can interact with one another through integration of applications, even those written in different languages. For example, Windows applications can talk to UNIX applications, and Java can talk to Perl.

Most companies develop software for their specific business functions. However, due to increased associations among businesses, and international commercial collaboration, companies need a way to share information stored in separate computer systems. A Web Service makes it possible for applications within computer systems to communicate with each other.

Through a Web Service, business logic, processes, and data can be reused by different applications. This significantly reduces the amount of time it takes to develop applications by eliminating the need to replicate business functions.

Components of a Web Service

The main components of a Web Service are:

- **Extensible Markup Language (XML).** XML is a flexible text format that allows the exchange of data on the Internet, intranets, and elsewhere. It is a standard of the World Wide Web Consortium (W3C).

- **Simple Object Access Protocol (SOAP).** SOAP is a communications protocol designed for exchanging information in a distributed environment. It provides a messaging framework that is independent of implementation specifics, and it enables a program on one kind of operating system (such as OS/400) to communicate with a program on another kind of operating system (such as Linux). SOAP uses the Hypertext Transfer Protocol (HTTP) and XML to exchange information.

- **Web Services Description Language (WSDL).** WSDL, expressed in XML, describes how to access a Web Service and the operations that it will perform. WSDL, which was co-developed by Microsoft and IBM, describes the protocols and formats used by the Web Service.
- **Universal Description, Discovery, and Integration (UDDI).** UDDI is the component that enables businesses to find each other on the Web and make their systems interoperable for e-commerce. A business can list itself by name, product, location, or by the Web Service it offers. Information is entered on a registry server and then shared by servers in other businesses. Services are defined through a UDDI document called a Type Model or tModel. The tModel often contains a WSDL file that describes a SOAP interface to an XML Web Service.

You can use XML, SOAP, WSDL, and UDDI together to integrate Web-based applications. Use XML to tag the data, SOAP to transfer the data, WSDL to describe the available services, and UDDI to list the available services.

Using Web Query Web Services

In this section:

Web Query Web Services allow you to develop applications in the .NET or Java environments and perform Web Query functionality from it. A business can pass parameters from its own application to Web Query and retrieve output in the form of a formatted report or data. In Web Query reports, input is in the form of parameters and output is in the form of a formatted report.

Businesses benefit from Web Services because of their accessibility and efficiency. A Web Query Web Service allows businesses to use their own customized front-end interfaces to implement a variety of Web Query reporting capabilities.

Web Query provides the Web Query Web Services Enablement product, also known as the Publish option, to present Web Query reports as Web Services operations. It allows external applications built with Web Services supported languages (such as C++, C#, VB.NET, Java, and Flash Action Script) to integrate the report output. It also includes a set of generic Web Services operations that allow an application to integrate some basic Web Query functionality.

What Can You Do Through a Web Query Web Service?

You can do the following through a Web Query Web Service:

❏ Run Web Query reports.

❏ Determine the parameters for a report.

❏ Retrieve metadata information.

❏ Retrieve Domain information.

❏ Retrieve the report list.

❏ Pass a .NET data set to a Web Query report.

Web Query WSDL Creation

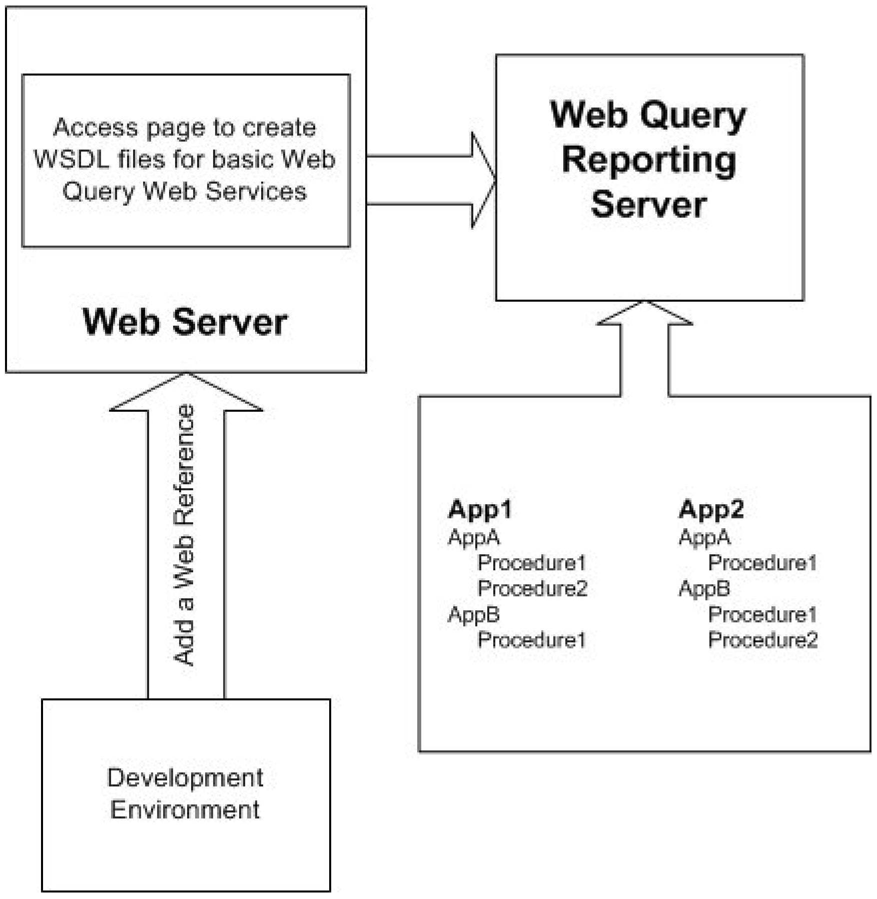
Web Query provides a link for WSDL creation, which describes the capabilities available through a Web Query Web Service. A WSDL file is an XML document that defines the input parameters for Web Query Web Services functions, the expected output, and the method for calling each function.

.NET or Java programmers can use this to develop their own applications.

Web Query Web Services Architecture

In this example, the developer creates a Web reference from a development environment to the WSDL (Web Services Description Language) file for the basic Web Query Web Service.

This Web Service allows the user to explore the Web Query Reporting Server. For instance, the user can access a list of the applications and procedures residing on the server. The developer can also run one of the existing procedures or send an ad hoc procedure to the server for execution.



2 Using the Web Query WSDL Utility

Web Query SDK provides the Web Query WSDL utility that creates a WSDL file that .NET and Java programmers can use to call the set of Web Query Web Services functions. The WSDL creation utility expedites application development because it automatically generates a WSDL file.

Topics:

Creating a WSDL File

A WSDL file is needed in development to call desired Web Query Web Services functions in either a .NET or Java environment. WSDL (Web Services Description Language) files are XML documents that are programming-language neutral and standards-based. They define input parameters for Web Services functions, expected output, and the messages used to call each function.

A valid 5733QU4 IBM DB2 Web Query Software Development Kit (SDK) license is required to generate the WSDL.

You can generate a WSDL file for Web Query by navigating to:

`http://`*`target_machine`*`[:`*`port`*`]/webquery/uddi/WebQuery.jsp?wsdl`

where:

`target_machine`

Is the name of the machine where Web Query is installed.

`port`

Is the port number used by Web Query.

If Web Query Report Broker is licensed, generate a WSDL for the Report Broker runschedule function for Version 1 Release 1 Modification 1 by navigating to:

`http://`*`target_machine`*`[:`*`port`*`]/webquery_rcaster/services`

If Web Query Report Broker is licensed, generate a WSDL for the Report Broker runschedule function for Version 1 Release 1 Modification 2 by navigating to:

`http://`*`target_machine`*`[:`*`port`*`]/webquery/services`

where:

`target_machine`

Is the name of the machine where Web Query is installed.

`port`

Is the port number used by Web Query.

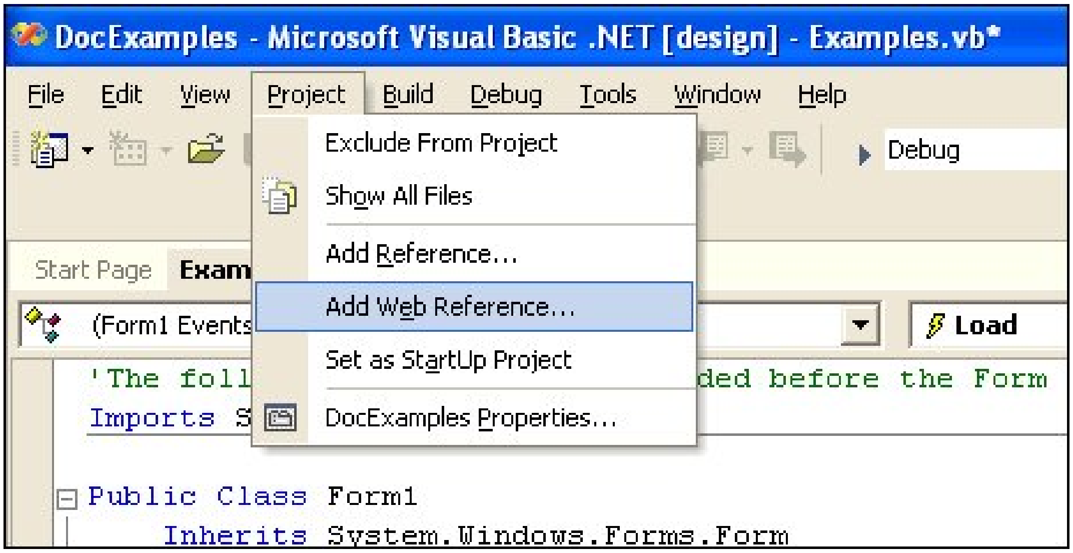
Consuming a Web Service in .NET

In order for a .NET program to communicate with a Web Service, the Web Services Description Language (WSDL) file must first be consumed. The consumption process reads the WSDL file and creates all the necessary classes and code to be used in the development of a program within .NET.

In .NET, perform the following steps:

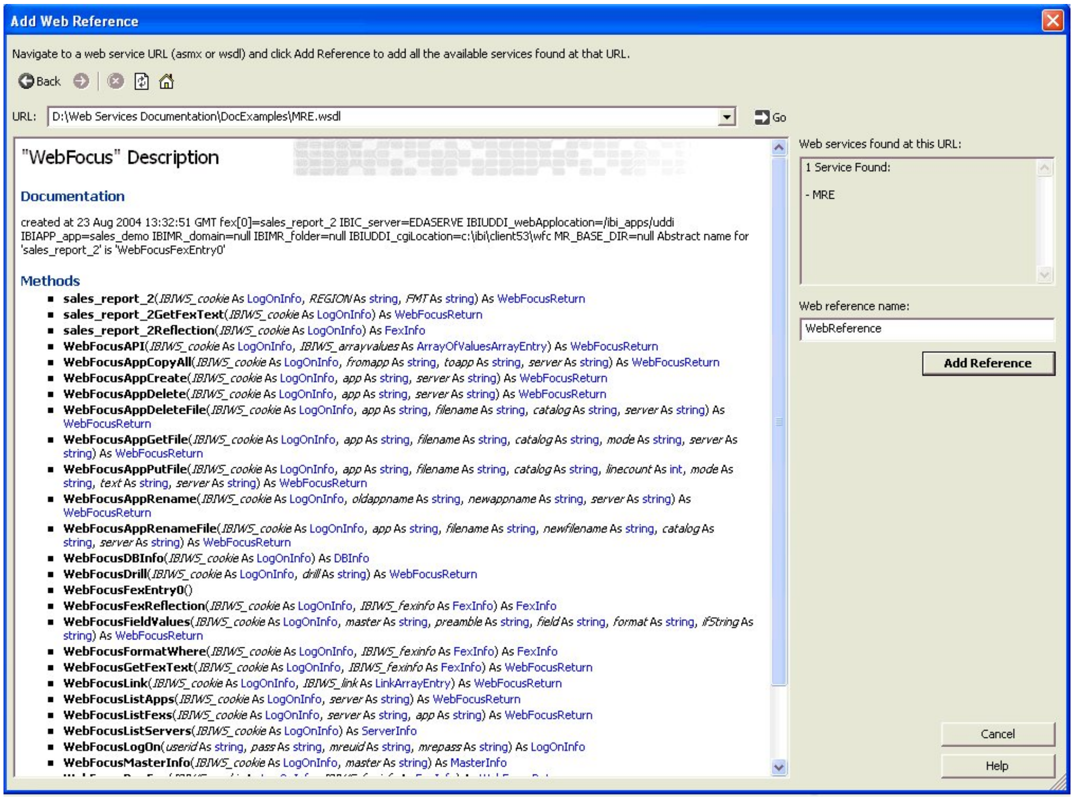
1. Select *Add Web Reference* from the Project menu.

   The Add Web Reference screen appears.



```
'The foll                                  ded before the Form
Imports S

Public Class Form1
    Inherits System.Windows.Forms.Form
```

2. Enter the location of the WSDL file.



Navigate to a web service URL (asmx or wsdl) and click Add Reference to add all the available services found at that URL.

URL: D:\Web Services Documentation\DocExamples\MRE.wsdl

"WebFocus" Description

Documentation

created at 23 Aug 2004 13:32:51 GMT fex[0]=sales_report_2 IBIC_server=EDASERVE IBIUDDI_webApplocation=/ibi_apps/uddi IBIAPP_app=sales_demo IBIMR_domain=null IBIMR_folder=null IBIUDDI_cgiLocation=c:\ibi\client53\wfc MR_BASE_DIR=null Abstract name for 'sales_report_2' is 'WebFocusFexEntry0'

Methods

- **sales_report_2**(*IBIWS_cookie* As LogOnInfo, *REGION* As string, *FMT* As string) As WebFocusReturn
- **sales_report_2GetFexText**(*IBIWS_cookie* As LogOnInfo) As WebFocusReturn
- **sales_report_2Reflection**(*IBIWS_cookie* As LogOnInfo) As FexInfo
- **WebFocusAPI**(*IBIWS_cookie* As LogOnInfo, *IBIWS_arrayvalues* As ArrayOfValuesArrayEntry) As WebFocusReturn
- **WebFocusAppCopyAll**(*IBIWS_cookie* As LogOnInfo, *fromapp* As string, *toapp* As string, *server* As string) As WebFocusReturn
- **WebFocusAppCreate**(*IBIWS_cookie* As LogOnInfo, *app* As string, *server* As string) As WebFocusReturn
- **WebFocusAppDelete**(*IBIWS_cookie* As LogOnInfo, *app* As string, *server* As string) As WebFocusReturn
- **WebFocusAppDeleteFile**(*IBIWS_cookie* As LogOnInfo, *app* As string, *filename* As string, *catalog* As string, *server* As string) As WebFocusReturn
- **WebFocusAppGetFile**(*IBIWS_cookie* As LogOnInfo, *app* As string, *filename* As string, *catalog* As string, *mode* As string, *server* As string) As WebFocusReturn
- **WebFocusAppPutFile**(*IBIWS_cookie* As LogOnInfo, *app* As string, *filename* As string, *catalog* As string, *linecount* As int, *mode* As string, *text* As string, *server* As string) As WebFocusReturn
- **WebFocusAppRename**(*IBIWS_cookie* As LogOnInfo, *oldappname* As string, *newappname* As string, *server* As string) As WebFocusReturn
- **WebFocusAppRenameFile**(*IBIWS_cookie* As LogOnInfo, *app* As string, *filename* As string, *newfilename* As string, *catalog* As string, *server* As string) As WebFocusReturn
- **WebFocusDBInfo**(*IBIWS_cookie* As LogOnInfo) As DBInfo
- **WebFocusDrill**(*IBIWS_cookie* As LogOnInfo, *drill* As string) As WebFocusReturn
- **WebFocusFexEntry0**()
- **WebFocusFexReflection**(*IBIWS_cookie* As LogOnInfo, *IBIWS_fexinfo* As FexInfo) As FexInfo
- **WebFocusFieldValues**(*IBIWS_cookie* As LogOnInfo, *master* As string, *preamble* As string, *field* As string, *format* As string, *ifString* As string) As WebFocusReturn
- **WebFocusFormatWhere**(*IBIWS_cookie* As LogOnInfo, *IBIWS_fexinfo* As FexInfo) As FexInfo
- **WebFocusGetFexText**(*IBIWS_cookie* As LogOnInfo, *IBIWS_fexinfo* As FexInfo) As WebFocusReturn
- **WebFocusLink**(*IBIWS_cookie* As LogOnInfo, *IBIWS_link* As LinkArrayEntry) As WebFocusReturn
- **WebFocusListApps**(*IBIWS_cookie* As LogOnInfo, *server* As string) As WebFocusReturn
- **WebFocusListFexs**(*IBIWS_cookie* As LogOnInfo, *server* As string, *app* As string) As WebFocusReturn
- **WebFocusListServers**(*IBIWS_cookie* As LogOnInfo) As ServerInfo
- **WebFocusLogOn**(*userid* As string, *pass* As string, *mreuid* As string, *mrepass* As string) As LogOnInfo
- **WebFocusMasterInfo**(*IBIWS_cookie* As LogOnInfo, *master* As string) As MasterInfo

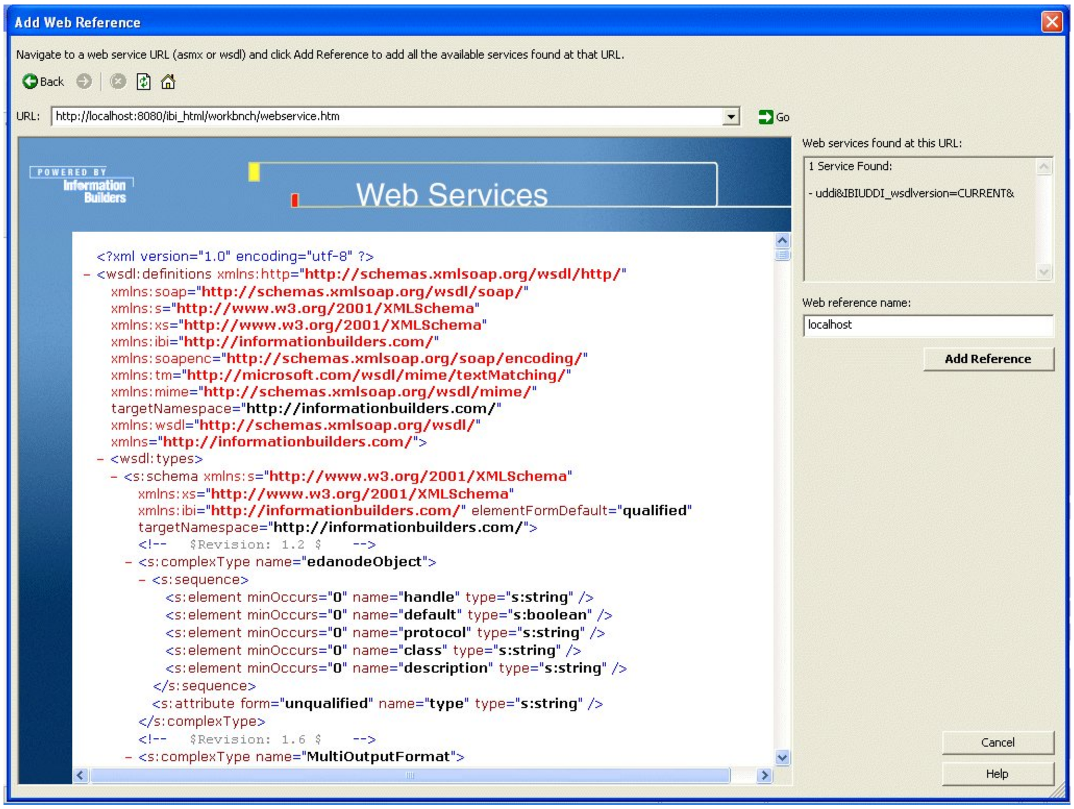
Web services found at this URL:

1 Service Found:

- MRE

Web reference name: WebReference

3. Once the WSDL file appears, click the *Add Reference* button.



Navigate to a web service URL (asmx or wsdl) and click Add Reference to add all the available services found at that URL.

URL: http://localhost:8080/ibi_html/workbnch/webservice.htm

Web services found at this URL:

1 Service Found:

- uddi&IBIUDDI_wsdlversion=CURRENT&

Web reference name: localhost

```
  <?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ibi="http://informationbuilders.com/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://informationbuilders.com/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns="http://informationbuilders.com/">
  - <wsdl:types>
    - <s:schema xmlns:s="http://www.w3.org/2001/XMLSchema"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:ibi="http://informationbuilders.com/" elementFormDefault="qualified"
        targetNamespace="http://informationbuilders.com/">
        <!--   $Revision: 1.2 $    -->
      - <s:complexType name="edanodeObject">
        - <s:sequence>
            <s:element minOccurs="0" name="handle" type="s:string" />
            <s:element minOccurs="0" name="default" type="s:boolean" />
            <s:element minOccurs="0" name="protocol" type="s:string" />
            <s:element minOccurs="0" name="class" type="s:string" />
            <s:element minOccurs="0" name="description" type="s:string" />
          </s:sequence>
          <s:attribute form="unqualified" name="type" type="s:string" />
        </s:complexType>
        <!--   $Revision: 1.6 $    -->
      - <s:complexType name="MultiOutputFormat">
```

Consuming a Web Service With Apache Axis

How to:

Consume a Web Service Using Apache Axis

In order for a Java program to communicate with a Web Service, the Web Services Description Language (WSDL) file must first be consumed. This means that the consumption process would read the WSDL file and create all the necessary classes and code to be used within the Java development environment. There are tools available which perform this consumption process. A commonly used tool is Apache Axis.

You can download Apache Axis from www.apache.org.

Procedure: How to Consume a Web Service Using Apache Axis

Perform the following steps to consume a Web Service using Apache Axis:

1. Create the following .bat file

```
SET WF=Axis_jar_file_directory
SET AXISJARS=%WF%axis.jar
SET AXISJARS=%AXISJARS%;%WF%axis-ant.jar
SET AXISJARS=%AXISJARS%;%WF%commons-discovery.jar
SET AXISJARS=%AXISJARS%;%WF%commons-logging.jar
SET AXISJARS=%AXISJARS%;%WF%jaxrpc.jar
SET AXISJARS=%AXISJARS%;%WF%log4j-1.2.l4.jar
SET AXISJARS=%AXISJARS%;%WF%mail.jar
SET AXISJARS=%AXISJARS%;%WF%activation.jar
SET AXISJARS=%AXISJARS%;%WF%saaj.jar
SET AXISJARS=%AXISJARS%;%WF%wsdl4j-1.5.1.jar
java -classpath %AXISJARS%;
    org.apache.axis.wsdl.WSDL2Java -v -s -p javademo WebFocus.wsdl
```

   where:

   `Axis_jar_file_directory`
   Is the location of the Axis .jar files.

   `javademo`
   Is the name of the package that will contain the Java code.

   `WebFocus.wsdl`
   Is the name of the WSDL file that was saved after being created with the WSDL creation utility. Ensure that the path to this file is correct.

   Note: The java command must be on one line. This is a Java requirement.

2. Run the .bat file. A sub-directory with a name identified by the package name in the .bat file is created.

   In the above example, a sub-directory called javademo is created containing the Java code to call the Web Service functions.

IBM

3 Web Query Web Services Structures

Web Query Web Services contain a set of structures to use as input to a Web Query Web Services function and output from a Web Query Web Services function.

Topics:

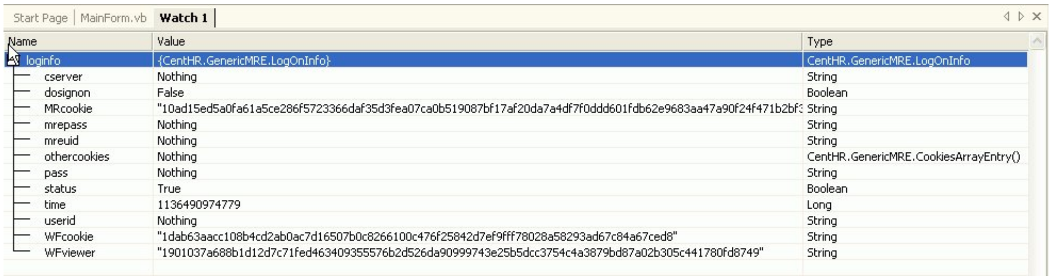
LogOnInfo (Authentication Structure)

LogOnInfo is a structure that contains the output values from the WebQueryLogOn function. It contains all of the Web Query authentication information. Since this structure is the first parameter of all the other Web Query Web Services functions, the LogOnInfo structure must first be set before any of the other Web Query Web Services functions can be run. It can be set through the WebQueryLogOn function or authentication can be performed on every Web Query Web Services function call by setting cserver, dosignon, mrepass, mreuid, userid, and pass.

Name	Type	Description
cserver	String	Node name of the Reporting Server as defined in the odin.cfg file.
dosignon	Boolean	True. Will authenticate on every Web Services function call. False. Will not perform authentication. This is the default.
MRcookie	String	Logon cookie for Web Query.
mrepass	String	Web Query password.
mreuid	String	Web Query Reporting user ID.
othercookies	*CookiesArrayEntry*	Custom cookies which get added to the context and are picked up in the Web Query variable table.
pass	String	Web Query Reporting Server password.
status	Boolean	True if logon is successful. False if logon is unsuccessful.
time	Long	Time of run in milliseconds since January 1, 1970.
userid	String	Web Query Reporting Server user ID.
WFcookie	String	Logon cookie for the Web Query Reporting Server.

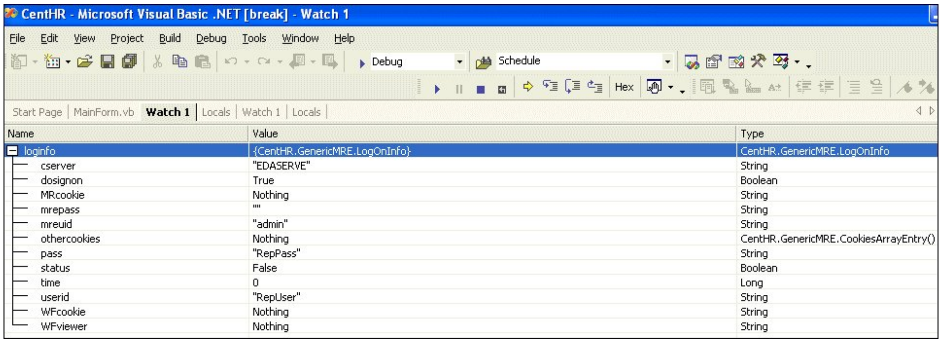
Name	Type	Description
WFviewer	String	Session cookie for the Web Query environment.

The following image shows an example of the LogOnInfo structure.



Name	Value	Type
loginfo	{CentHR.GenericMRE.LogOnInfo}	CentHR.GenericMRE.LogOnInfo
cserver	Nothing	String
dosignon	False	Boolean
MRcookie	"10ad15ed5a0fa61a5ce286f5723366daf35d3fea07ca0b519087bf17af20da7a4df7f0ddd601fdb62e9683aa47a90f24f471b2bf3	String
mrepass	Nothing	String
mreuid	Nothing	String
othercookies	Nothing	CentHR.GenericMRE.CookiesArrayEntry()
pass	Nothing	String
status	True	Boolean
time	1136490974779	Long
userid	Nothing	String
WFcookie	"1dab63aacc108b4cd2ab0ac7d16507b0c8266100c476f25842d7ef9fff78028a58293ad67c84a67ced8"	String
WFviewer	"1901037a688b1d12d7c71fed463409355576b2d526da90999743e25b5dcc3754c4a3879bd87a02b305c441780fd8749"	String

The following image shows an example of the LogOnInfo structure if it is used to authenticate on every Web Query Web Services function call.



Name	Value	Type
loginfo	{CentHR.GenericMRE.LogOnInfo}	CentHR.GenericMRE.LogOnInfo
cserver	"EDASERVE"	String
dosignon	True	Boolean
MRcookie	Nothing	String
mrepass	""	String
mreuid	"admin"	String
othercookies	Nothing	CentHR.GenericMRE.CookiesArrayEntry()
pass	"RepPass"	String
status	False	Boolean
time	0	Long
userid	"RepUser"	String
WFcookie	Nothing	String
WFviewer	Nothing	String

WebQueryReturn (Report Output Structure)

In this section:

WebQueryReturn is a structure that contains the output values from running a Web Query report through Web Query Web Services.

Name	Type	Description
binaryData	Byte array	Contains binary data. This contains the output of a Web Query report if the report is a graph with a format of JPG, JPEG, or PNG.
binaryDataLength	Integer	Length of *binaryData.*
ext	String	Suggested extension of file for output.
links	*LinkArrayEntry*	Array of links that link information to drill-down reports, graphs, Cascading Style Sheets, and JavaScripts. Use the value to determine the format of the report being executed. Evaluate the LinkArrayEntry structure's Type Value as described in *LinkArrayEntry (Report Links Structure)* on page 38.

Name	**Type**	**Description**

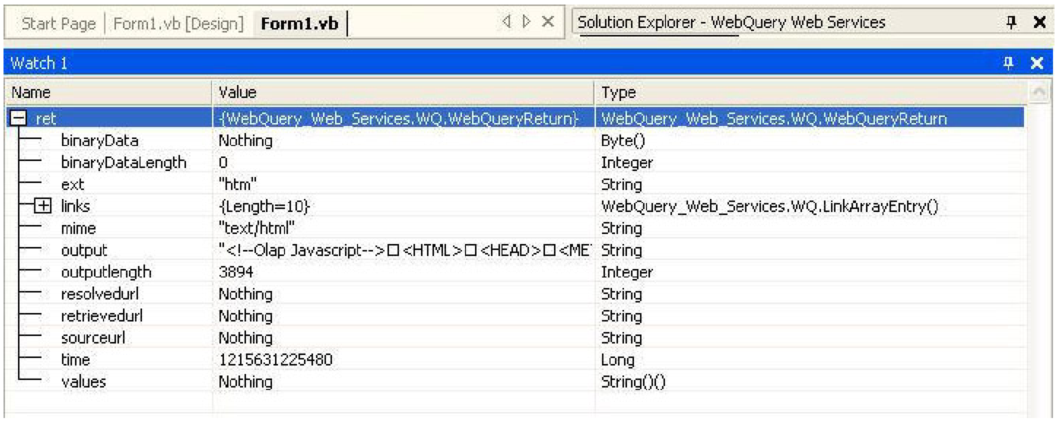
mime	String	Type of output being returned.
		`text/html`
		Is HTML output.
		`application/vnd.ms-excel`
		Is Excel output.
		`application/pdf`
		Is PDF output.
		`image/jpg`
		Is a graph or an image in .jpg format.
		`image/jpeg`
		Is a graph or an image in .jpeg format.
		`image/png`
		Is a graph or an image in .png format.
		`image/svg+xml`
		Is a graph or an image in .svg format.
		`text/xml`
		Is XML output.

output	String	Contains the output of a Web Query report for all formats except JPG, JPEG, and PNG.
outputlength	Integer	Size of *output* in bytes.
resolvedurl	String	URL of the output component.
retrievedurl	String	URL used by Web Query to retrieve the output component.
sourceurl	String	Location of the output component within the Web Query environment.
time	Long	Time of run in milliseconds since January 1, 1970.

Name	**Type**	**Description**
values	String array	Values of the output of a Web Query report if the mime value is text/xml.

HTML Format

The following image shows information that is returned to the WebQueryReturn structure if the output of a Web Query report is in HTML format.



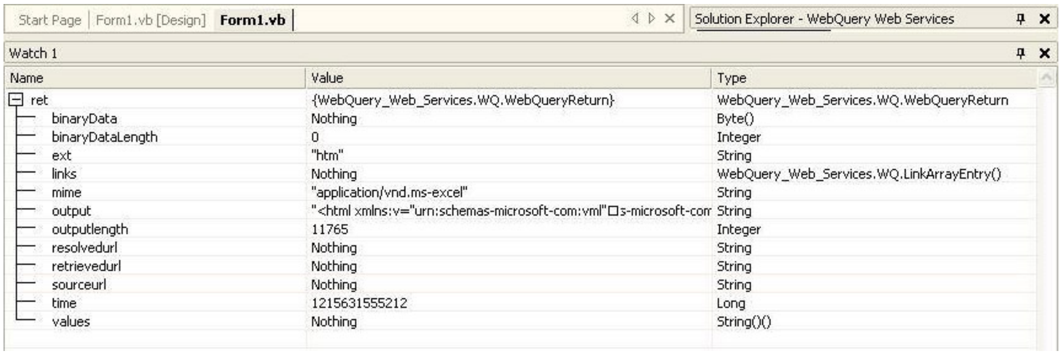
Start Page | Form1.vb [Design] | **Form1.vb** — Solution Explorer - WebQuery Web Services

Watch 1

Name	Value	Type
ret	{WebQuery_Web_Services.WQ.WebQueryReturn}	WebQuery_Web_Services.WQ.WebQueryReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
ext	"htm"	String
links	{Length=10}	WebQuery_Web_Services.WQ.LinkArrayEntry()
mime	"text/html"	String
output	"<!--Olap Javascript-->□<HTML>□<HEAD>□<ME	String
outputlength	3894	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	Nothing	String
time	1215631225480	Long
values	Nothing	String()()

Excel Format

The following image shows information that is returned to the WebQueryReturn structure if the output of a Web Query report is in Excel format.



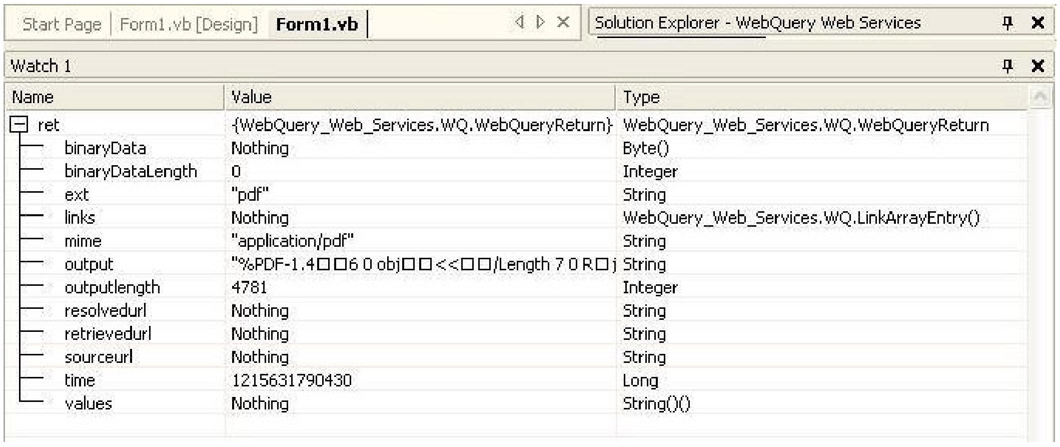
Start Page | Form1.vb [Design] | **Form1.vb** — Solution Explorer - WebQuery Web Services

Watch 1

Name	Value	Type
ret	{WebQuery_Web_Services.WQ.WebQueryReturn}	WebQuery_Web_Services.WQ.WebQueryReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
ext	"htm"	String
links	Nothing	WebQuery_Web_Services.WQ.LinkArrayEntry()
mime	"application/vnd.ms-excel"	String
output	"<html xmlns:v="urn:schemas-microsoft-com:vml"□s-microsoft-com	String
outputlength	11765	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	Nothing	String
time	1215631555212	Long
values	Nothing	String()()

PDF Format

The following image shows information that is returned to the WebQueryReturn structure if the output of a Web Query report is in PDF format.



```
Name                  Value                                     Type
⊟ ret                 {WebQuery_Web_Services.WQ.WebQueryReturn} WebQuery_Web_Services.WQ.WebQueryReturn
   binaryData         Nothing                                   Byte()
   binaryDataLength   0                                         Integer
   ext                "pdf"                                     String
   links              Nothing                                   WebQuery_Web_Services.WQ.LinkArrayEntry()
   mime               "application/pdf"                         String
   output             "%PDF-1.4□□6 0 obj□□<<□□/Length 7 0 R□j   String
   outputlength       4781                                      Integer
   resolvedurl        Nothing                                   String
   retrievedurl       Nothing                                   String
   sourceurl          Nothing                                   String
   time               1215631790430                             Long
   values             Nothing                                   String()()
```

Graph Output

In this section:

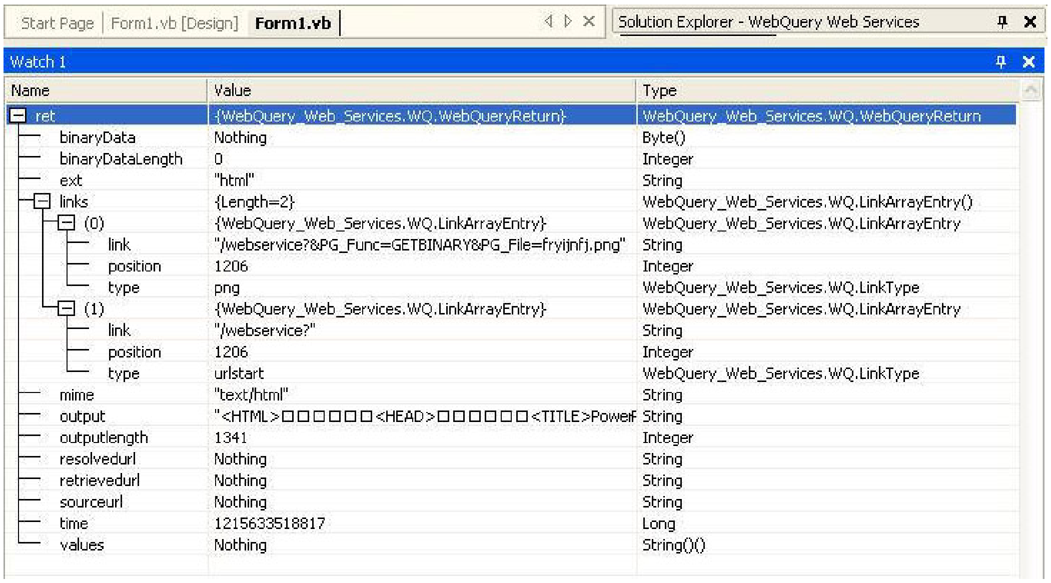
Retrieving Graph Output From an Application Server (.jpg Format)

Retrieving Graph Output From an Application Server (.JPEG format)

Retrieving Graph Output From an Application Server (.PNG format)

Retrieving Graph Output From an Application Server (.SVG format)

The following image shows information that is returned to the WebQueryReturn structure when you run a graph. At this point, the .jpg, .jpeg, .png, or .svg file is not returned to the output value. The graph resides on the application server. Note that WebQueryReturn shows the existence of two links of a type LinkArrayEntry (see LinkArrayEntry). The WebQueryLink function should be used to retrieve the graph from the application server.


Start Page | Form1.vb [Design] | **Form1.vb**

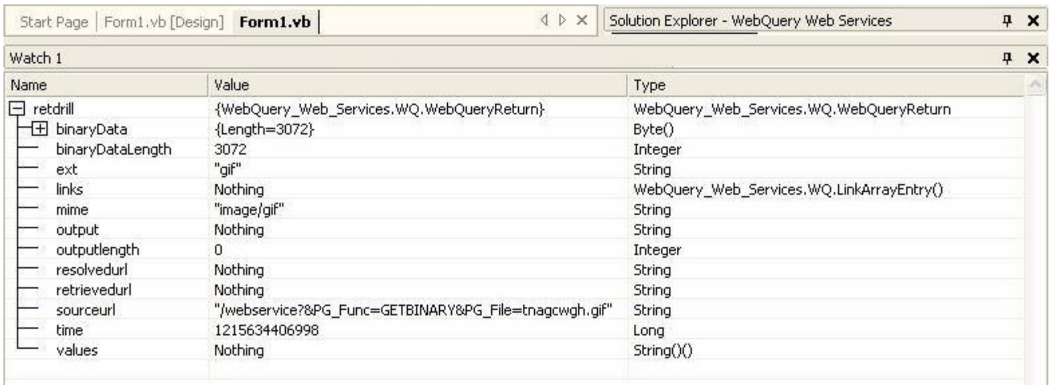
Solution Explorer - WebQuery Web Services

Watch 1

Name	Value	Type
ret	{WebQuery_Web_Services.WQ.WebQueryReturn}	WebQuery_Web_Services.WQ.WebQueryReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
ext	"html"	String
links	{Length=2}	WebQuery_Web_Services.WQ.LinkArrayEntry()
(0)	{WebQuery_Web_Services.WQ.LinkArrayEntry}	WebQuery_Web_Services.WQ.LinkArrayEntry
link	"/webservice?&PG_Func=GETBINARY&PG_File=fryijnfj.png"	String
position	1206	Integer
type	png	WebQuery_Web_Services.WQ.LinkType
(1)	{WebQuery_Web_Services.WQ.LinkArrayEntry}	WebQuery_Web_Services.WQ.LinkArrayEntry
link	"/webservice?"	String
position	1206	Integer
type	urlstart	WebQuery_Web_Services.WQ.LinkType
mime	"text/html"	String
output	"<HTML>□□□□□□<HEAD>□□□□□□<TITLE>PoweF	String
outputlength	1341	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	Nothing	String
time	1215633518817	Long
values	Nothing	String()()

Retrieving Graph Output From an Application Server (.jpg Format)

The following image shows information that is returned to the WebQueryReturn structure after the WebQueryLink function is used to retrieve a graph in .jpg format from the application server. Note that BinaryData and BinaryDataLength get populated.


Start Page | Form1.vb [Design] | **Form1.vb**

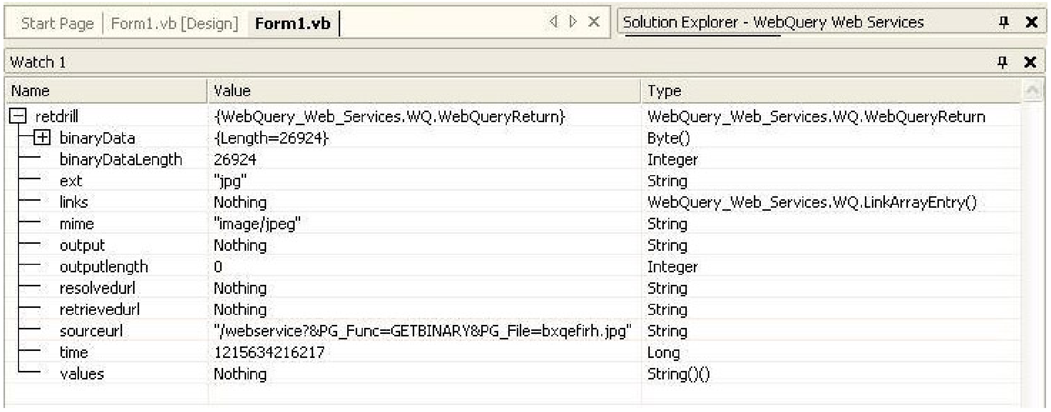
Solution Explorer - WebQuery Web Services

Watch 1

Name	Value	Type
retdrill	{WebQuery_Web_Services.WQ.WebQueryReturn}	WebQuery_Web_Services.WQ.WebQueryReturn
binaryData	{Length=3072}	Byte()
binaryDataLength	3072	Integer
ext	"gif"	String
links	Nothing	WebQuery_Web_Services.WQ.LinkArrayEntry()
mime	"image/gif"	String
output	Nothing	String
outputlength	0	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	"/webservice?&PG_Func=GETBINARY&PG_File=tnagcwgh.gif"	String
time	1215634406998	Long
values	Nothing	String()()

Retrieving Graph Output From an Application Server (.JPEG format)

The following image shows information that is returned to the WebQueryReturn structure after the WebQueryLink function is used to retrieve a graph in .jpeg format from the application server. Note that BinaryData and BinaryDataLength get populated.


Start Page | Form1.vb [Design] | **Form1.vb**

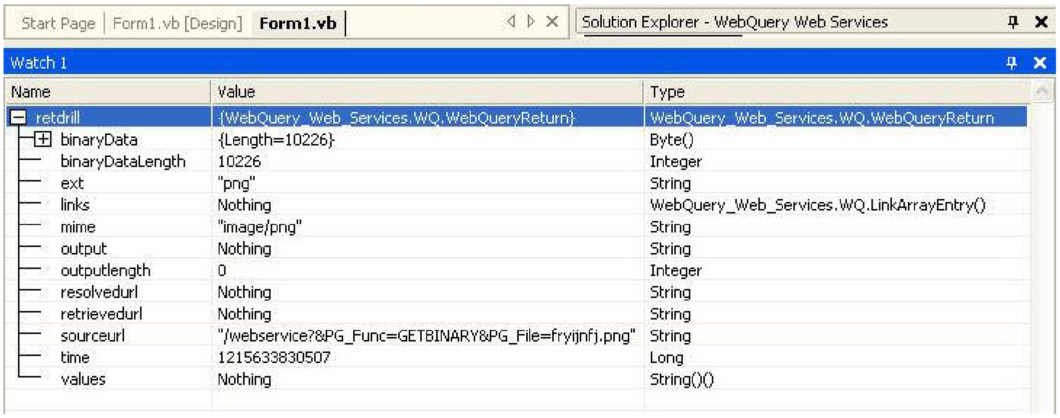
Solution Explorer - WebQuery Web Services

Watch 1

Name	Value	Type
retdrill	{WebQuery_Web_Services.WQ.WebQueryReturn}	WebQuery_Web_Services.WQ.WebQueryReturn
binaryData	{Length=26924}	Byte()
binaryDataLength	26924	Integer
ext	"jpg"	String
links	Nothing	WebQuery_Web_Services.WQ.LinkArrayEntry()
mime	"image/jpeg"	String
output	Nothing	String
outputlength	0	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	"/webservice?&PG_Func=GETBINARY&PG_File=bxqefirh.jpg"	String
time	1215634216217	Long
values	Nothing	String()()

Retrieving Graph Output From an Application Server (.PNG format)

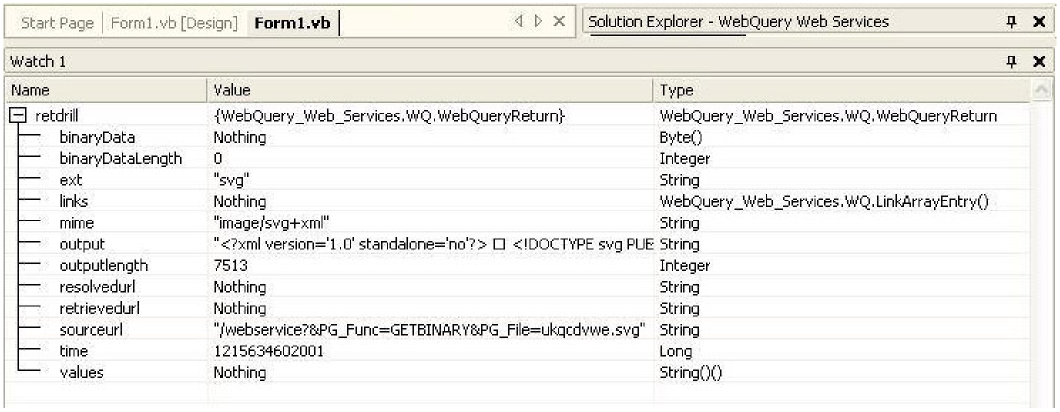
The following image shows information that is returned to the WebQueryReturn structure after the WebQueryLink function is used to retrieve a graph in .png format from the application server. Note that BinaryData and BinaryDataLength get populated.



Name	Value	Type
retdrill	{WebQuery_Web_Services.WQ.WebQueryReturn}	WebQuery_Web_Services.WQ.WebQueryReturn
binaryData	{Length=10226}	Byte()
binaryDataLength	10226	Integer
ext	"png"	String
links	Nothing	WebQuery_Web_Services.WQ.LinkArrayEntry()
mime	"image/png"	String
output	Nothing	String
outputlength	0	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	"/webservice?&PG_Func=GETBINARY&PG_File=fryijnfj.png"	String
time	1215633830507	Long
values	Nothing	String()()

Retrieving Graph Output From an Application Server (.SVG format)

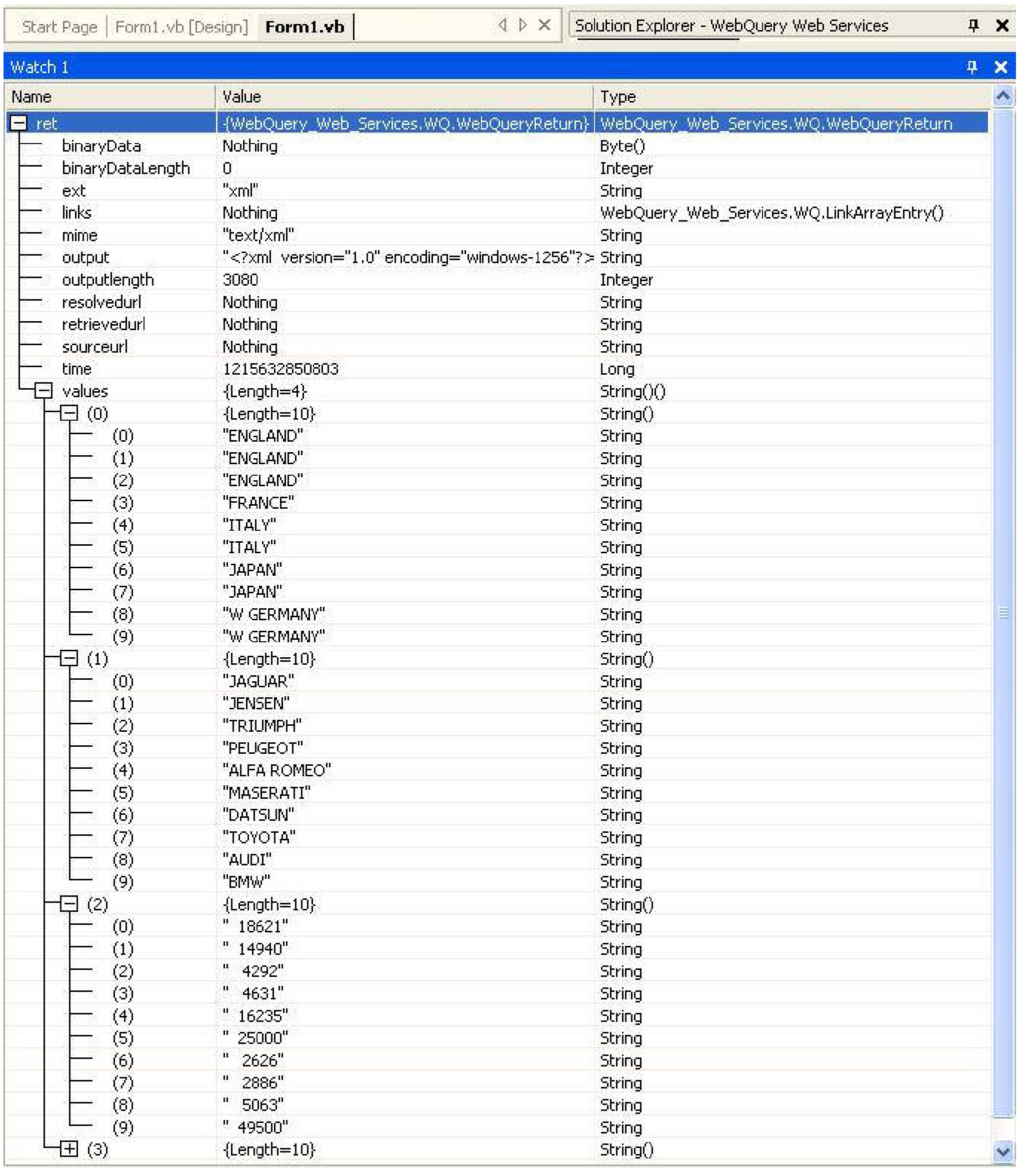
The following image shows information that is returned to the WebQueryReturn structure after the WebQueryLink function is used to retrieve a graph in .svg format from the application server. Note that Output and OutputLength get populated.



Name	Value	Type
retdrill	{WebQuery_Web_Services.WQ.WebQueryReturn}	WebQuery_Web_Services.WQ.WebQueryReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
ext	"svg"	String
links	Nothing	WebQuery_Web_Services.WQ.LinkArrayEntry()
mime	"image/svg+xml"	String
output	"<?xml version='1.0' standalone='no'?> ☐ <!DOCTYPE svg PUE	String
outputlength	7513	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	"/webservice?&PG_Func=GETBINARY&PG_File=ukqcdvwe.svg"	String
time	1215634602001	Long
values	Nothing	String()()

XML Format

The following image shows information that is returned to the WebQueryReturn structure if the output of a Web Query report is in XML format. Note that Values, XML, and XMLlength get populated.



Name	Value	Type
ret	{WebQuery_Web_Services.WQ.WebQueryReturn}	WebQuery_Web_Services.WQ.WebQueryReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
ext	"xml"	String
links	Nothing	WebQuery_Web_Services.WQ.LinkArrayEntry()
mime	"text/xml"	String
output	"<?xml version="1.0" encoding="windows-1256"?>	String
outputlength	3080	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	Nothing	String
time	1215632850803	Long
values	{Length=4}	String()()
(0)	{Length=10}	String()
(0)	"ENGLAND"	String
(1)	"ENGLAND"	String
(2)	"ENGLAND"	String
(3)	"FRANCE"	String
(4)	"ITALY"	String
(5)	"ITALY"	String
(6)	"JAPAN"	String
(7)	"JAPAN"	String
(8)	"W GERMANY"	String
(9)	"W GERMANY"	String
(1)	{Length=10}	String()
(0)	"JAGUAR"	String
(1)	"JENSEN"	String
(2)	"TRIUMPH"	String
(3)	"PEUGEOT"	String
(4)	"ALFA ROMEO"	String
(5)	"MASERATI"	String
(6)	"DATSUN"	String
(7)	"TOYOTA"	String
(8)	"AUDI"	String
(9)	"BMW"	String
(2)	{Length=10}	String()
(0)	" 18621"	String
(1)	" 14940"	String
(2)	" 4292"	String
(3)	" 4631"	String
(4)	" 16235"	String
(5)	" 25000"	String
(6)	" 2626"	String
(7)	" 2886"	String
(8)	" 5063"	String
(9)	" 49500"	String
(3)	{Length=10}	String()

FexInfo (Run Report Structure)

In this section:

Running a Report

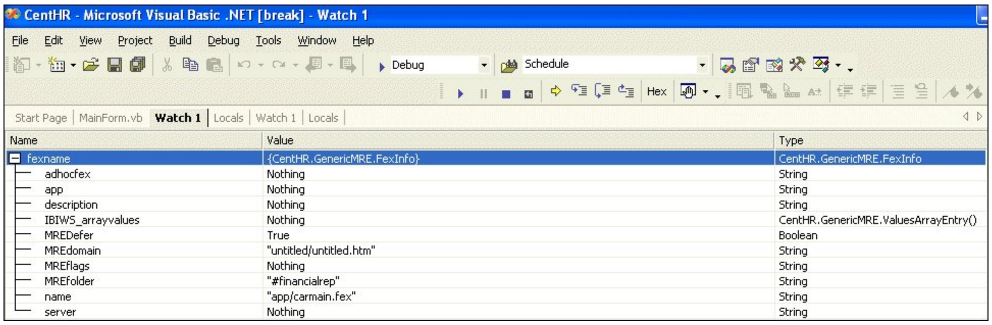
FexInfo is the parent structure that contains input information about a Web Query report that will be used by certain Web Query Web Services functions.

Name	Type	Description
description	String	Long name of the Web Query report. It is populated through the WebQueryFexReflection function.
IBIWS_arrayvalues	*ValuesArrayEntry*	Array of input parameters for the Web Query report.
MREdomain	String	HREF of the domain. This can be obtained by viewing the domain properties of the domain within Web Query or by getting a list of domains by using the Web Services function *MREGetUserDomains.*
MREflags	String	Web Query report flags are populated through the WebQueryFexReflection function.
MREfolder	String	HREF of the Standard Reports group. This can be obtained by viewing the Standard Reports group properties of the Standard Reports group within Web Query, or by getting a list of Standard Reports groups by using the Web Services function MREOpenDomain.

Name	Type	Description
name	String	HREF of the Web Query report if the report resides in Web Query. This can be obtained by viewing the file name property from the Standard Reports properties of the Web Query report within Web Query or by getting a list of Web Query reports by using the Web Services function MREOpenDomain.
server	String	Name of the Web Query Reporting Server used for running the Web Query report.

Running a Report

The following image shows an example of information that must be set in order to run a Web Query report using the WebQueryRunFex function.


CentHR - Microsoft Visual Basic .NET [break] - Watch 1
File Edit View Project Build Debug Tools Window Help
Debug
Schedule
Hex
Start Page | MainForm.vb | Watch 1 | Locals | Watch 1 | Locals

Name	Value	Type
fexname	{CentHR.GenericMRE.FexInfo}	CentHR.GenericMRE.FexInfo
adhocfex	Nothing	String
app	Nothing	String
description	Nothing	String
IBIWS_arrayvalues	Nothing	CentHR.GenericMRE.ValuesArrayEntry()
MREDefer	True	Boolean
MREdomain	"untitled/untitled.htm"	String
MREflags	Nothing	String
MREfolder	"#financialrep"	String
name	"app/carmain.fex"	String
server	Nothing	String

MREReturn (Web Query Functions Return Structure)

In this section:

Return Codes

MREReturn is a structure that contains the XML returned from MREGetUserDomains and MREOpenDomain functions. Functions pertaining to listing domain information would return the results in this structure.

Name	Type	Description
rc	Integer	Return code of the function. RC=1000 signifies that the function call was successful. For details, see *Return Codes* on page 35.
time	Long	Time of run in milliseconds since January 1, 1970.
xml	String	XML that is returned as the result of the function calls.

Return Codes

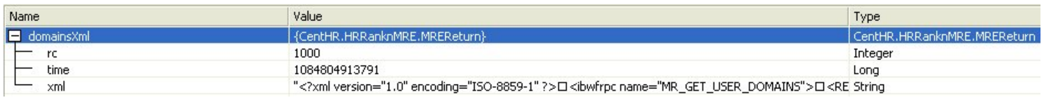
The following is a valid list of return codes.

```
1000 = ERROR_MR_NO_ERROR
1001 = ERROR_MR_NO_REPOSITORY_FILE
1002 = ERROR_MR_NO_OPEN
1003 = ERROR_MR_NO_USER_ID
1004 = ERROR_MR_NO_USER
1005 = ERROR_MR_DOMAIN_UNKNOWN
1006 = ERROR_MR_NO_DOMAIN
1007 = ERROR_MR_ACTION_UNKNOWN
1008 = ERROR_MR_MISSING_STRING
1009 = ERROR_MR_GROUP_UNKNOWN
1010 = ERROR_MR_FEX_NOT_FOUND
1011 = ERROR_MR_URL_ITEM
1012 = ERROR_MR_CANT_OPEN_FEX
1013 = ERROR_MR_BAD_FORCEDNAME
1014 = ERROR_MR_SUB_ACTION_UNKNOWN
1015 = ERROR_MR_CANT_DELETE_FEX
1016 = ERROR_MR_INTERNALTABLE_ERROR
1017 = ERROR_MR_CANT_OPEN_FOLDER
1018 = ERROR_MR_CREATE_FILE
1019 = ERROR_MR_BAD_WRITE
1020 = ERROR_NO_INCLUDE
1021 = ERROR_MR_BADDRILL
1022 = ERROR_MR_CANT_FIND_FOLDER
1023 = ERROR_MR_PASSWORD
1024 = ERROR_MR_ENCRIPT
1025 = ERROR_MR_NOT_ADMINISTRATOR
1026 = ERROR_MR_BAD_GROUP
1027 = ERROR_MR_BAD_DIR
1028 = ERROR_MR_MAX_USER
1029 = ERROR_MR_BAD_READ
1030 = ERROR_MR_CANT_FULFILL
1031 = ERROR_MR_CANT_MOVE_FEX
1032 = ERROR_MR_USER_REQUIRED
1033 = ERROR_MR_RERUN_ERROR
1034 = ERROR_MR_NO_FILE_EXTENSION
1035 = ERROR_MR_ITEM_EXISTS
```

```
1036 = ERROR_MR_DIFF_FOLDER_TYPE
1037 = ERROR_MR_FOLDER_HAS_RO_FOLDER
1038 = ERROR_MR_PASSWORD_EXPIRED
1039 = ERROR_MR_USER_EXISTS
1040 = ERROR_MR_ADD_USER_GROUP
1041 = ERROR_MR_USER_CANT_SAVE
1042 = ERROR_MR_BAD_ROLE
1043 = ERROR_MR_ROLE_SUPPORT
1044 = ERROR_MR_CANT_DELETE_BASE_ROLE
1045 = ERROR_MR_CANT_DELETE_USED_ROLE
1046 = ERROR_MR_CANT_PARSE
1047 = ERROR_MR_CANT_OPEN_DOMAIN_FOR_USER
1048 = ERROR_MR_NO_URL_ITEM
1060 = ERROR_OLAP_INTERNAL_ERROR
1061 = ERROR_OLAP_BAD_WRITE
1062 = ERROR_OLAP_BAD_READ
1063 = ERROR_OLAP_BAD_FOCEXEC
1064 = ERROR_OLAP_XML_PARSING
1065 = ERROR_OLAP_OLAPRULES
1066 = ERROR_NO_OLAPRULES
1067 = ERROR_IN_TRANSFORMATION
1068 = ERROR_NO_GRAPHRULES
1080 = ERROR_DST_MRE_UADMIN_INIT
1081 = ERROR_DST_AUTH_INIT
1082 = ERROR_DST_CREATE_USER
1083 = ERROR_DST_DELETE_USER
1084 = ERROR_DST_DESTROY_USER
1085 = ERROR_DST_SET_USER_FLAGS
```

```
1086 = ERROR_DST_SET_USER_PWD
1087 = ERROR_DST_CREATE_GROUP
1088 = ERROR_DST_DELETE_GROUP
1089 = ERROR_DST_ADD_GROUP_USER
1090 = ERROR_DST_DELETE_GROUP_USER
1091 = ERROR_DST_IS_ADMIN
1092 = ERROR_DST_AUTH_FAILED
1099 = FAIL_LOAD_DRIVER_FACTORY
1100 = ERROR_MR_NOT_IMPLEMENTED
1110 = ERROR_REALM_AUTHENTICATION
1111 = ERROR_REALM_GET_USERS
1112 = ERROR_REALM_GET_USERS_FOR_DOMAIN
1113 = ERROR_REALM_GET_DOMAINS_FOR_USER
1114 = ERROR_REALM_DELETE_GROUP
1115 = ERROR_REALM_DELETE_GROUPS
1116 = ERROR_REALM_DELETE_DOMAINS_FROM_GROUP
1117 = ERROR_REALM_CREATE_NEW_GROUP
1118 = ERROR_REALM_ADD_DOMAINS_TO_GROUP
1119 = ERROR_REALM_REMOVE_GROUPS_FROM_USER
1120 = ERROR_REALM_REMOVE_USERS_FROM_GROUP
1121 = ERROR_REALM_ADD_GROUPS_TO_USER
1122 = ERROR_REALM_UPDATE_GROUP
1123 = ERROR_REALM_ADD_USERS_TO_GROUP
1124 = ERROR_REALM_CREATE_NEW_USER
1125 = ERROR_REALM_DELETE_USER
1126 = ERROR_REALM_DELETE_USERS
1127 = ERROR_REALM_UPDATE_USER
1128 = ERROR_REALM_DELETE_ROLE
1129 = ERROR_REALM_CREATE_UPDATE_ROLE
1130 = ERROR_REALM_GET_DOMAINS_TO_ADMIN
1131 = ERROR_REALM_GET_GROUPS_TO_ADMIN
1132 = ERROR_REALM_GET_GROUPS_DOMAINS_TO_ADMIN
1133 = ERROR_REALM_REMOVE_GROUPS_DOMAINS_TO_ADMIN
1134 = ERROR_REALM_SET_GROUPS_DOMAINS_TO_ADMIN
1135 = ERROR_REALM_IS_USER_GROUP_ADMIN
1136 = ERROR_REALM_IS_USER_DOMAIN_ADMIN
1137 = ERROR_REALM_IS_USER_SHARED
1138 = ERROR_REALM_CREATE_NEW_DOMAIN
1139 = ERROR_REALM_RENAME_DOMAIN
```

The following image shows an example of information that is returned in the MREReturn structure from making a MREGetUserDomains function call.



Name	Value	Type
domainsXml	{CentHR.HRRanknMRE.MREReturn}	CentHR.HRRanknMRE.MREReturn
rc	1000	Integer
time	1084804913791	Long
xml	"<?xml version="1.0" encoding="ISO-8859-1" ?>☐<ibwfrpc name="MR_GET_USER_DOMAINS">☐<RE	String

LinkArrayEntry (Report Links Structure)

In this section:

LinkArrayEntry is a structure that contains the link information for drill-down reports, graphs, images, Cascading Style Sheets, JavaScripts, and ibiOptions. It is mainly used to drill down to another Web Query report or the location of a graph on the application server. The link for a drill-down report or graph is then used as input to the WebQueryLink function. The LinkArrayEntry array contains pairs of links for each drill-down report line or graph. One of the link entries has a value for type 'urlstart'. This link entry is for information purposes only and should not be used as input to the WebQueryLink function. It is used to indicate the starting position of the link within the HTML.

Name	Type	Description
link	String	If the type value is 'urlstart' then the link value will be '/webservice?'. If the type value is 'optionsstart' then the link value will be 'var ibiOptions = new Array('. If the type is 'css' then the link value will contain the name of the .css file used in the report. If the type is 'js' then the link value will contain the location of the JavaScript file. Otherwise, the link value will contain the link to the drill-down report or graph for use by the WebQueryLink function.
position	Integer	Character offset from start of HTML of this link.

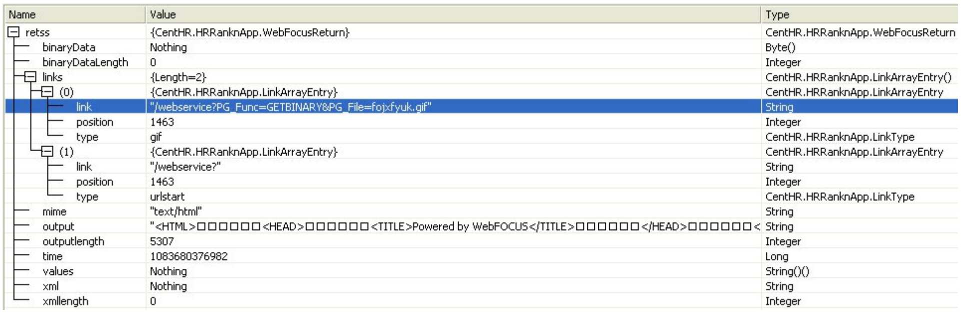
Name	Type	Description

type	*LinkType*	The type of link being returned.
		`jpg`
		Is a .jpg file.
		`jpeg`
		Is a .jpeg file.
		`png`
		Is a .png file.
		`svg`
		Is an .svg file.
		`js`
		Is a JavaScript file.
		`css`
		Is a Cascading Style Sheet.
		`urlstart`
		Is the starting position of each link within the HTML output.
		`optionsstart`
		Is the starting position within the HTML output of the list of ibiOptions that must be invoked. An OLAP-enabled report would have this link populated.
		`fexdrill`
		Is a drill-down report.
		`base`
		Is the starting position of <BASE within the HTML.
		`baseurlstart`
		Is the starting position of the Base URL within the HTML.

LinkType is an enumeration of valid values.

Graph in .jpg Format

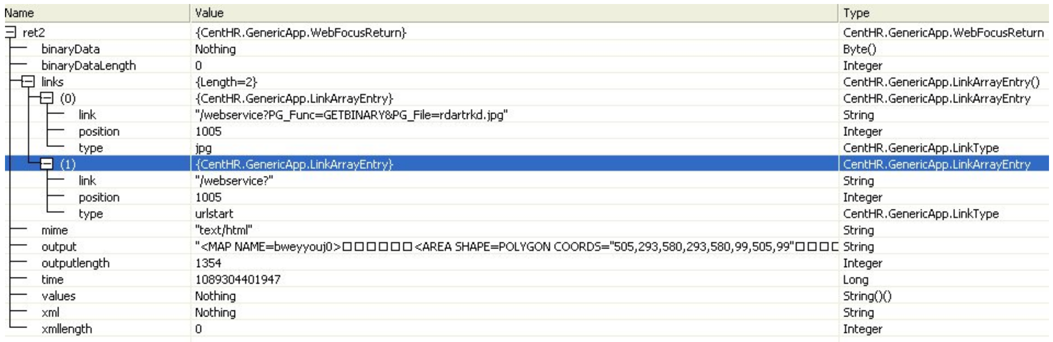
The following shows links contained in the WebQueryReturn structure for a graph in .jpg format.



Name	Value	Type
retss	{CentHR.HRRanknApp.WebFocusReturn}	CentHR.HRRanknApp.WebFocusReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
links	{Length=2}	CentHR.HRRanknApp.LinkArrayEntry()
(0)	{CentHR.HRRanknApp.LinkArrayEntry}	CentHR.HRRanknApp.LinkArrayEntry
link	"/webservice?PG_Func=GETBINARY&PG_File=fojxfyuk.gif"	String
position	1463	Integer
type	gif	CentHR.HRRanknApp.LinkType
(1)	{CentHR.HRRanknApp.LinkArrayEntry}	CentHR.HRRanknApp.LinkArrayEntry
link	"/webservice?"	String
position	1463	Integer
type	urlstart	CentHR.HRRanknApp.LinkType
mime	"text/html"	String
output	"<HTML>□□□□□□<HEAD>□□□□□□<TITLE>Powered by WebFOCUS</TITLE>□□□□□□</HEAD>□□□□□□<	String
outputlength	5307	Integer
time	1083680376982	Long
values	Nothing	String()()
xml	Nothing	String
xmllength	0	Integer

Graph in .JPEG Format

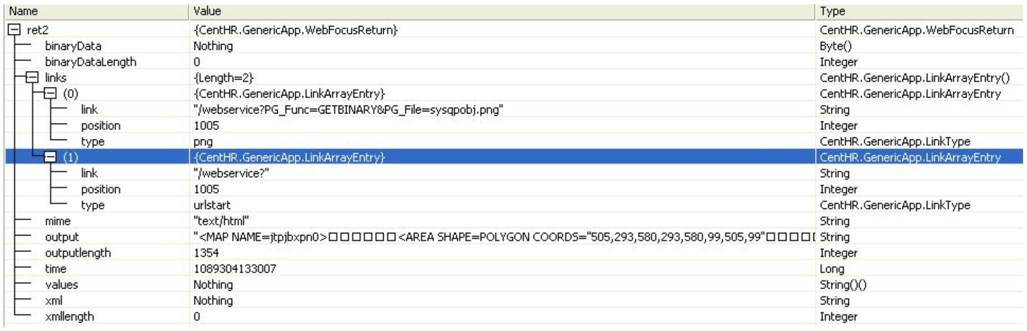
The following shows links contained in the WebQueryReturn structure for a graph in .jpeg format.



Name	Value	Type
ret2	{CentHR.GenericApp.WebFocusReturn}	CentHR.GenericApp.WebFocusReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
links	{Length=2}	CentHR.GenericApp.LinkArrayEntry()
(0)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"/webservice?PG_Func=GETBINARY&PG_File=rdartrkd.jpg"	String
position	1005	Integer
type	jpg	CentHR.GenericApp.LinkType
(1)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"/webservice?"	String
position	1005	Integer
type	urlstart	CentHR.GenericApp.LinkType
mime	"text/html"	String
output	"<MAP NAME=bweyyouj0>□□□□□□<AREA SHAPE=POLYGON COORDS="505,293,580,293,580,99,505,99"□□□□	String
outputlength	1354	Integer
time	1089304401947	Long
values	Nothing	String()()
xml	Nothing	String
xmllength	0	Integer

Graph in .PNG Format

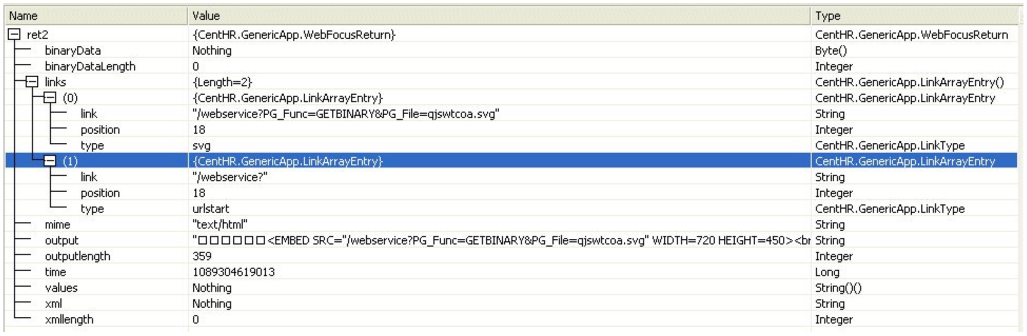
The following shows links contained in the WebQueryReturn structure for a graph in .png format.



Name	Value	Type
ret2	{CentHR.GenericApp.WebFocusReturn}	CentHR.GenericApp.WebFocusReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
links	{Length=2}	CentHR.GenericApp.LinkArrayEntry()
(0)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"/webservice?PG_Func=GETBINARY&PG_File=sysqpobj.png"	String
position	1005	Integer
type	png	CentHR.GenericApp.LinkType
(1)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"/webservice?"	String
position	1005	Integer
type	urlstart	CentHR.GenericApp.LinkType
mime	"text/html"	String
output	"<MAP NAME=jtpjbxpn0>□□□□□□<AREA SHAPE=POLYGON COORDS="505,293,580,293,580,99,505,99"□□□□	String
outputlength	1354	Integer
time	1089304133007	Long
values	Nothing	String()()
xml	Nothing	String
xmllength	0	Integer

Graph in .SVG Format

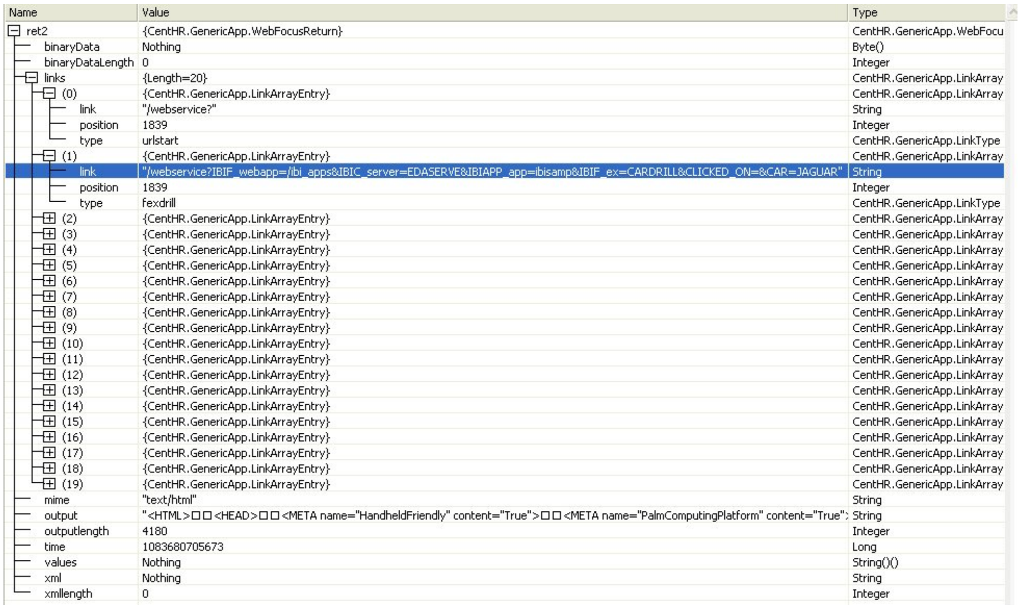
The following shows links contained in the WebQueryReturn structure for a graph in .svg format.



Name	Value	Type
ret2	{CentHR.GenericApp.WebFocusReturn}	CentHR.GenericApp.WebFocusReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
links	{Length=2}	CentHR.GenericApp.LinkArrayEntry()
(0)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"/webservice?PG_Func=GETBINARY&PG_File=qjswtcoa.svg"	String
position	18	Integer
type	svg	CentHR.GenericApp.LinkType
(1)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"/webservice?"	String
position	18	Integer
type	urlstart	CentHR.GenericApp.LinkType
mime	"text/html"	String
output	"□□□□□□<EMBED SRC="/webservice?PG_Func=GETBINARY&PG_File=qjswtcoa.svg" WIDTH=720 HEIGHT=450><br	String
outputlength	359	Integer
time	1089304619013	Long
values	Nothing	String()()
xml	Nothing	String
xmllength	0	Integer

Drill-Down

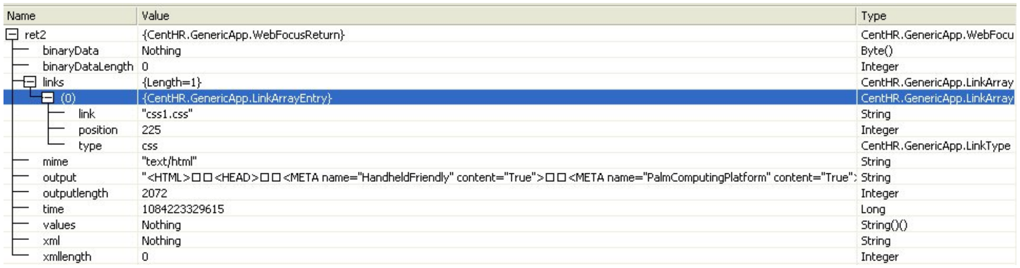
The following shows the links contained in the WebQueryReturn structure if a drill-down exists in a WebQuery report.



Name	Value	Type
ret2	{CentHR.GenericApp.WebFocusReturn}	CentHR.GenericApp.WebFocu
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
links	{Length=20}	CentHR.GenericApp.LinkArray
(0)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
link	"/webservice?"	String
position	1839	Integer
type	urlstart	CentHR.GenericApp.LinkType
(1)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
link	"/webservice?IBIF_webapp=/ibi_apps&IBIC_server=EDASERVE&IBIAPP_app=ibisamp&IBIF_ex=CARDRILL&CLICKED_ON=&CAR=JAGUAR"	String
position	1839	Integer
type	fexdrill	CentHR.GenericApp.LinkType
(2)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(3)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(4)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(5)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(6)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(7)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(8)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(9)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(10)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(11)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(12)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(13)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(14)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(15)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(16)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(17)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(18)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
(19)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
mime	"text/html"	String
output	"<HTML>□□<HEAD>□□<META name="HandheldFriendly" content="True">□□<META name="PalmComputingPlatform" content="True"	String
outputlength	4180	Integer
time	1083680705673	Long
values	Nothing	String()()
xml	Nothing	String
xmllength	0	Integer

Cascading Style Sheet

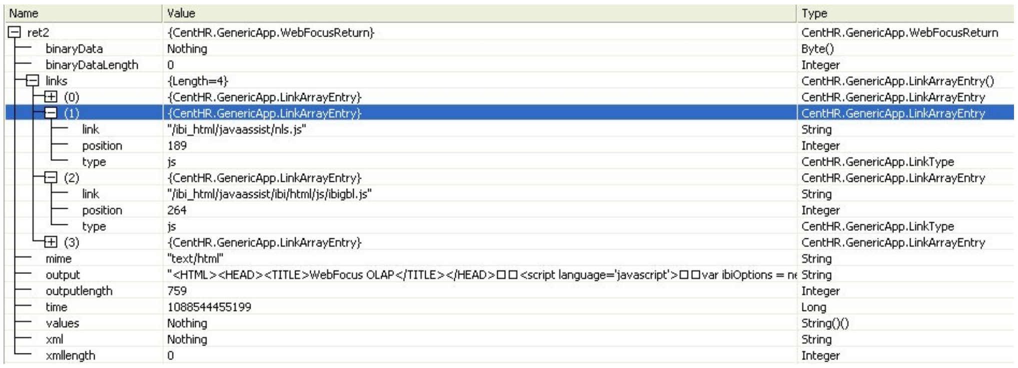
The following is an example of a link contained in the WebQueryReturn structure if a Cascading Style Sheet exists in a WebQuery report.



Name	Value	Type
ret2	{CentHR.GenericApp.WebFocusReturn}	CentHR.GenericApp.WebFocu
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
links	{Length=1}	CentHR.GenericApp.LinkArray
(0)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArray
link	"css1.css"	String
position	225	Integer
type	css	CentHR.GenericApp.LinkType
mime	"text/html"	String
output	"<HTML>□□<HEAD>□□<META name="HandheldFriendly" content="True">□□<META name="PalmComputingPlatform" content="True"	String
outputlength	2072	Integer
time	1084223329615	Long
values	Nothing	String()()
xml	Nothing	String
xmllength	0	Integer

JavaScript

The following is an example of the links contained in the WebQueryReturn structure if JavaScript exists within a Web Query report. The example is the result of running an OLAP-enabled report.



Name	Value	Type
ret2	{CentHR.GenericApp.WebFocusReturn}	CentHR.GenericApp.WebFocusReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
links	{Length=4}	CentHR.GenericApp.LinkArrayEntry()
(0)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(1)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"/ibi_html/javaassist/nls.js"	String
position	189	Integer
type	js	CentHR.GenericApp.LinkType
(2)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"/ibi_html/javaassist/ibi/html/js/ibigbl.js"	String
position	264	Integer
type	js	CentHR.GenericApp.LinkType
(3)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
mime	"text/html"	String
output	"<HTML><HEAD><TITLE>WebFocus OLAP</TITLE></HEAD>☐☐<script language='javascript'>☐☐var ibiOptions = ne	String
outputlength	759	Integer
time	1088544455199	Long
values	Nothing	String()()
xml	Nothing	String
xmllength	0	Integer

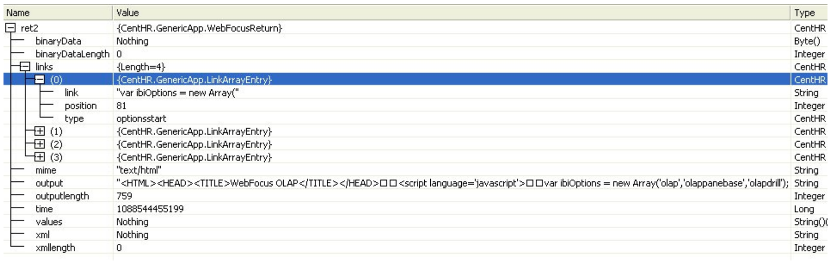
Optionsstart

The following is an example of a link contained in the WebQueryReturn structure if the HTML output from a Web Query report contains ibiOptions in its source.

For example:

```
var ibiOptions = new Array('olap','olappanebase','olapdrill');
```

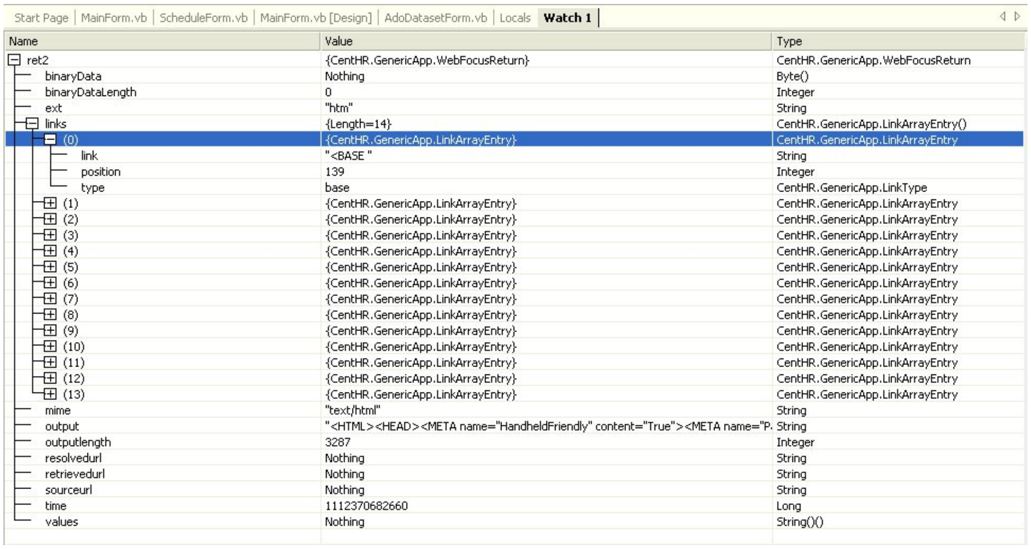
The example is the result of running an OLAP-enabled report.



Name	Value	Type
ret2	{CentHR.GenericApp.WebFocusReturn}	CentHR
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
links	{Length=4}	CentHR
(0)	{CentHR.GenericApp.LinkArrayEntry}	CentHR
link	"var ibiOptions = new Array("	String
position	81	Integer
type	optionsstart	CentHR
(1)	{CentHR.GenericApp.LinkArrayEntry}	CentHR
(2)	{CentHR.GenericApp.LinkArrayEntry}	CentHR
(3)	{CentHR.GenericApp.LinkArrayEntry}	CentHR
mime	"text/html"	String
output	"<HTML><HEAD><TITLE>WebFocus OLAP</TITLE></HEAD>☐☐<script language='javascript'>☐☐var ibiOptions = new Array('olap','olappanebase','olapdrill');	String
outputlength	759	Integer
time	1088544455199	Long
values	Nothing	String()(
xml	Nothing	String
xmllength	0	Integer

Base

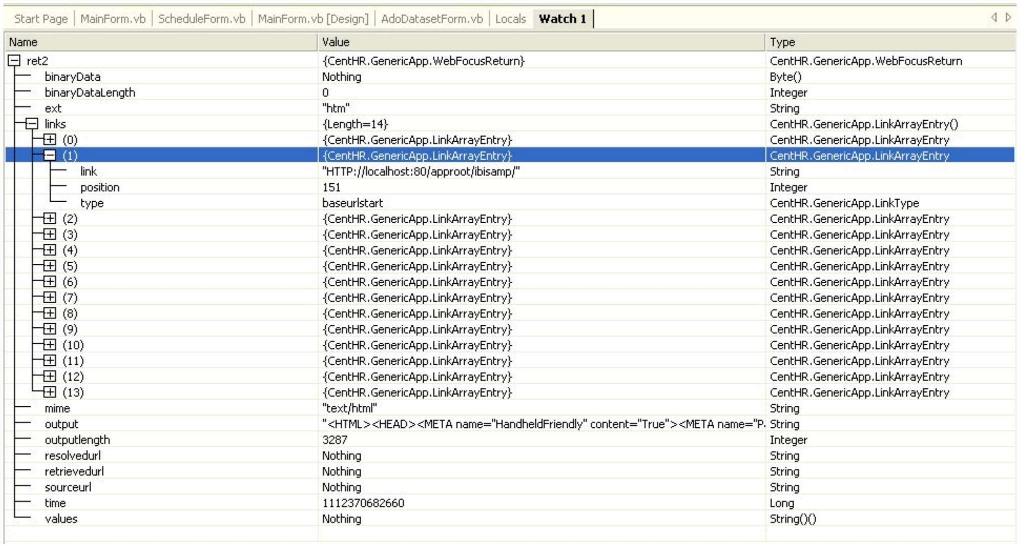
The following is an example of a link contained in the WebQueryReturn structure showing the starting position of <BASE within the HTML.



Name	Value	Type
ret2	{CentHR.GenericApp.WebFocusReturn}	CentHR.GenericApp.WebFocusReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
ext	"htm"	String
links	{Length=14}	CentHR.GenericApp.LinkArrayEntry()
(0)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"<BASE "	String
position	139	Integer
type	base	CentHR.GenericApp.LinkType
(1)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(2)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(3)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(4)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(5)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(6)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(7)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(8)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(9)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(10)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(11)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(12)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(13)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
mime	"text/html"	String
output	"<HTML><HEAD><META name="HandheldFriendly" content="True"><META name="P	String
outputlength	3287	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	Nothing	String
time	1112370682660	Long
values	Nothing	String()()

BaseUrlstart

The following is an example of a link contained in the WebQueryReturn structure showing the starting position of the Base URL within the HTML.



Name	Value	Type
ret2	{CentHR.GenericApp.WebFocusReturn}	CentHR.GenericApp.WebFocusReturn
binaryData	Nothing	Byte()
binaryDataLength	0	Integer
ext	"htm"	String
links	{Length=14}	CentHR.GenericApp.LinkArrayEntry()
(0)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(1)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
link	"HTTP://localhost:80/approot/ibisamp/"	String
position	151	Integer
type	baseurlstart	CentHR.GenericApp.LinkType
(2)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(3)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(4)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(5)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(6)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(7)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(8)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(9)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(10)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(11)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(12)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
(13)	{CentHR.GenericApp.LinkArrayEntry}	CentHR.GenericApp.LinkArrayEntry
mime	"text/html"	String
output	"<HTML><HEAD><META name="HandheldFriendly" content="True"><META name="P	String
outputlength	3287	Integer
resolvedurl	Nothing	String
retrievedurl	Nothing	String
sourceurl	Nothing	String
time	1112370682660	Long
values	Nothing	String()()

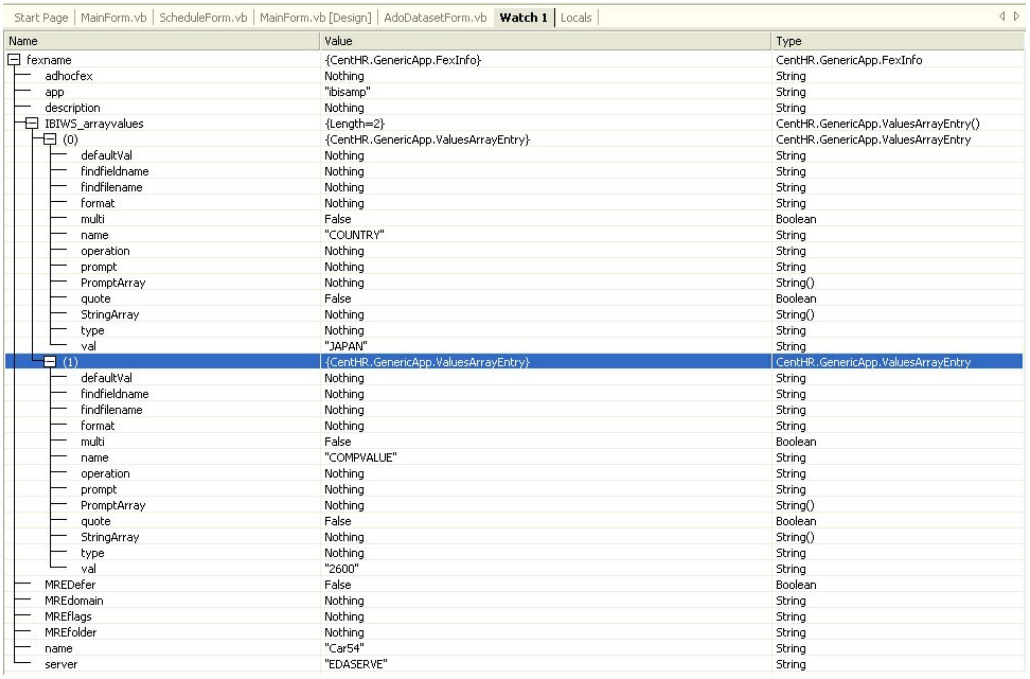
ValuesArrayEntry (Report Parameters Structure)

ValuesArrayEntry is a structure that contains information about the input parameters for a Web Query report.

Name	**Type**	**Description**
defaultVal	String	The Default value for the parameter as set by -DEFAULTS. This is retrieved from the WebQueryFexReflection function.
findfieldname	String	The field name used to retrieve a list of values for a Dynamic list.
findfilename	String	The file name used to retrieve a list of values for a Dynamic list.
format	String	The format of the parameter if one was set up in the Web Query report. For example, &COUNTRY.A24. This is retrieved from the WebQueryFexReflection function.

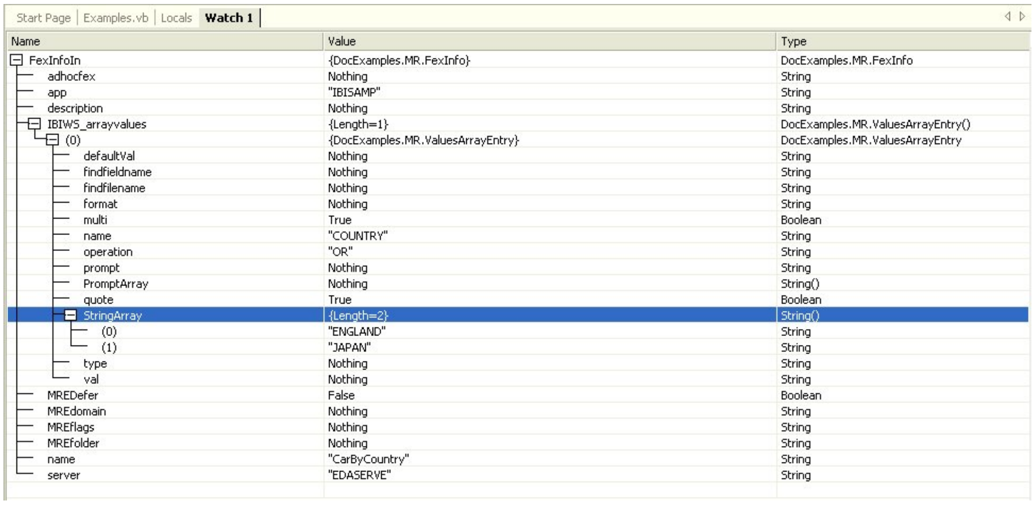
Name	Type	Description
multi	Boolean	True. Multi-select has been turned on for this parameter in the Web Query report. False. Single-select has been turned on for this parameter. This is the default.
operation	String	If multi-select has been turned on for this parameter in the Web Query report, this variable will either be set to AND or OR.
name	String	The name of the Web Query report parameter.
prompt	String	The prompt for the parameter if one was set up in the Web Query report. This is retrieved from the WebQueryFexReflection function.
PromptArray	String array	An array of descriptions for each valid value for the parameter if a pick list is set up in the Web Query report. This is retrieved from the WebQueryFexReflection function.
quote	Boolean	True. Each value should be surrounded by quotes. False. No quotes surround values. This is the default.
StringArray	String array	An array of valid values for the parameter if a pick list was set up in the Web Query report. This is retrieved from the WebQueryFexReflection function. It is a list of values sent as input to the parameter.
type	String	The default if -DEFAULTS is specified for this parameter in the Web Query report. Otherwise, it is unresolved.
val	String	The parameter value sent as input to the Web Query report. This is blank if StringArray is set.

The following is an example of information that has to be set in order to pass parameters to a Web Query report using the WebQueryRunFex function.



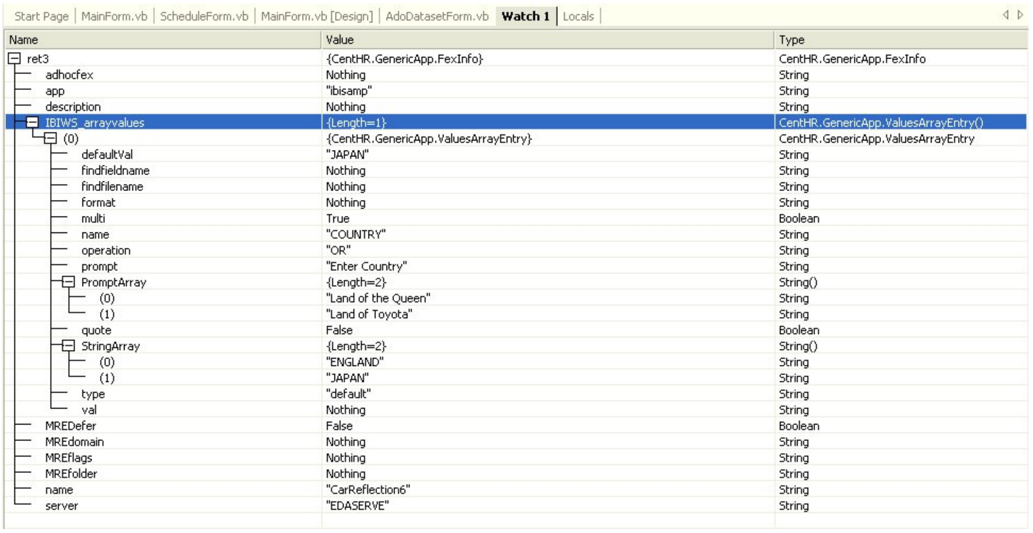
Name	Value	Type
fexname	{CentHR.GenericApp.FexInfo}	CentHR.GenericApp.FexInfo
adhocfex	Nothing	String
app	"ibisamp"	String
description	Nothing	String
IBIWS_arrayvalues	{Length=2}	CentHR.GenericApp.ValuesArrayEntry()
(0)	{CentHR.GenericApp.ValuesArrayEntry}	CentHR.GenericApp.ValuesArrayEntry
defaultVal	Nothing	String
findfieldname	Nothing	String
findfilename	Nothing	String
format	Nothing	String
multi	False	Boolean
name	"COUNTRY"	String
operation	Nothing	String
prompt	Nothing	String
PromptArray	Nothing	String()
quote	False	Boolean
StringArray	Nothing	String()
type	Nothing	String
val	"JAPAN"	String
(1)	{CentHR.GenericApp.ValuesArrayEntry}	CentHR.GenericApp.ValuesArrayEntry
defaultVal	Nothing	String
findfieldname	Nothing	String
findfilename	Nothing	String
format	Nothing	String
multi	False	Boolean
name	"COMPVALUE"	String
operation	Nothing	String
prompt	Nothing	String
PromptArray	Nothing	String()
quote	False	Boolean
StringArray	Nothing	String()
type	Nothing	String
val	"2600"	String
MREDefer	False	Boolean
MREdomain	Nothing	String
MREflags	Nothing	String
MREfolder	Nothing	String
name	"Car54"	String
server	"EDASERVE"	String

The following is an example of information that has to be set in order to pass multiple values to a parameter.



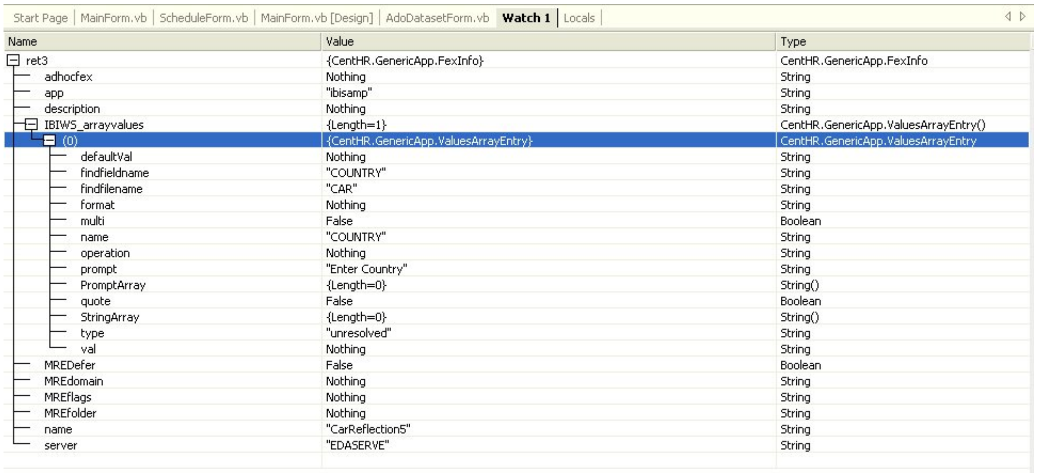
Name	Value	Type
FexInfoIn	{DocExamples.MR.FexInfo}	DocExamples.MR.FexInfo
adhocfex	Nothing	String
app	"IBISAMP"	String
description	Nothing	String
IBIWS_arrayvalues	{Length=1}	DocExamples.MR.ValuesArrayEntry()
(0)	{DocExamples.MR.ValuesArrayEntry}	DocExamples.MR.ValuesArrayEntry
defaultVal	Nothing	String
findfieldname	Nothing	String
findfilename	Nothing	String
format	Nothing	String
multi	True	Boolean
name	"COUNTRY"	String
operation	"OR"	String
prompt	Nothing	String
PromptArray	Nothing	String()
quote	True	Boolean
StringArray	{Length=2}	String()
(0)	"ENGLAND"	String
(1)	"JAPAN"	String
type	Nothing	String
val	Nothing	String
MREDefer	False	Boolean
MREdomain	Nothing	String
MREflags	Nothing	String
MREfolder	Nothing	String
name	"CarByCountry"	String
server	"EDASERVE"	String

The following is an example of the parameter information returned from the WebQueryFexReflection function if the Web Query report parameters contain pick lists.



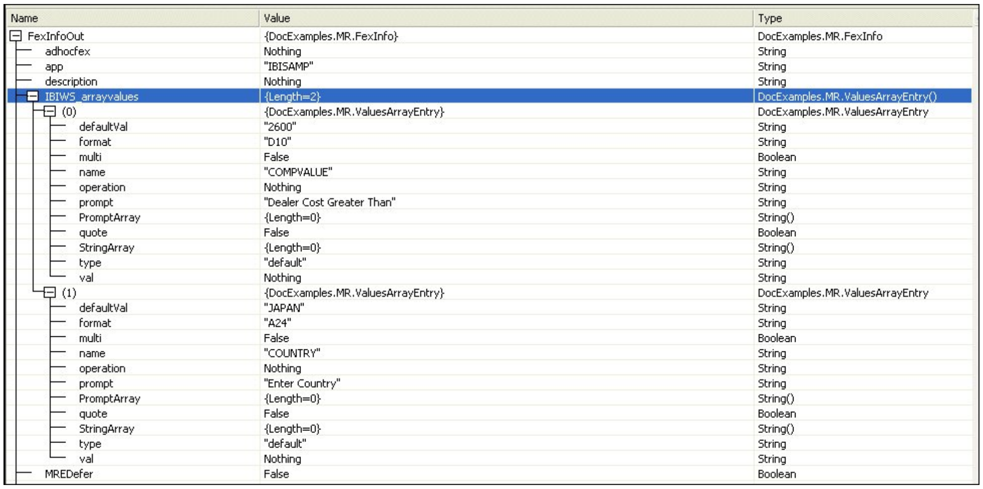
Name	Value	Type
ret3	{CentHR.GenericApp.FexInfo}	CentHR.GenericApp.FexInfo
adhocfex	Nothing	String
app	"ibisamp"	String
description	Nothing	String
IBIWS_arrayvalues	{Length=1}	CentHR.GenericApp.ValuesArrayEntry()
(0)	{CentHR.GenericApp.ValuesArrayEntry}	CentHR.GenericApp.ValuesArrayEntry
defaultVal	"JAPAN"	String
findfieldname	Nothing	String
findfilename	Nothing	String
format	Nothing	String
multi	True	Boolean
name	"COUNTRY"	String
operation	"OR"	String
prompt	"Enter Country"	String
PromptArray	{Length=2}	String()
(0)	"Land of the Queen"	String
(1)	"Land of Toyota"	String
quote	False	Boolean
StringArray	{Length=2}	String()
(0)	"ENGLAND"	String
(1)	"JAPAN"	String
type	"default"	String
val	Nothing	String
MREDefer	False	Boolean
MREdomain	Nothing	String
MREflags	Nothing	String
MREfolder	Nothing	String
name	"CarReflection6"	String
server	"EDASERVE"	String

The following is an example of the parameter information returned from WebQueryFexReflection function if the Web Query report parameters contain a Dynamic list.



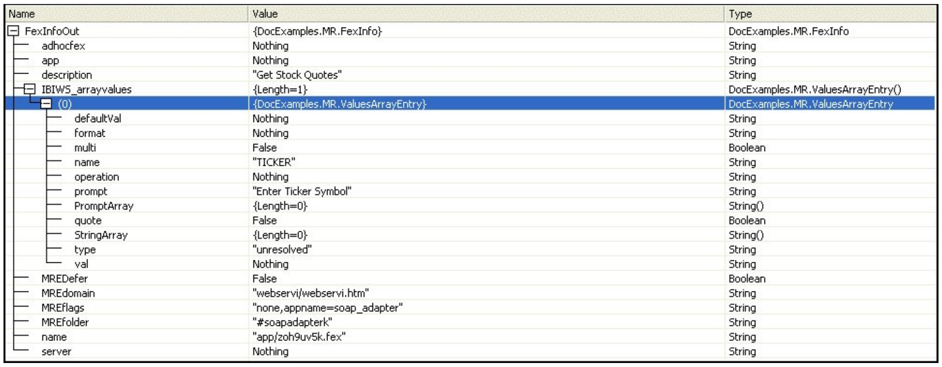
Name	Value	Type
ret3	{CentHR.GenericApp.FexInfo}	CentHR.GenericApp.FexInfo
adhocfex	Nothing	String
app	"ibisamp"	String
description	Nothing	String
IBIWS_arrayvalues	{Length=1}	CentHR.GenericApp.ValuesArrayEntry()
(0)	{CentHR.GenericApp.ValuesArrayEntry}	CentHR.GenericApp.ValuesArrayEntry
defaultVal	Nothing	String
findfieldname	"COUNTRY"	String
findfilename	"CAR"	String
format	Nothing	String
multi	False	Boolean
name	"COUNTRY"	String
operation	Nothing	String
prompt	"Enter Country"	String
PromptArray	{Length=0}	String()
quote	False	Boolean
StringArray	{Length=0}	String()
type	"unresolved"	String
val	Nothing	String
MREDefer	False	Boolean
MREdomain	Nothing	String
MREflags	Nothing	String
MREfolder	Nothing	String
name	"CarReflection5"	String
server	"EDASERVE"	String

The following is an example of the parameter information returned from the WebQueryFexReflection function if the Web Query report parameters contain formats.



Name	Value	Type
FexInfoOut	{DocExamples.MR.FexInfo}	DocExamples.MR.FexInfo
adhocfex	Nothing	String
app	"IBISAMP"	String
description	Nothing	String
IBIWS_arrayvalues	{Length=2}	DocExamples.MR.ValuesArrayEntry()
(0)	{DocExamples.MR.ValuesArrayEntry}	DocExamples.MR.ValuesArrayEntry
defaultVal	"2600"	String
format	"D10"	String
multi	False	Boolean
name	"COMPVALUE"	String
operation	Nothing	String
prompt	"Dealer Cost Greater Than"	String
PromptArray	{Length=0}	String()
quote	False	Boolean
StringArray	{Length=0}	String()
type	"default"	String
val	Nothing	String
(1)	{DocExamples.MR.ValuesArrayEntry}	DocExamples.MR.ValuesArrayEntry
defaultVal	"JAPAN"	String
format	"A24"	String
multi	False	Boolean
name	"COUNTRY"	String
operation	Nothing	String
prompt	"Enter Country"	String
PromptArray	{Length=0}	String()
quote	False	Boolean
StringArray	{Length=0}	String()
type	"default"	String
val	Nothing	String
MREDefer	False	Boolean

The following is an example of the parameter information in a Web Query report within Web Query. The WebQueryFexReflection function is used to retrieve these parameters.



Name	Value	Type
FexInfoOut	{DocExamples.MR.FexInfo}	DocExamples.MR.FexInfo
adhocfex	Nothing	String
app	Nothing	String
description	"Get Stock Quotes"	String
IBIWS_arrayvalues	{Length=1}	DocExamples.MR.ValuesArrayEntry()
(0)	{DocExamples.MR.ValuesArrayEntry}	DocExamples.MR.ValuesArrayEntry
defaultVal	Nothing	String
format	Nothing	String
multi	False	Boolean
name	"TICKER"	String
operation	Nothing	String
prompt	"Enter Ticker Symbol"	String
PromptArray	{Length=0}	String()
quote	False	Boolean
StringArray	{Length=0}	String()
type	"unresolved"	String
val	Nothing	String
MREDefer	False	Boolean
MREdomain	"webservi/webservi.htm"	String
MREflags	"none,appname=soap_adapter"	String
MREfolder	"#soapadapterk"	String
name	"app/zoh9uv5k.fex"	String
server	Nothing	String

Report Broker Schedule Structure

In this section:

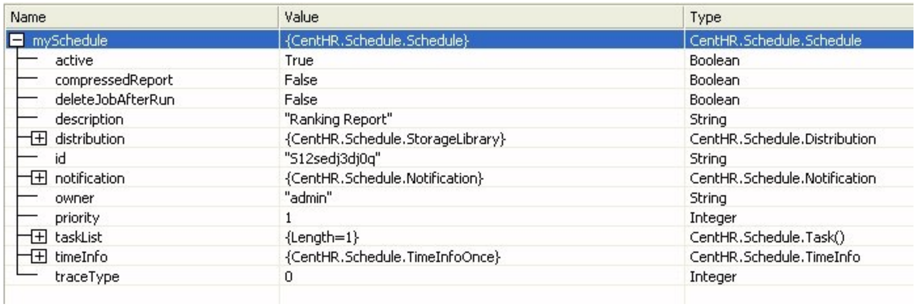
The schedule structure encapsulates the process of scheduling reports for distribution by Report Broker. Schedule information is represented by a unique identifier called a scheduleId and is comprised of various information, such as Destination, Distribution, and TimeInfo, representing components of the schedule.

Name	Type	Description
Active	Boolean	The flag indicating whether or not a scheduled distribution is active. If True, the schedule is active. If False, the schedule is inactive.

Name	Type	Description
DeleteJobAfterRun	Boolean	A flag indicating whether or not a schedule is deleted after running the job. If True, the job is deleted.
Description	String	The text describing the job that is being scheduled. This text needs to be unique for each owner of a scheduled job. The maximum size of the description is 90 characters.
Distribution	DistributionEmail DistributionPrint StorageMRE	Assigns the distribution information associated with this schedule.
Id	String	A unique identifier to this schedule. The developer/user should never manipulate this identifier. If it is set by the user during schedule creation, Report Broker will override that setting.
Notification	Notification	Specifies the notification information associated with this schedule.
Owner	String	The owner of this schedule. The maximum size of the owner is 48 characters.
Priority	Integer	The priority level for the scheduled job. The value ranges from 1 (highest priority) to 5 (lowest priority).
TaskList	Task TaskStandardReport	A list of task information, with each task information containing one report that is to be distributed as part of this schedule. The order of the tasks in the list corresponds to the order the tasks will be synchronously run and then distributed by the Distributed Server.

Name	Type	Description
TimeInfo	TimeInfoDay TimeInfoHour TimeInfoMinute TimeInfoMonth TimeInfoOnce TimeInfoWeek TimeInfoYear TimeInterval	The information associated with this schedule that contains the frequency and time parameters indicating when and how often this schedule should run.
TraceType	Integer	0 = Default Trace. Uses Report Broker trace configuration setting. 1 = No Traces 2 = Trace Schedule 3 = Trace Schedule and Report

The following is an example of the Schedule structure.



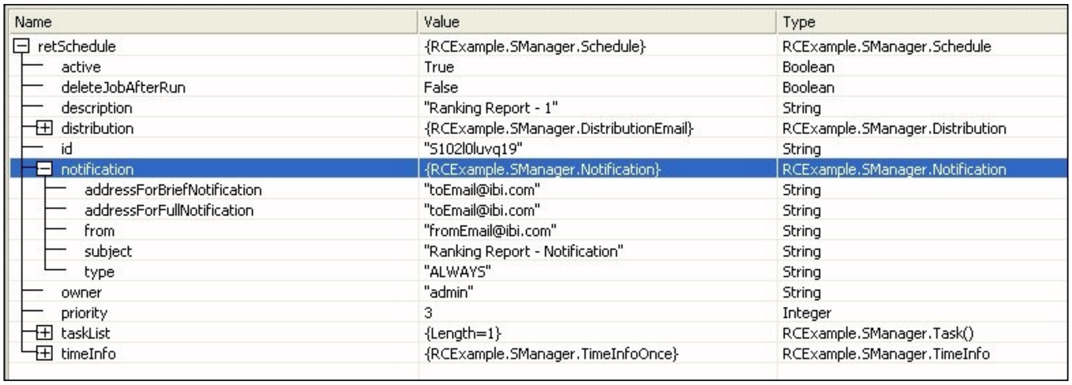
Name	Value	Type
mySchedule	{CentHR.Schedule.Schedule}	CentHR.Schedule.Schedule
active	True	Boolean
compressedReport	False	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report"	String
distribution	{CentHR.Schedule.StorageLibrary}	CentHR.Schedule.Distribution
id	"S12sedj3dj0q"	String
notification	{CentHR.Schedule.Notification}	CentHR.Schedule.Notification
owner	"admin"	String
priority	1	Integer
taskList	{Length=1}	CentHR.Schedule.Task()
timeInfo	{CentHR.Schedule.TimeInfoOnce}	CentHR.Schedule.TimeInfo
traceType	0	Integer

Notification (Notification Structure)

When scheduled reports are distributed, Report Broker allows selected individuals to be notified with log information about the distribution. This notification feature can be altered on a per schedule basis and can be set to inactive, always notify, or notify only on error. Each schedule allows two types of notification to be sent simultaneously, brief and full. Where brief notification contains partial log information and full notification contains complete log information.

Name	Type	Description
AddressForBriefNotification	String	The e-mail address where a brief notification message will be sent after running a schedule in Report Broker. The content of the brief notification e-mail is partial log information for a given schedule run. The maximum size of the brief notification e-mail address is 75 characters.
AddressForFullNotification	String	The e-mail address where a full notification message will be sent upon running a schedule in Report Broker. The content of the full notification e-mail is the complete log information for a given schedule run. The maximum size of the full notification e-mail address is 75 characters.
From	String	The e-mail address linked to the From header to which notification will be sent upon running a schedule in Report Broker. The maximum size of the From address is 75 characters.
Subject	String	The subject header in the e-mail to which notification will be sent upon running a schedule in Report Broker. The maximum size of the e-mail Subject is 255 characters.
Type	String	The type of notification message to be sent upon the running of a Report Broker schedule. The three possible types are: ALWAYS, INACTIVE, and ONERROR.

The following is an example of the Notification structure.



Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.DistributionEmail}	RCExample.SManager.Distribution
id	"S102l0luvq19"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
addressForBriefNotification	"toEmail@ibi.com"	String
addressForFullNotification	"toEmail@ibi.com"	String
from	"fromEmail@ibi.com"	String
subject	"Ranking Report - Notification"	String
type	"ALWAYS"	String
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
timeInfo	{RCExample.SManager.TimeInfoOnce}	RCExample.SManager.TimeInfo

Task (Task Structure)

The Task structure is used to define Report Broker task information. Since a Report Broker schedule can contain many tasks of different task types, the task structure is used to define the array of tasks. For example, the TaskStandardReport subscript within the array is of a structure dependent on its task type. If a schedule is made up of tasks with all of them being of the same task type, TaskStandardReport can be used to define the array of tasks instead of the task structure.

TaskStandardReport (Standard Report Structure)

TaskStandardReport consists of its own structure that defines the information needed for the schedule. The TaskStandardReport structure is supported by Report Broker (Web Query Server Procedure, MyReport, File, and URL are the others). Each task type corresponds to a means of running or accessing a report that can be distributed by Report Broker. TaskStandardReport allows Report Broker to schedule the distribution of Managed Reporting Standard Reports. With TaskStandardReport, you can associate an alert with the report, which allows you to schedule actions that are contingent upon specific alert conditions being triggered.

Name	**Type**	**Description**
Description	String	The text used to describe the task. The maximum size for the description is 255 characters.

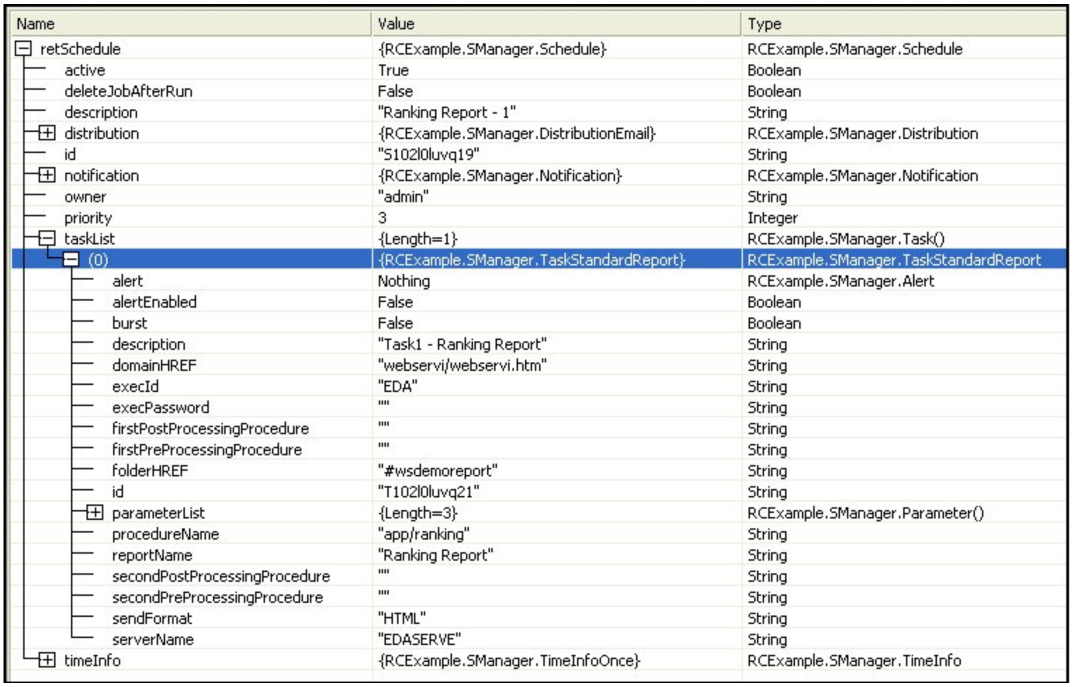
Name	Type	Description
Id	String	The unique identifier Report Broker generates for every saved task. The user is not allowed to manipulate this ID. If it is assigned a value different from the generated value, Report Broker will override that with the generated ID.
ReportName	String	The name of the report associated with this task. It is used when sending output as an attachment. The maximum size for report name is 64 characters.
TaskRetry	TimeInterval	A structure defining the time interval for restarting a task if the task did not complete successfully.
Alert	Alert	An Alert structure.
AlertEnabled	Boolean	The value that determines whether or not an alert is enabled. If True, the alert is enabled. If False, it is disabled.
DomainHREF	String	The internal value for a Reporting folder assigned by Managed Reporting at the time the folder is created.
FolderHREF	String	The internal value for a Reporting folder assigned by Managed Reporting at the time the folder is created.
ProcedureDescription	String	The description of the ProcedureName. The value is not stored in the Report Broker repository, but can be populated using Web Query Web Services.

Name	Type	Description
Burst	Boolean	The value that specifies whether or not report bursting is enabled. Report bursting allows you to segment a report into sections based upon a primary sort field. The report segments are then distributed as separate reports by the Distribution Server. Access to these report segments is based upon burst values (specific values of the primary sort field) that are associated with e-mail addresses in distribution lists or user IDs in library access lists.
ExecID	String	The user name needed to establish a connection to the Web Query Reporting Server. The user name is one of the credentials necessary for a user to access a Web Query procedure that resides on the Web Query Reporting Server during scheduling, as well as to run this procedure at the time the job is run.
ExecPassword	String	The password needed to establish a connection to the Web Query Reporting Server. The password is one of the credentials necessary for a user to access a Web Query procedure that resides on the Web Query Reporting Server during scheduling, as well as run this procedure at the time the job is run.

Name	**Type**	**Description**
FirstPostProcessingProcedure	String	The name of the first of two possible post-processing procedures that can be associated with this Web Query Server procedure task. Post-processing procedures (available for Web Query Server procedure, Standard Report, and My Report tasks only) are non-reporting Web Query procedures that run synchronously after the execution of their associated task. They are often used to reset computing or data environments.
FirstPreprocessingProcedure	String	The name of the first two possible pre-processing procedures that can be associated with this Web Query Server procedure task. Pre-processing procedures (available for Web Query Server procedure, Standard Report, and MyReport tasks only) are non-reporting Web Query procedures that run synchronously prior the execution of their associated task. They are often used to set or test conditions before the running of reports.
ParameterList	Parameter	An array of parameter information, with each structure representing a name value pair that is matched and then resolved upon the execution of this Web Query Server procedure.
ProcedureName	String	The internal value for a Web Query procedure assigned by Managed Reporting at the time the procedure is created (for example, app/ranking.)

Name	Type	Description
SecondPostProcessingProcedure	String	The name of the second of two possible post-processing procedures that can be associated with this Web Query Server Procedure task. Post-processing procedures (available for Web Query Server procedure, Standard Report, and MyReport Tasks only) are non-reporting Web Query procedures that run synchronously after the execution of their associated task. They are often used to reset computing or data environments.
SecondPreProcessingProcedure	String	The name of the second of two possible pre-processing procedures that can be associated with this Web Query Server procedure task. Pre-processing procedures (available for Web Query Server Procedure, Standard Report, and My Report Tasks only) are non-reporting Web Query procedures that run synchronously prior the execution of their associated task. They are often used to set or test conditions before the running of reports.
SendFormat	String	The report format that will be generated by the Web Query Reporting Server. For example, PDF or HTML.
ServerName	String	The name of the Web Query Reporting Server housing the Web Query procedure associated with this task.

The following is an example of the TaskStandardReport structure.



Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.DistributionEmail}	RCExample.SManager.Distribution
id	"S102l0luvq19"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
(0)	{RCExample.SManager.TaskStandardReport}	RCExample.SManager.TaskStandardReport
alert	Nothing	RCExample.SManager.Alert
alertEnabled	False	Boolean
burst	False	Boolean
description	"Task1 - Ranking Report"	String
domainHREF	"webservi/webservi.htm"	String
execId	"EDA"	String
execPassword	""	String
firstPostProcessingProcedure	""	String
firstPreProcessingProcedure	""	String
folderHREF	"#wsdemoreport"	String
id	"T102l0luvq21"	String
parameterList	{Length=3}	RCExample.SManager.Parameter()
procedureName	"app/ranking"	String
reportName	"Ranking Report"	String
secondPostProcessingProcedure	""	String
secondPreProcessingProcedure	""	String
sendFormat	"HTML"	String
serverName	"EDASERVE"	String
timeInfo	{RCExample.SManager.TimeInfoOnce}	RCExample.SManager.TimeInfo

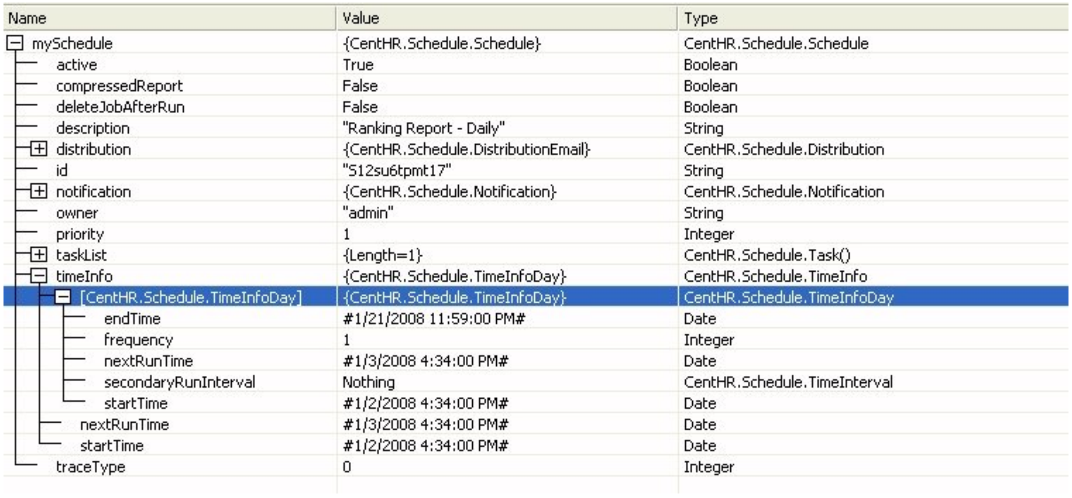
TimeInfoDay (Daily Scheduling Structure)

The TimeInfoDay structure represents time information pertaining to scheduled reports that run in intervals of days.

Name	Type	Description
NextRunTime	Date	Calendar information indicating the next run time for a scheduled event. NextRunTime is generally used only as an internal function. It is not recommended for general usage and is not formally supported. Manually re-setting system designated fields might result in unpredictable behavior. However, the expected behavior if a user or programmer uses NextRunTime to designate a next run time that is different from the run time calculated internally is that the scheduled job will run at the new next run time. It then resumes normal calculation.
StartTime	Date	Calendar information indicating the start time for a scheduled event. The start time is designated as the first time a new schedule is set to run. Creating a new schedule and altering any jobs that are to run in the future will create an entirely new start time. The default start time is the current time. After the first time a schedule runs, subsequent run times can be obtained via NextRunTime.
EndTime	Date	Calendar information indicating the end time for a scheduled event. Report Broker will not run a job for this schedule after this date.
Frequency	Integer	The frequency for a scheduled event in days. For example, if an e-mail report distribution is set to run every third day, the frequency would be 3.

Name	Type	Description
SecondaryRunInterval	TimeInterval	A structure that defines a secondary run interval within the day the schedule runs. The interval is setup to run on a minute(s) basis for a specified number of minutes or until a specified time.

The following is an example of the TimeInfoDay structure.



Name	Value	Type
mySchedule	{CentHR.Schedule.Schedule}	CentHR.Schedule.Schedule
active	True	Boolean
compressedReport	False	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - Daily"	String
distribution	{CentHR.Schedule.DistributionEmail}	CentHR.Schedule.Distribution
id	"S12su6tpmt17"	String
notification	{CentHR.Schedule.Notification}	CentHR.Schedule.Notification
owner	"admin"	String
priority	1	Integer
taskList	{Length=1}	CentHR.Schedule.Task()
timeInfo	{CentHR.Schedule.TimeInfoDay}	CentHR.Schedule.TimeInfo
[CentHR.Schedule.TimeInfoDay]	{CentHR.Schedule.TimeInfoDay}	CentHR.Schedule.TimeInfoDay
endTime	#1/21/2008 11:59:00 PM#	Date
frequency	1	Integer
nextRunTime	#1/3/2008 4:34:00 PM#	Date
secondaryRunInterval	Nothing	CentHR.Schedule.TimeInterval
startTime	#1/2/2008 4:34:00 PM#	Date
nextRunTime	#1/3/2008 4:34:00 PM#	Date
startTime	#1/2/2008 4:34:00 PM#	Date
traceType	0	Integer

TimeInfoHour (Hourly Scheduling Structure)

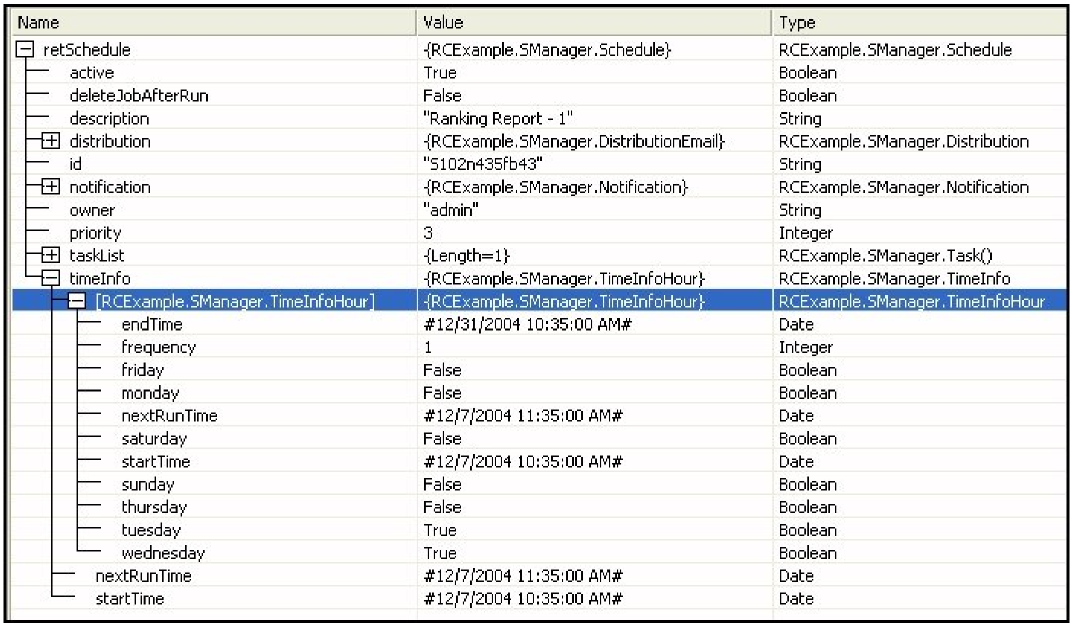
The TimeInfoHour structure represents time information pertaining to scheduled reports that run in intervals of hours.

Name	Type	Description
EndTime	Date	Calendar information indicating the end time for a scheduled event. Report Broker will not run a job for this schedule after this date.
Frequency	Integer	The frequency for a scheduled event, in hours. For example, if an e-mail report distribution is set to run every five hours, the frequency would be 5.

Name	Type	Description
Friday	Boolean	Whether or not the Report Broker job is scheduled for a Friday. If true, the job will run on a Friday.
Monday	Boolean	Whether or not the Report Broker job is scheduled for a Monday. If true, the job will run on a Monday.
Saturday	Boolean	Whether or not the Report Broker job is scheduled for a Saturday. If true, the job will run on a Saturday.
Sunday	Boolean	Whether or not the Report Broker job is scheduled for a Sunday. If true, the job will run on a Sunday.
Thursday	Boolean	Whether or not the Report Broker job is scheduled for a Thursday. If true, the job will run on a Thursday.
Tuesday	Boolean	Whether or not the Report Broker job is scheduled for a Tuesday. If true, the job will run on a Tuesday.
Wednesday	Boolean	Whether or not the Report Broker job is scheduled for a Wednesday. If true, the job will run on a Wednesday.

Name	Type	Description
NextRunTime	Date	Calendar information indicating the next run time for a scheduled event. NextRunTime is generally used only as an internal function. It is not recommended for general usage and is not formally supported. Manually re-setting system designated fields might result in unpredictable behavior. However, the expected behavior if a user or programmer uses NextRunTime to designate a next run time that is different from the run time calculated internally is that the scheduled job will run at the new next run time. It then resumes normal calculation.
StartTime	Date	Calendar information indicating the start time for a scheduled event. The start time is designated as the first time a new schedule is set to run. Creating a new schedule and altering any jobs that are to run in the future will create an entirely new start time. The default start time is the current time. After the first time a schedule runs, subsequent run times can be obtained via NextRunTime.

The following is an example of the TimeInfoHour structure.



Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.DistributionEmail}	RCExample.SManager.Distribution
id	"S102n435fb43"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
timeInfo	{RCExample.SManager.TimeInfoHour}	RCExample.SManager.TimeInfo
[RCExample.SManager.TimeInfoHour]	{RCExample.SManager.TimeInfoHour}	RCExample.SManager.TimeInfoHour
endTime	#12/31/2004 10:35:00 AM#	Date
frequency	1	Integer
friday	False	Boolean
monday	False	Boolean
nextRunTime	#12/7/2004 11:35:00 AM#	Date
saturday	False	Boolean
startTime	#12/7/2004 10:35:00 AM#	Date
sunday	False	Boolean
thursday	False	Boolean
tuesday	True	Boolean
wednesday	True	Boolean
nextRunTime	#12/7/2004 11:35:00 AM#	Date
startTime	#12/7/2004 10:35:00 AM#	Date

TimeInfoMinute (Minute Scheduling Structure)

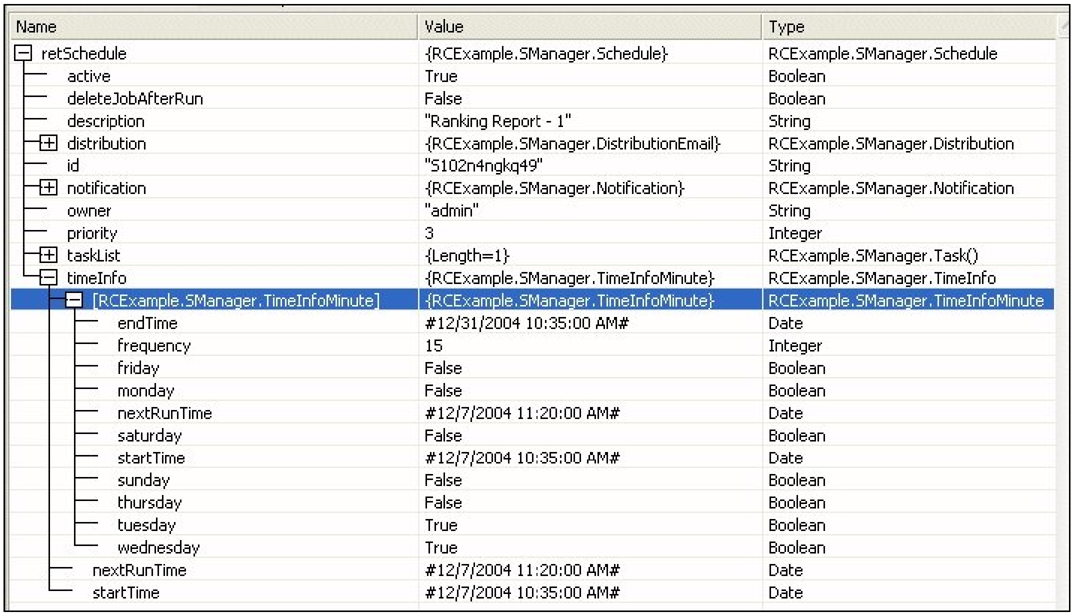
The TimeInfoMinute structure represents time information pertaining to scheduled reports that run in intervals of minutes.

Name	Type	Description
EndTime	Date	Calendar information indicating the end time for a scheduled event. Report Broker will not run a job for this schedule after this date.
Frequency	Integer	The frequency for a scheduled event, in minutes. For example, if an e-mail report distribution is set to run every five minutes, the frequency would be 5.
Friday	Boolean	Whether or not the Report Broker job is scheduled for a Friday. If true, the job will run on a Friday.

Name	Type	Description
Monday	Boolean	Whether or not the Report Broker job is scheduled for a Monday. If true, the job will run on a Monday.
Saturday	Boolean	Whether or not the Report Broker job is scheduled for a Saturday. If true, the job will run on a Saturday.
Sunday	Boolean	Whether or not the Report Broker job is scheduled for a Sunday. If true, the job will run on a Sunday.
Thursday	Boolean	Whether or not the Report Broker job is scheduled for a Thursday. If true, the job will run on a Thursday.
Tuesday	Boolean	Whether or not the Report Broker job is scheduled for a Tuesday. If true, the job will run on a Tuesday.
Wednesday	Boolean	Whether or not the Report Broker job is scheduled for a Wednesday. If true, the job will run on a Wednesday.
NextRunTime	Date	Calendar information indicating the next run time for a scheduled event. NextRunTime is generally used only as an internal function. It is not recommended for general usage and is not formally supported. Manually re-setting system designated fields might result in unpredictable behavior. However, the expected behavior if a user or programmer uses NextRunTime to designate a next run time that is different from the run time calculated internally is that the scheduled job will run at the new next run time. It then resumes normal calculation.

Name	Type	Description
StartTime	Date	Calendar information indicating the start time for a scheduled event. The start time is designated as the first time a new schedule is set to run. Creating a new schedule and altering any jobs that are to run in the future will create an entirely new start time. The default start time is the current time. After the first time a schedule runs, subsequent run times can be obtained via NextRunTime.

The following is an example of the TimeInfoMinute structure.



Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.DistributionEmail}	RCExample.SManager.Distribution
id	"S102n4ngkq49"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
timeInfo	{RCExample.SManager.TimeInfoMinute}	RCExample.SManager.TimeInfo
[RCExample.SManager.TimeInfoMinute]	{RCExample.SManager.TimeInfoMinute}	RCExample.SManager.TimeInfoMinute
endTime	#12/31/2004 10:35:00 AM#	Date
frequency	15	Integer
friday	False	Boolean
monday	False	Boolean
nextRunTime	#12/7/2004 11:20:00 AM#	Date
saturday	False	Boolean
startTime	#12/7/2004 10:35:00 AM#	Date
sunday	False	Boolean
thursday	False	Boolean
tuesday	True	Boolean
wednesday	True	Boolean
nextRunTime	#12/7/2004 11:20:00 AM#	Date
startTime	#12/7/2004 10:35:00 AM#	Date

TimeInfoMonth (Monthly Scheduling Structure)

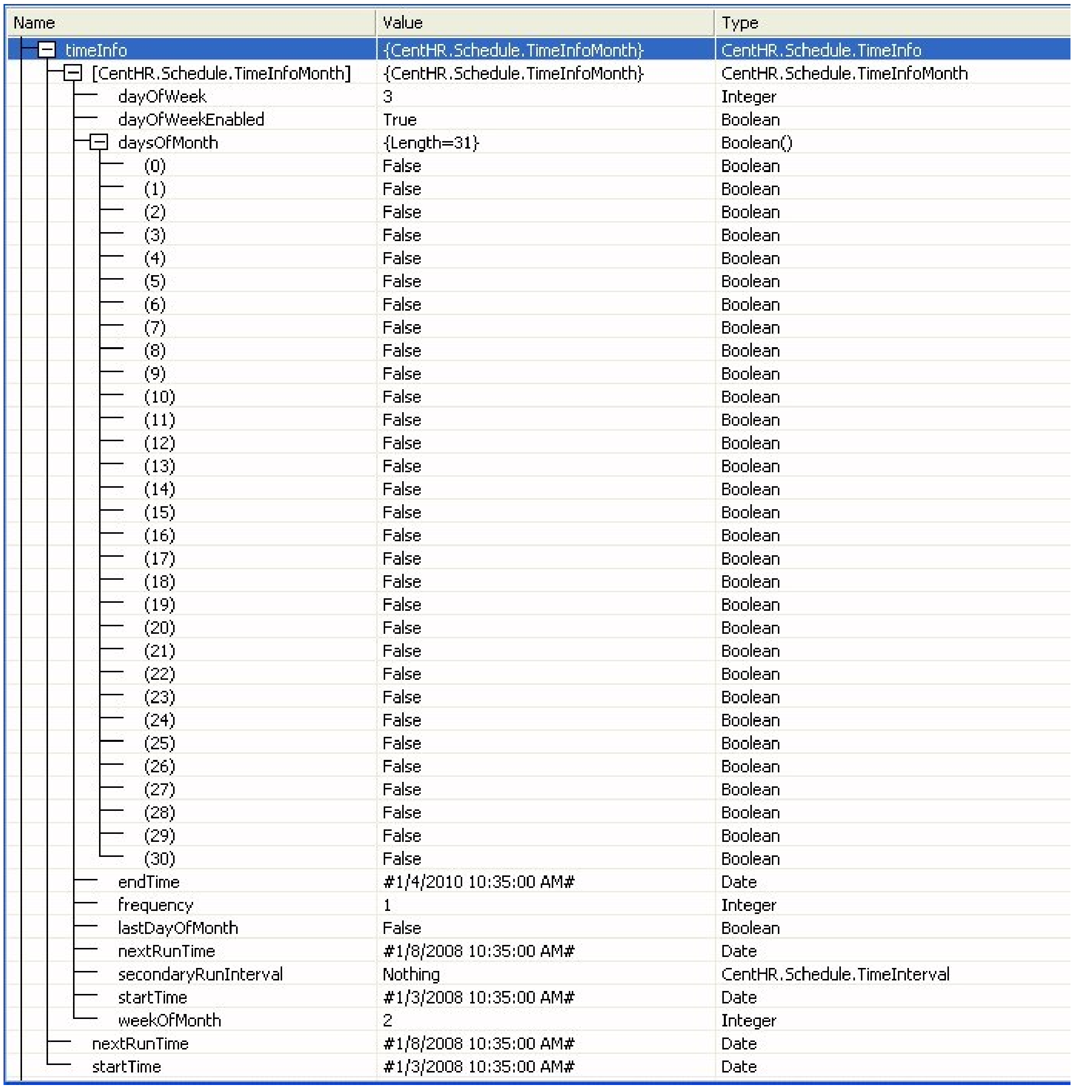
The TimeInfoMonth structure represents time information pertaining to scheduled reports that run in intervals of months.

Name	Type	Description
DayOfWeek	Integer	Day of the week for the report to run. DayOfWeekEnabled must be true. Valid Values are: ❏ 1- Sunday ❏ 2 - Monday ❏ 3 - Tuesday ❏ 4 - Wednesday ❏ 5 - Thursday ❏ 6 - Friday ❏ 7 - Saturday
DayOfWeekEnabled	Boolean	true - DayOfWeek and WeekOfMonth must be set. false - DaysOfMonth and/or LastDayOfMonth must be set.
DaysOfMonth	Boolean	A 31 element array indicating which days of the month have been selected for a report to run. All array members are initialized to False. Those members of the array that are then set to true are the days of the month the schedule will run. DayOfWeekEnabled must be false.
EndTime	Date	Calendar information indicating the end time for a scheduled event. Report Broker will not run a job for this schedule after this date.

Name	Type	Description
Frequency	Integer	The frequency for a scheduled distribution. For example, if an e-mail report distribution is set to run every six months, the frequency is 6. Report Broker uses standard Java data arithmetic to determine run dates when adding months to a start date.
LastDayOfMonth	Boolean	An indicator whether or not the last day of the month flag is set. When this flag is set, Report Broker runs a schedule on the last day of the month regardless of what day it is. For example, a schedule set to run on February 28th will next run on March 31st if this flag is set. Otherwise, Report Broker will run the job on the corresponding day of the next month. If this flag isn't set then it will run the report on March 28th. If the corresponding day of the next month doesn't exist, then Report Broker will not run the report.
SecondaryRunInterval	TimeInterval	A structure that defines a secondary run interval within the day the schedule runs. The interval is setup to run on a minute(s) basis for a specified number of minutes or until a specified time.
WeekOfMonth	Integer	Week of the month for the report to run. DayOfWeekEnabled must be true. Valid Values are: ❏ 1 - first week ❏ 2 - second week ❏ 3 - third week ❏ 4 - fourth week ❏ 5 - last week

Name	Type	Description
NextRunTime	Date	Calendar information indicating the next run time for a scheduled event. NextRunTime is generally used only as an internal function. It is not recommended for general usage and is not formally supported. Manually re-setting system designated fields might result in unpredictable behavior. However, the expected behavior if a user or programmer uses NextRunTime to designate a next run time that is different from the run time calculated internally is that the scheduled job will run at the new next run time. It then resumes normal calculation.
StartTime	Date	Calendar information indicating the start time for a scheduled event. The start time is designated as the first time a new schedule is set to run. Creating a new schedule and altering any jobs that are to run in the future will create an entirely new start time. The default start time is the current time. After the first time a schedule runs, subsequent run times can be obtained via NextRunTime.

The following is an example of the TimeInfoMonth structure.



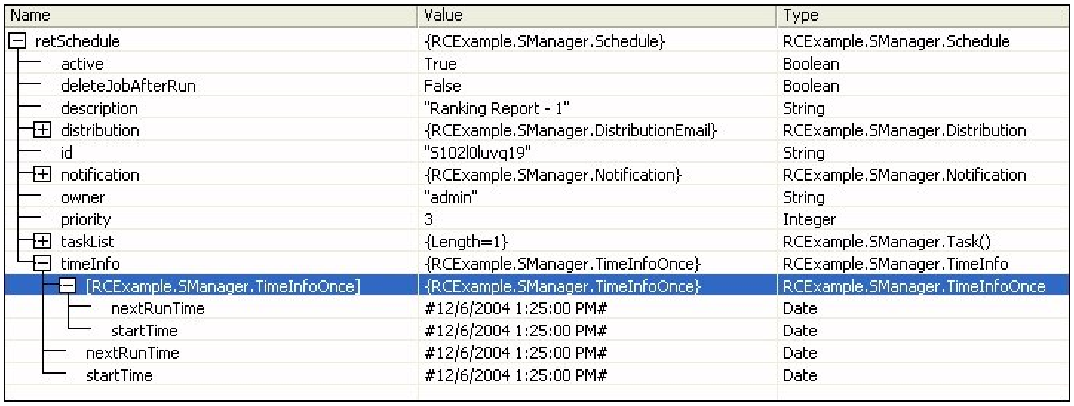
Name	Value	Type
timeInfo	{CentHR.Schedule.TimeInfoMonth}	CentHR.Schedule.TimeInfo
[CentHR.Schedule.TimeInfoMonth]	{CentHR.Schedule.TimeInfoMonth}	CentHR.Schedule.TimeInfoMonth
dayOfWeek	3	Integer
dayOfWeekEnabled	True	Boolean
daysOfMonth	{Length=31}	Boolean()
(0)	False	Boolean
(1)	False	Boolean
(2)	False	Boolean
(3)	False	Boolean
(4)	False	Boolean
(5)	False	Boolean
(6)	False	Boolean
(7)	False	Boolean
(8)	False	Boolean
(9)	False	Boolean
(10)	False	Boolean
(11)	False	Boolean
(12)	False	Boolean
(13)	False	Boolean
(14)	False	Boolean
(15)	False	Boolean
(16)	False	Boolean
(17)	False	Boolean
(18)	False	Boolean
(19)	False	Boolean
(20)	False	Boolean
(21)	False	Boolean
(22)	False	Boolean
(23)	False	Boolean
(24)	False	Boolean
(25)	False	Boolean
(26)	False	Boolean
(27)	False	Boolean
(28)	False	Boolean
(29)	False	Boolean
(30)	False	Boolean
endTime	#1/4/2010 10:35:00 AM#	Date
frequency	1	Integer
lastDayOfMonth	False	Boolean
nextRunTime	#1/8/2008 10:35:00 AM#	Date
secondaryRunInterval	Nothing	CentHR.Schedule.TimeInterval
startTime	#1/3/2008 10:35:00 AM#	Date
weekOfMonth	2	Integer
nextRunTime	#1/8/2008 10:35:00 AM#	Date
startTime	#1/3/2008 10:35:00 AM#	Date

TimeInfoOnce (Schedule Once Structure)

The TimeInfoOnce structure represents time information pertaining to scheduled jobs that are to run only once.

Name	**Type**	**Description**
NextRunTime	Date	Calendar information indicating the next run time for a scheduled event. NextRunTime is generally used only as an internal function. It is not recommended for general usage and is not formally supported. Manually re-setting system designated fields might result in unpredictable behavior. However, the expected behavior if a user or programmer uses NextRunTime to designate a next run time that is different from the run time calculated internally is that the scheduled job will run at the new next run time. It then resumes normal calculation.
StartTime	Date	Calendar information indicating the start time for a scheduled event. The start time is designated as the first time a new schedule is set to run. Creating a new schedule and altering any jobs that are to run in the future will create an entirely new start time. The default start time is the current time. After the first time a schedule runs, subsequent run times can be obtained via NextRunTime.

The following is an example of the TimeInfoOnce structure.



Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.DistributionEmail}	RCExample.SManager.Distribution
id	"S102l0luvq19"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
timeInfo	{RCExample.SManager.TimeInfoOnce}	RCExample.SManager.TimeInfo
[RCExample.SManager.TimeInfoOnce]	{RCExample.SManager.TimeInfoOnce}	RCExample.SManager.TimeInfoOnce
nextRunTime	#12/6/2004 1:25:00 PM#	Date
startTime	#12/6/2004 1:25:00 PM#	Date
nextRunTime	#12/6/2004 1:25:00 PM#	Date
startTime	#12/6/2004 1:25:00 PM#	Date

TimeInfoWeek (Weekly Schedule Structure)

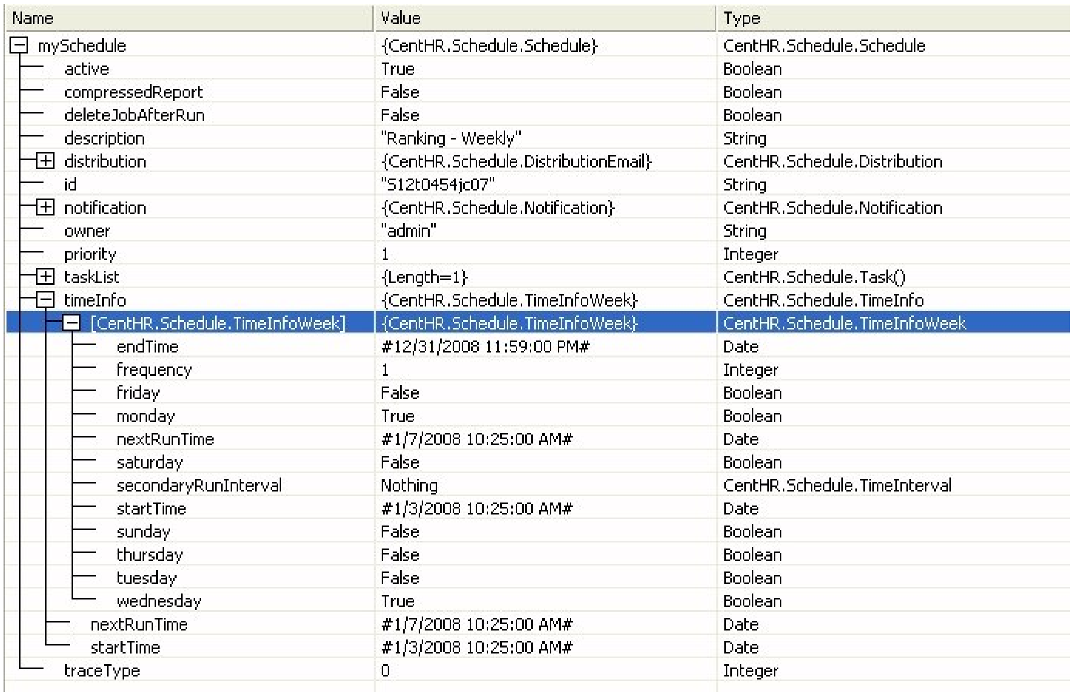
The TimeInfoWeek structure represents time information pertaining to scheduled reports that run in intervals of weeks.

Name	Type	Description
EndTime	Date	Calendar information indicating the end time for a scheduled event. Report Broker will not run a job for this schedule after this date.
Frequency	Integer	The frequency for a scheduled event, in weeks. For example, if an e-mail report distribution is set to run every five weeks, the frequency would be 5.
Friday	Boolean	Whether or not the Report Broker job is scheduled for a Friday. If true, the job will run on a Friday.
Monday	Boolean	Whether or not the Report Broker job is scheduled for a Monday. If true, the job will run on a Monday.
Saturday	Boolean	Whether or not the Report Broker job is scheduled for a Saturday. If true, the job will run on a Saturday.

Name	Type	Description
Sunday	Boolean	Whether or not the Report Broker job is scheduled for a Sunday. If true, the job will run on a Sunday.
Thursday	Boolean	Whether or not the Report Broker job is scheduled for a Thursday. If true, the job will run on a Thursday.
Tuesday	Boolean	Whether or not the Report Broker job is scheduled for a Tuesday. If true, the job will run on a Tuesday.
Wednesday	Boolean	Whether or not the Report Broker job is scheduled for a Wednesday. If true, the job will run on a Wednesday.
SecondaryRunInterval	TimeInterval	A structure that defines a secondary run interval within the day the schedule runs. The interval is setup to run on a minute(s) basis for a specified number of minutes or until a specified time.
NextRunTime	Date	Calendar information indicating the next run time for a scheduled event. NextRunTime is generally used only as an internal function. It is not recommended for general usage and is not formally supported. Manually re-setting system designated fields might result in unpredictable behavior. However, the expected behavior if a user or programmer uses NextRunTime to designate a next run time that is different from the run time calculated internally is that the scheduled job will run at the new next run time. It then resumes normal calculation.

Name	Type	Description
StartTime	Date	Calendar information indicating the start time for a scheduled event. The start time is designated as the first time a new schedule is set to run. Creating a new schedule and altering any jobs that are to run in the future will create an entirely new start time. The default start time is the current time. After the first time a schedule runs, subsequent run times can be obtained via NextRunTime.

The following is an example of the TimeInfoWeek structure.



Name	Value	Type
mySchedule	{CentHR.Schedule.Schedule}	CentHR.Schedule.Schedule
active	True	Boolean
compressedReport	False	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking - Weekly"	String
distribution	{CentHR.Schedule.DistributionEmail}	CentHR.Schedule.Distribution
id	"S12t0454jc07"	String
notification	{CentHR.Schedule.Notification}	CentHR.Schedule.Notification
owner	"admin"	String
priority	1	Integer
taskList	{Length=1}	CentHR.Schedule.Task()
timeInfo	{CentHR.Schedule.TimeInfoWeek}	CentHR.Schedule.TimeInfo
[CentHR.Schedule.TimeInfoWeek]	{CentHR.Schedule.TimeInfoWeek}	CentHR.Schedule.TimeInfoWeek
endTime	#12/31/2008 11:59:00 PM#	Date
frequency	1	Integer
friday	False	Boolean
monday	True	Boolean
nextRunTime	#1/7/2008 10:25:00 AM#	Date
saturday	False	Boolean
secondaryRunInterval	Nothing	CentHR.Schedule.TimeInterval
startTime	#1/3/2008 10:25:00 AM#	Date
sunday	False	Boolean
thursday	False	Boolean
tuesday	False	Boolean
wednesday	True	Boolean
nextRunTime	#1/7/2008 10:25:00 AM#	Date
startTime	#1/3/2008 10:25:00 AM#	Date
traceType	0	Integer

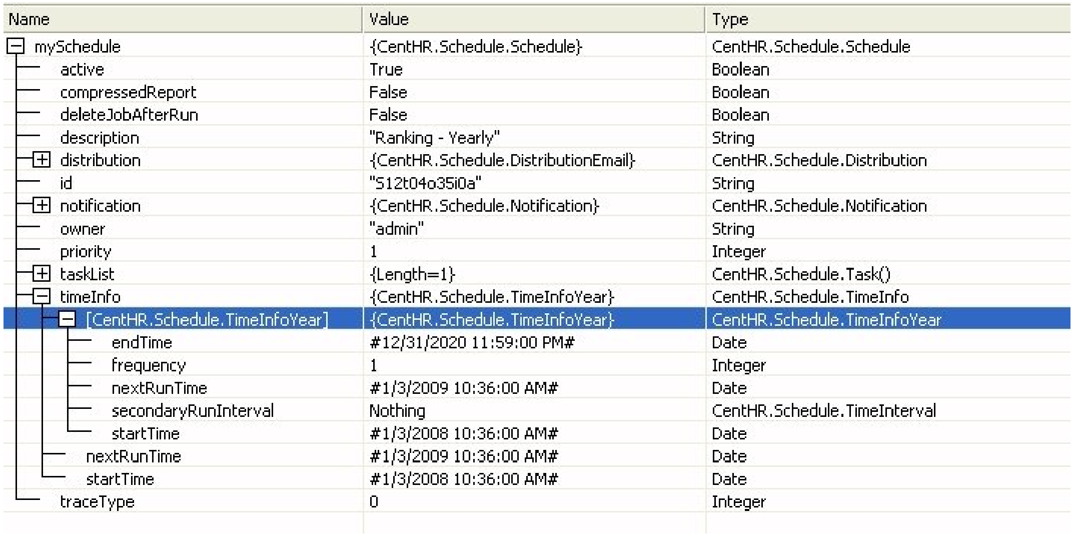
TimeInfoYear Structure (Yearly Schedule Structure)

The TimeInfoYear structure represents time information pertaining to scheduled reports that run in intervals of years.

Name	Type	Description
EndTime	Date	Calendar information indicating the end time for a scheduled event. Report Broker will not run a job for this schedule after this date.
Frequency	Integer	Assigns the frequency for a scheduled event, in years. For example, if an e-mail report distribution is set to run every five years, the frequency would be 5.
SecondaryRunInterval	TimeInterval	A structure that defines a secondary run interval within the day the schedule runs. The interval is setup to run on a minute(s) basis for a specified number of minutes or until a specified time.
NextRunTime	Date	Calendar information indicating the next run time for a scheduled event. NextRunTime is generally used only as an internal function. It is not recommended for general usage and is not formally supported. Manually re-setting system designated fields might result in unpredictable behavior. However, the expected behavior if a user or programmer uses NextRunTime to designate a next run time that is different from the run time calculated internally is that the scheduled job will run at the new next run time. It then resumes normal calculation.

Name	Type	Description
StartTime	Date	Calendar information indicating the start time for a scheduled event. The start time is designated as the first time a new schedule is set to run. Creating a new schedule and altering any jobs that are to run in the future will create an entirely new start time. The default start time is the current time. After the first time a schedule runs, subsequent run times can be obtained via NextRunTime.

The following is an example of the TimeInfoYear structure.



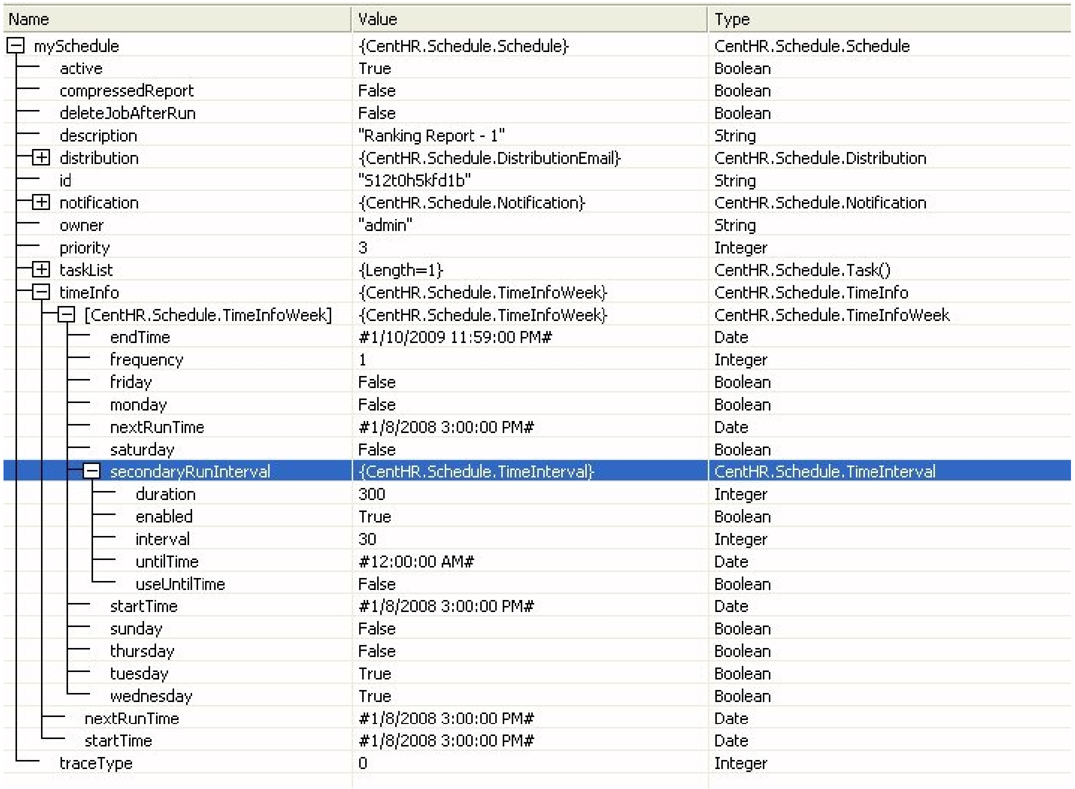
Name	Value	Type
mySchedule	{CentHR.Schedule.Schedule}	CentHR.Schedule.Schedule
active	True	Boolean
compressedReport	False	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking - Yearly"	String
distribution	{CentHR.Schedule.DistributionEmail}	CentHR.Schedule.Distribution
id	"S12t04o35i0a"	String
notification	{CentHR.Schedule.Notification}	CentHR.Schedule.Notification
owner	"admin"	String
priority	1	Integer
taskList	{Length=1}	CentHR.Schedule.Task()
timeInfo	{CentHR.Schedule.TimeInfoYear}	CentHR.Schedule.TimeInfo
[CentHR.Schedule.TimeInfoYear]	{CentHR.Schedule.TimeInfoYear}	CentHR.Schedule.TimeInfoYear
endTime	#12/31/2020 11:59:00 PM#	Date
frequency	1	Integer
nextRunTime	#1/3/2009 10:36:00 AM#	Date
secondaryRunInterval	Nothing	CentHR.Schedule.TimeInterval
startTime	#1/3/2008 10:36:00 AM#	Date
nextRunTime	#1/3/2009 10:36:00 AM#	Date
startTime	#1/3/2008 10:36:00 AM#	Date
traceType	0	Integer

TimeInterval (Secondary Run Time and Task Retry Structure)

The TimeInterval structure is used to define a secondary run interval within the day the schedule runs for Daily, Weekly, Monthly, and Yearly schedules. This structure is also used for defining the time interval for restarting a task if the task did not complete successfully. The interval is setup to run on a minute(s) basis for a specified number of minutes or until a specified time. The secondary run interval cannot exceed the NEXTRUNTIME for the primary run interval. For example, a daily schedule cannot have a secondary run interval greater than 1 day. If a secondary run interval is scheduled to run after the schedule's next primary run interval, the secondary run interval is stopped and an error message is displayed to the user and written to the log file.

Name	**Type**	**Description**
Duration	Integer	The duration, specified in minutes, during which the time interval will be applied. UseUntilTime must be false.
Enabled	Boolean	true - Time Interval settings are active. false - Time Interval settings are inactive.
Interval	Integer	Applies the time interval every *n* minutes.
UntilTime	Date	The end time for which the time interval will be applied. UseUntilTime must be true.
UseUntilTime	Boolean	true - the Until Time value determines the end of the time interval. false - the Duration value determines the end of the time interval.

The following is an example of the TimeInterval structure. It will run the schedule every 30 minutes for 5 hours (300 minutes) starting at 3:00PM.



```
Name                                      Value                                 Type
mySchedule                                {CentHR.Schedule.Schedule}            CentHR.Schedule.Schedule
    active                                True                                  Boolean
    compressedReport                      False                                 Boolean
    deleteJobAfterRun                     False                                 Boolean
    description                           "Ranking Report - 1"                  String
    distribution                          {CentHR.Schedule.DistributionEmail}   CentHR.Schedule.Distribution
    id                                    "S12t0h5kfd1b"                        String
    notification                          {CentHR.Schedule.Notification}        CentHR.Schedule.Notification
    owner                                 "admin"                               String
    priority                              3                                     Integer
    taskList                              {Length=1}                            CentHR.Schedule.Task()
    timeInfo                              {CentHR.Schedule.TimeInfoWeek}        CentHR.Schedule.TimeInfo
        [CentHR.Schedule.TimeInfoWeek]    {CentHR.Schedule.TimeInfoWeek}        CentHR.Schedule.TimeInfoWeek
            endTime                       #1/10/2009 11:59:00 PM#               Date
            frequency                     1                                     Integer
            friday                        False                                 Boolean
            monday                        False                                 Boolean
            nextRunTime                   #1/8/2008 3:00:00 PM#                 Date
            saturday                      False                                 Boolean
            secondaryRunInterval          {CentHR.Schedule.TimeInterval}        CentHR.Schedule.TimeInterval
                duration                  300                                   Integer
                enabled                   True                                  Boolean
                interval                  30                                    Integer
                untilTime                 #12:00:00 AM#                         Date
                useUntilTime              False                                 Boolean
            startTime                     #1/8/2008 3:00:00 PM#                 Date
            sunday                        False                                 Boolean
            thursday                      False                                 Boolean
            tuesday                       True                                  Boolean
            wednesday                     True                                  Boolean
        nextRunTime                       #1/8/2008 3:00:00 PM#                 Date
        startTime                         #1/8/2008 3:00:00 PM#                 Date
    traceType                             0                                     Integer
```

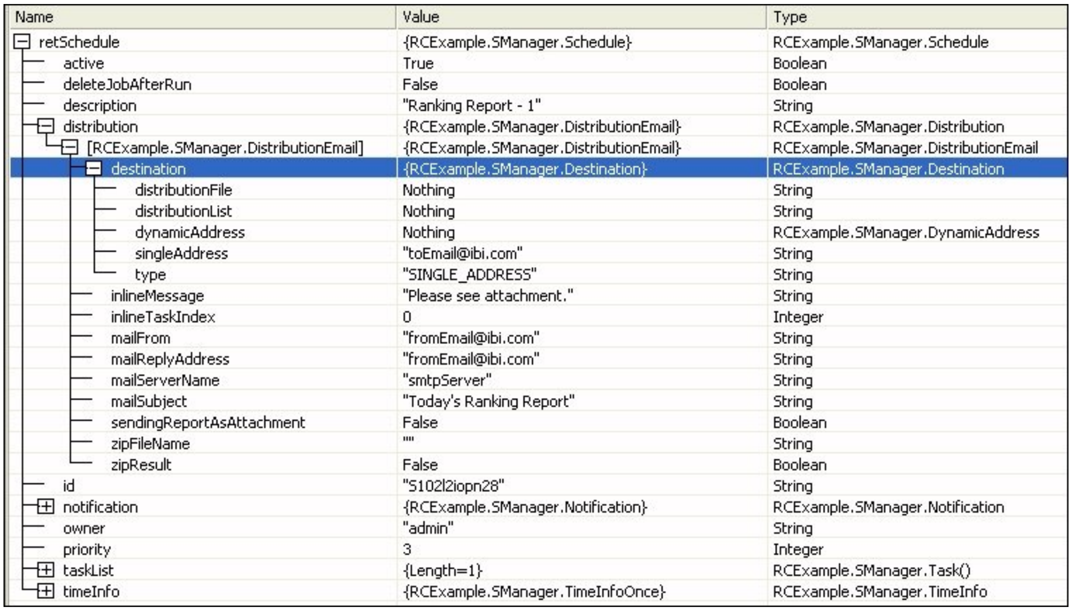
Destination (Destination Structure)

When Report Broker distributes output, it determines the recipient(s) by accessing the destination definition. A destination definition possesses one or many recipients of the scheduled distribution.

Name	Type	Description
DistributionList	String	A list of one or many recipients stored within the Report Broker Address Book. Type must be set to ''DISTRIBUTION_LIST''.

Name	**Type**	**Description**
DistributionFile	String	A list of one or many recipients stored within a physical file accessible to the Distribution Server. Type must be set to ''DISTRIBUTION_FILE''.
SingleAddress	String	A single recipient and is entered by the person scheduling a job when the job is created. Type must be set to ''SINGLE_ADDRESS''.
DynamicAddress	dynamicAddress	A structure containing the definition of the Reporting Server procedure to run to obtain a list of one or more recipients. Type must be set to ''DYNAMIC_ADDRESS''.
Type	String	Valid values are: ❏ DISTRIBUTION_FILE ❏ DISTRIBUTION_LIST ❏ DYNAMIC_ADDRESS ❏ SINGLE_ADDRESS

The following is an example of the Destination structure.



Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.DistributionEmail}	RCExample.SManager.Distribution
[RCExample.SManager.DistributionEmail]	{RCExample.SManager.DistributionEmail}	RCExample.SManager.DistributionEmail
destination	{RCExample.SManager.Destination}	RCExample.SManager.Destination
distributionFile	Nothing	String
distributionList	Nothing	String
dynamicAddress	Nothing	RCExample.SManager.DynamicAddress
singleAddress	"toEmail@ibi.com"	String
type	"SINGLE_ADDRESS"	String
inlineMessage	"Please see attachment."	String
inlineTaskIndex	0	Integer
mailFrom	"fromEmail@ibi.com"	String
mailReplyAddress	"fromEmail@ibi.com"	String
mailServerName	"smtpServer"	String
mailSubject	"Today's Ranking Report"	String
sendingReportAsAttachment	False	Boolean
zipFileName	""	String
zipResult	False	Boolean
id	"S102l2iopn28"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
timeInfo	{RCExample.SManager.TimeInfoOnce}	RCExample.SManager.TimeInfo

DistributionEmail (E-mail Distribution Structure)

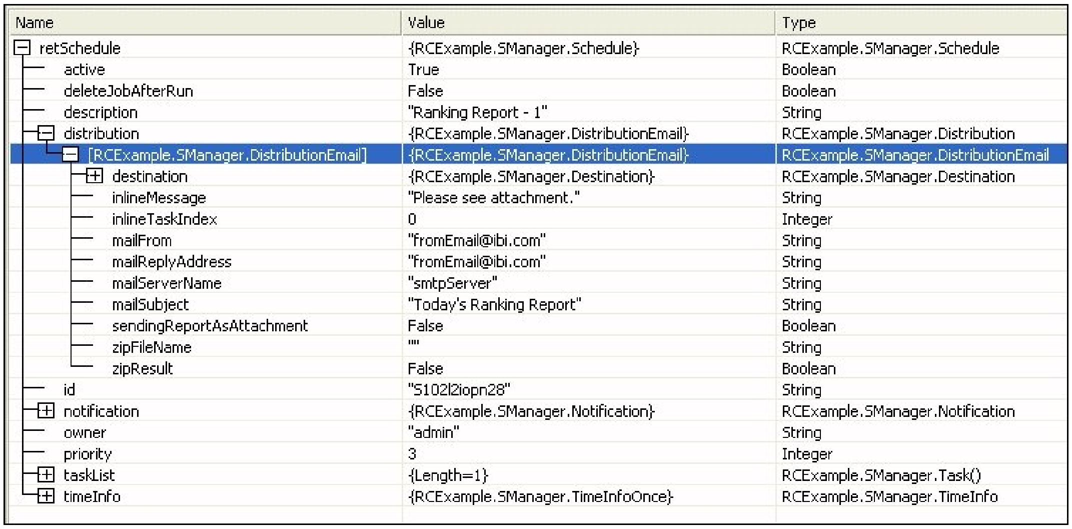
E-mail distribution is one of three distribution types supported by Report Broker. The DistributionEmail structure is used when the intended distribution method for the scheduled Report Broker job is via e-mail.

Name	**Type**	**Description**
Destination	Destination	The destination structure that can contain one of the four types: Distribution List, Distribution File, Single Address, and Dynamic Address. These destination types contain one or more recipients of the scheduled distribution. In this case where distribution is via e-mail, the recipients are listed as e-mail addresses.

Name	Type	Description
InlineMessage	String	The inline message associated with an e-mail report distribution. An inline message is the message contained in the body of the e-mail when the report is sent as an attachment. If the report is sent inline, this should not be set. The size limit for an inline message is 255 characters.
InlineTaskIndex	Integer	The index of the task that is going to be inline (in the body of the e-mail). It should always be set to 0.
MailFrom	String	The e-mail address associated with the From header field of a scheduled e-mail distribution. The size limit for MailFrom is 65 characters.
MailReplyAddress	String	The reply e-mail address from Reply Address header field of a scheduled e-mail distribution. The size limit for mail reply address is 65 characters.
mailServerName	String	An SMTP mail server name associated with scheduled e-mail distribution. The size limit for mail server name is 65 characters.
MailSubject	String	An e-mail subject corresponding to the "Subject" header field associated with scheduled e-mail distribution. The size limit for mail subject is 90 characters.
SendingReportAsAttachment	Boolean	True. Send report as an attachment. False. Send report within the body of the e-mail.

Name	Type	Description
ZipFileName	String	The name of the zip file associated with a scheduled e-mail distribution. The zip result should be set to true. The size limit for zip file name is 64 characters.
ZipResult	Boolean	True. Zip output. False. Do not zip output.

The following is an example of the Distribution Email structure.



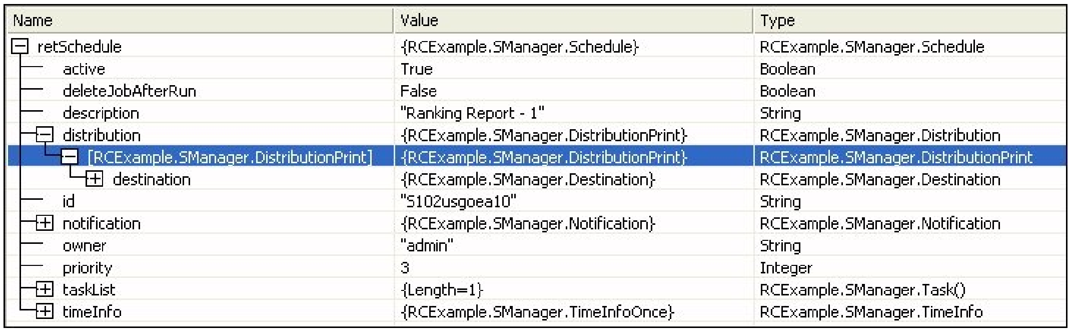
Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.DistributionEmail}	RCExample.SManager.Distribution
[RCExample.SManager.DistributionEmail]	{RCExample.SManager.DistributionEmail}	RCExample.SManager.DistributionEmail
destination	{RCExample.SManager.Destination}	RCExample.SManager.Destination
inlineMessage	"Please see attachment."	String
inlineTaskIndex	0	Integer
mailFrom	"fromEmail@ibi.com"	String
mailReplyAddress	"fromEmail@ibi.com"	String
mailServerName	"smtpServer"	String
mailSubject	"Today's Ranking Report"	String
sendingReportAsAttachment	False	Boolean
zipFileName	""	String
zipResult	False	Boolean
id	"S102l2iopn28"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
timeInfo	{RCExample.SManager.TimeInfoOnce}	RCExample.SManager.TimeInfo

DistributionPrint (Print Distribution Structure)

Print distribution is one of three distribution types supported by Report Broker. The DistributionPrint structure is used when the intended distribution method for the scheduled Report Broker job is via a printer.

Name	Type	Description
Destination	Destination	The destination structure that contains information indicating the target(s) of distribution.

The following is an example of the DistributionPrint structure.



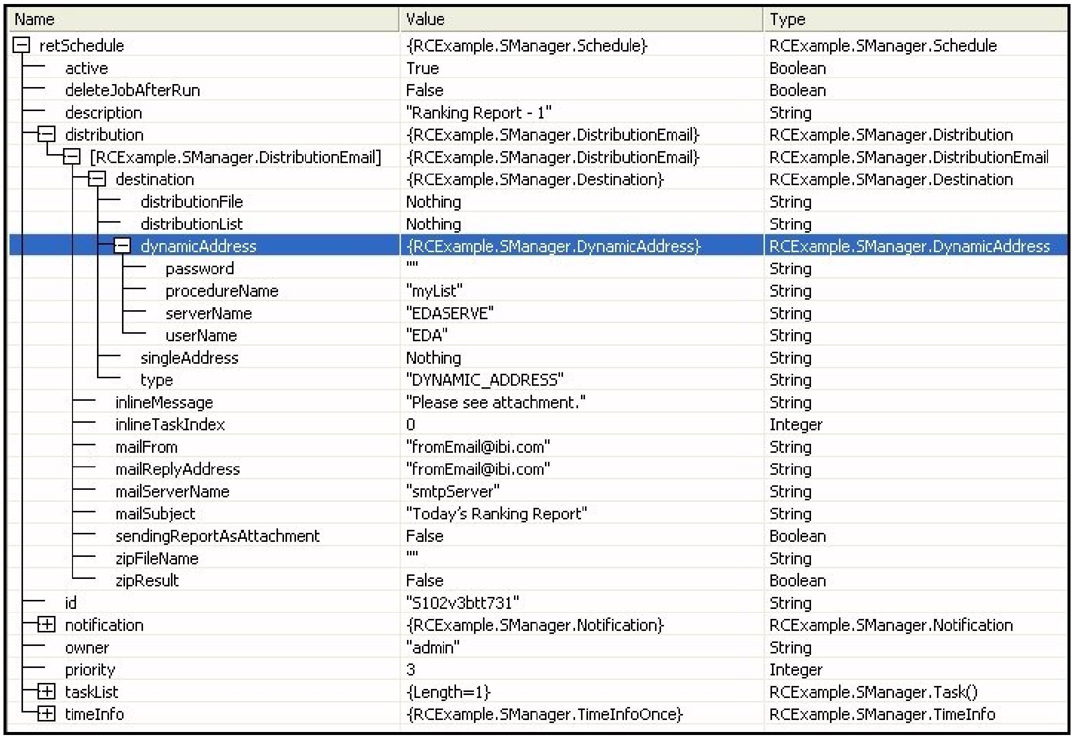
Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.DistributionPrint}	RCExample.SManager.Distribution
[RCExample.SManager.DistributionPrint]	{RCExample.SManager.DistributionPrint}	RCExample.SManager.DistributionPrint
destination	{RCExample.SManager.Destination}	RCExample.SManager.Destination
id	"S102usgoea10"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
timeInfo	{RCExample.SManager.TimeInfoOnce}	RCExample.SManager.TimeInfo

DynamicAddress (Dynamic Address Structure)

The DynamicAddress structure is used when a Report Broker distribution list is created dynamically at run time. The program creating a dynamic distribution list is a procedure residing on a Reporting Server.

Name	Type	Description
Password	String	The value of the password required for authentication to the Reporting Server containing the Web Query procedure that creates the dynamic distribution list.
ProcedureName	String	The name of the Web Query procedure that produces the dynamic distribution list. Accepts a maximum value of 64 characters.
ServerName	String	The name of the Reporting Server that contains the Web Query procedure that creates the dynamic distribution list.
UserName	String	The user ID to the Reporting Server that contains the Web Query procedure that creates the dynamic distribution list.

The following is an example of the DynamicAddress structure.



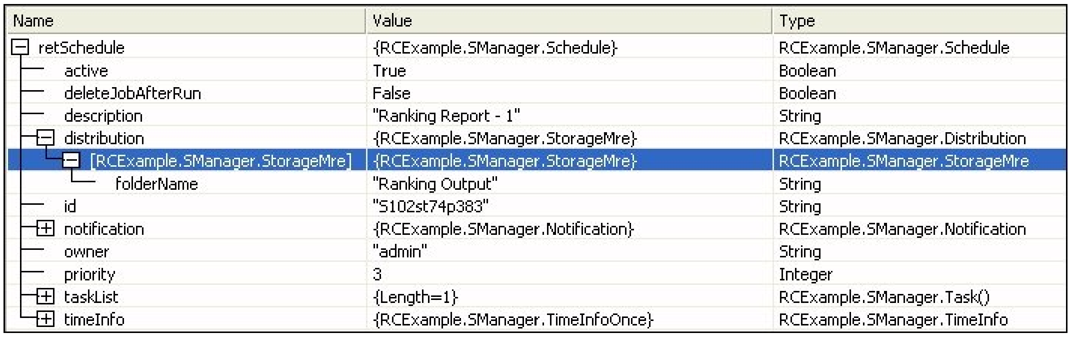
Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.DistributionEmail}	RCExample.SManager.Distribution
[RCExample.SManager.DistributionEmail]	{RCExample.SManager.DistributionEmail}	RCExample.SManager.DistributionEmail
destination	{RCExample.SManager.Destination}	RCExample.SManager.Destination
distributionFile	Nothing	String
distributionList	Nothing	String
dynamicAddress	{RCExample.SManager.DynamicAddress}	RCExample.SManager.DynamicAddress
password	""	String
procedureName	"myList"	String
serverName	"EDASERVE"	String
userName	"EDA"	String
singleAddress	Nothing	String
type	"DYNAMIC_ADDRESS"	String
inlineMessage	"Please see attachment."	String
inlineTaskIndex	0	Integer
mailFrom	"fromEmail@ibi.com"	String
mailReplyAddress	"fromEmail@ibi.com"	String
mailServerName	"smtpServer"	String
mailSubject	"Today's Ranking Report"	String
sendingReportAsAttachment	False	Boolean
zipFileName	""	String
zipResult	False	Boolean
id	"S102v3btt731"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
timeInfo	{RCExample.SManager.TimeInfoOnce}	RCExample.SManager.TimeInfo

StorageMRE (MRE Storage Structure)

MRE Storage is one of three distribution types supported by Report Broker. The StorageMRE is used when the intended distribution method for the scheduled Report Broker job is to store the output in the Web Query environment.

Name	Type	Description
FolderName	String	The name of the folder to which you are sending a report for scheduled distribution to Web Query.

The following is an example of the StorageMRE structure.



Name	Value	Type
retSchedule	{RCExample.SManager.Schedule}	RCExample.SManager.Schedule
active	True	Boolean
deleteJobAfterRun	False	Boolean
description	"Ranking Report - 1"	String
distribution	{RCExample.SManager.StorageMre}	RCExample.SManager.Distribution
[RCExample.SManager.StorageMre]	{RCExample.SManager.StorageMre}	RCExample.SManager.StorageMre
folderName	"Ranking Output"	String
id	"S102st74p383"	String
notification	{RCExample.SManager.Notification}	RCExample.SManager.Notification
owner	"admin"	String
priority	3	Integer
taskList	{Length=1}	RCExample.SManager.Task()
timeInfo	{RCExample.SManager.TimeInfoOnce}	RCExample.SManager.TimeInfo

IBM

4 Web Query Web Services Functions

Web Query Web Services contains a set of functions that are used to perform certain Web Query functionality. Each of these functions has an input and output definition associated with it. This chapter includes Visual Basic .NET and Java examples to illustrate how each function should be called within those programming environments.

Topics:

Functions

In this section:

This topic describes the functions contained within Web Query Web Services.

Web Query is the Web Service name associated with the set of Web Query Web Services functions. The Service Name <service name="WebQuery"> can be found towards the end of the WSDL file that is created in the steps described in *Using the Web Query WSDL Utility* on page 15.

Note: Java requires function names to be referenced as WebQueryLogOn. Function names are case sensitive.

Authentication

Function Name: WebQueryLogOn

Purpose: To authenticate against the security set up in the Web Query environment. If the authentication is successful, the Web Query cookies are set and the cookie information is returned. This cookie information is the first parameter of every subsequent Web Query Web Services function call.

Input:

Description	Type
Web Query Reporting Server user ID. This parameter is always set to null ("").	String
Web Query Reporting Server password. This parameter is always set to null ("").	String

Description	Type
Web Query user ID.	String
Web Query password.	String

Output:

Description	Type
Structure that contains cookie information.	*LogOnInfo*

Example: Authentication Status in Visual Basic .NET

In the following example, the status of authentication is written to the WebQueryLogOn.txt file in the c:\temp directory.

```
Dim wfs As New MR.WebQuery
Dim logon As New MR.LogOnInfo
Dim newOutput As String = ""
Dim tempfile As String

logon = wfs.WebQueryLogOn("","","RepUser","RepPass")

newOutput = logon.status
tempfile = "c:\temp\WebQueryLogOn.txt"
FileOpen(1, tempfile, OpenMode.Output)
Print(1, newOutput)
FileClose(1)
```

Note: MR is the name of the Web Reference.

Example: **Authentication Status in Java**

In the following example, the status of authentication is written to the WebQueryLogOn.txt file in the c:\temp directory.

```
try {
WebQuery WFservice = new WebQueryLocator();
WebQuerySoap_PortType wfs = WFservice.getWebQuerySoap();

LogOnInfo logon = wfs.webQueryLogOn("","","RepUser","RepPass");

boolean newOutput=logon.isStatus();
File tempfile = new File("c:\\temp\\WebQueryLogOn.txt");
FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);
out.println(newOutput);
out.close();
    }
catch (Throwable t)
 {
 System.err.println(t);
 t.printStackTrace();
 System.exit(1);
 }
```

Running a Web Query Report

Function Name: WebQueryRunFex

Purpose: To run a Web Query report.

Input:

Description	Type
Web Query cookie information.	*LogOnInfo*
Web Query report run information.	*FexInfo*

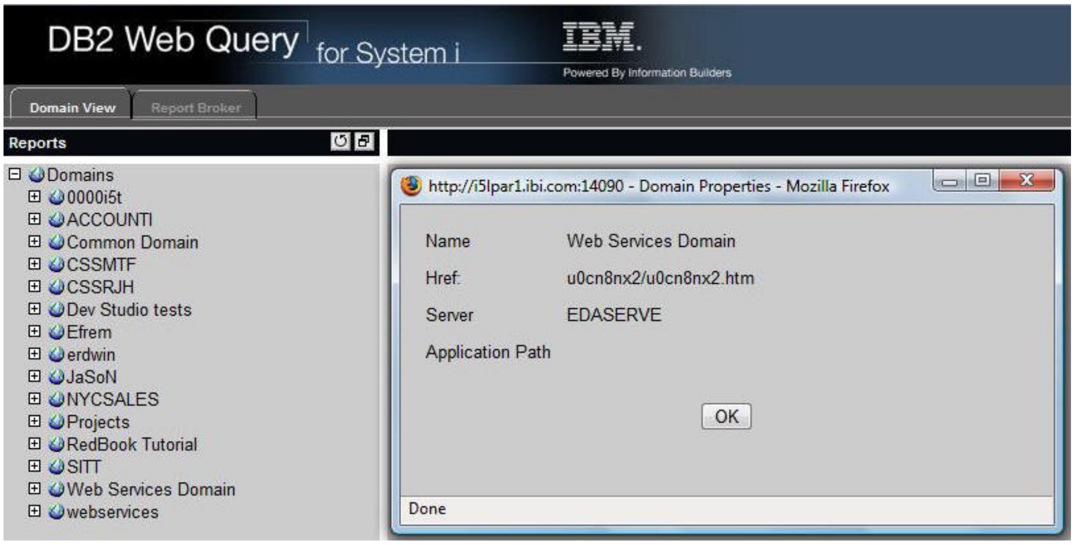
Output:

Description	Type
Structure that contains the output from a Web Query report.	*WebQueryReturn*

Example: Running a Web Query Report in Visual Basic .NET

In the following example, the output of a Web Query report named GetQuotes is written from Web Query to the GetQuotes.htm file in the c:\temp directory. The ticker symbol used as input is MSFT (Microsoft Corporation).

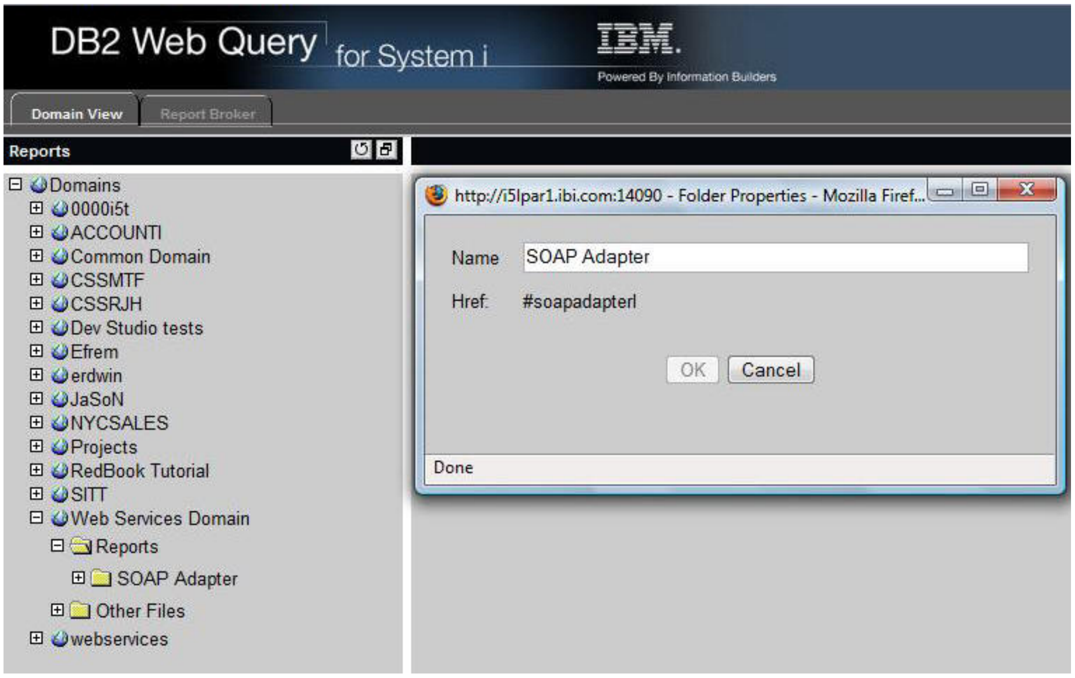
The HREF for the domain is used as input instead of the domain name. The following image shows an example of the HREF for a domain.



http://i5lpar1.ibi.com:14090 - Domain Properties - Mozilla Firefox

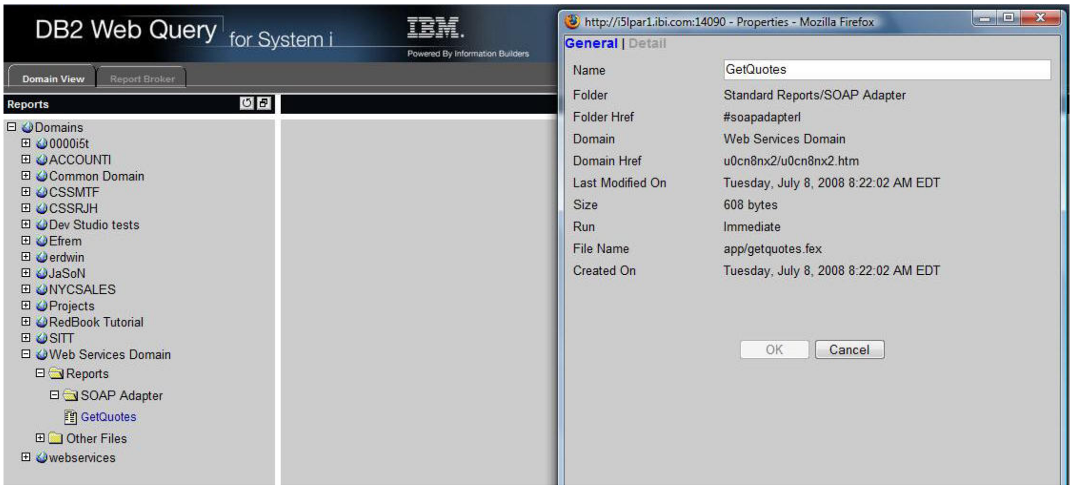
Name	Web Services Domain
Href:	u0cn8nx2/u0cn8nx2.htm
Server	EDASERVE
Application Path	

The HREF for the Standard Report group is used as input instead of the Standard Report group name. The following image shows an example of the HREF for a Standard Report group.



Name: SOAP Adapter
Href: #soapadapterl

The file name or HREF for the Web Query report is used as input instead of the Web Query report name. The following image shows an example of a file name for a Web Query report.



Name	GetQuotes
Folder	Standard Reports/SOAP Adapter
Folder Href	#soapadapterl
Domain	Web Services Domain
Domain Href	u0cn8nx2/u0cn8nx2.htm
Last Modified On	Tuesday, July 8, 2008 8:22:02 AM EDT
Size	608 bytes
Run	Immediate
File Name	app/getquotes.fex
Created On	Tuesday, July 8, 2008 8:22:02 AM EDT

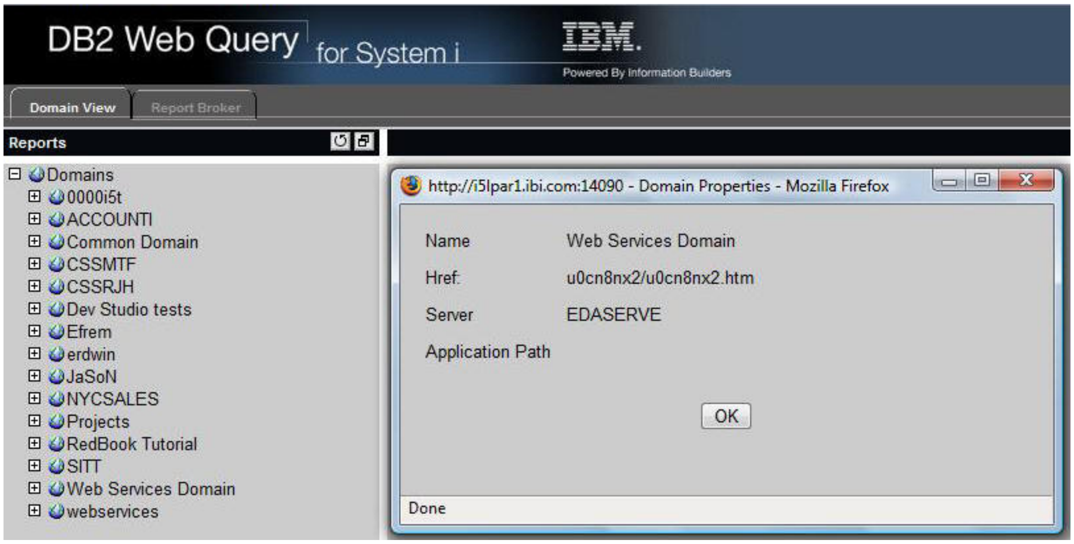
```
Dim wfs As New MR.WebQuery
Dim logon As New MR.LogOnInfo
Dim ret As New MR.WebQueryReturn
Dim report As New MR.FexInfo
Dim param1 As New MR.ValuesArrayEntry
Dim params As Array = Array.CreateInstance(GetType(MR.ValuesArrayEntry),1)
Dim newOutput As String = ""
Dim tempfile As String

param1.name = "TICKER"
param1.val = "MSFT"
params(0) = param1

report.MREdomain = "u0cn8nx2/u0cn8nx2.htm"
report.MREfolder = "#soapadapterl"
report.name = "app/getquotes.fex"
report.IBIWS_arrayvalues = params

logon = wfs.WebQueryLogOn("","","RepUser","RepPass")
ret = wfs.WebQueryRunFex(logon, report)

newOutput = ret.output

tempfile = "c:\temp\GetQuotes.htm"

FileOpen(1, tempfile, OpenMode.Output)
Print(1, newOutput)
FileClose(1)
```

Note: MR is the name of the Web Reference. Fex names are case sensitive.

Example: Running a Managed Web Query Report in Java

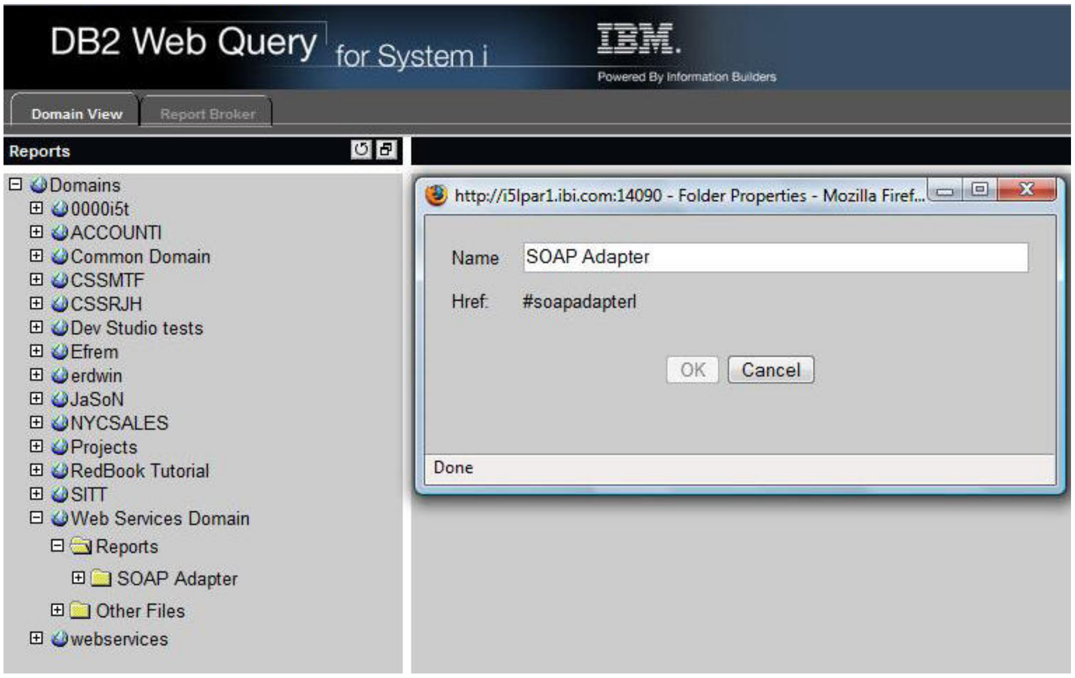
In the following example, the output of a Web Query report named GetQuotes is written from Web Query to the GetQuotes.htm file in the c:\temp directory. The ticker symbol used as input is MSFT (Microsoft Corporation).

The HREF for the domain is used as input instead of the domain name. The following image shows an example of the HREF for a domain.



Name	Web Services Domain
Href:	u0cn8nx2/u0cn8nx2.htm
Server	EDASERVE
Application Path	

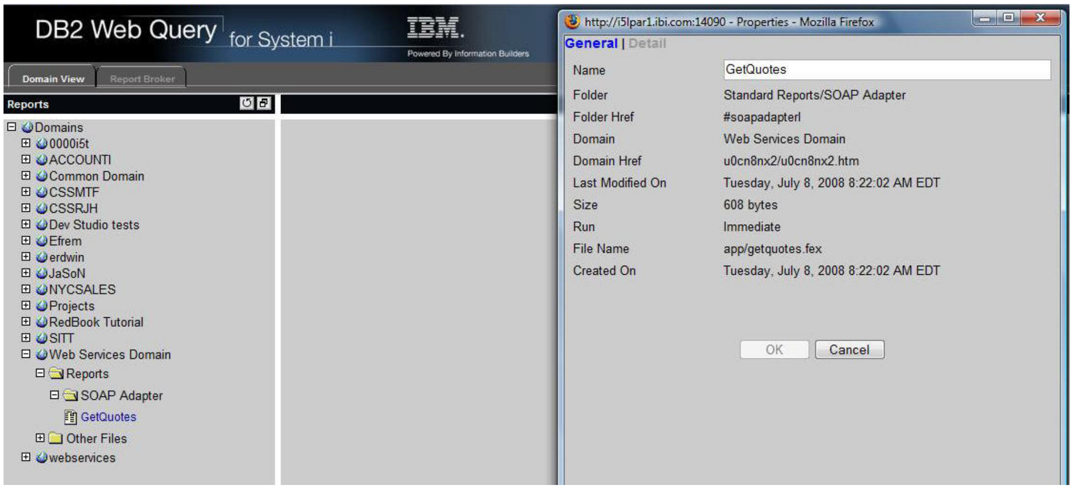
The HREF for the Standard Report group is used as input instead of the Standard Report group name. The following image shows an example of the HREF for a Standard Report group.



Name: SOAP Adapter

Href: #soapadapterl

The file name or HREF for the Web Query report is used as input instead of the Web Query report name. The following image shows an example of a file name for a Web Query report.



Name	GetQuotes
Folder	Standard Reports/SOAP Adapter
Folder Href	#soapadapterl
Domain	Web Services Domain
Domain Href	u0cn8nx2/u0cn8nx2.htm
Last Modified On	Tuesday, July 8, 2008 8:22:02 AM EDT
Size	608 bytes
Run	Immediate
File Name	app/getquotes.fex
Created On	Tuesday, July 8, 2008 8:22:02 AM EDT

```
try {
WebQuery WFservice = new WebQueryLocator();
WebQuerySoap_PortType wfs = WFservice.getWebQuerySoap();

ValuesArrayEntry[] param;
param = new ValuesArrayEntry[1];

ValuesArrayEntry param1 = new ValuesArrayEntry();
param1.setName("TICKER");
param1.setVal("MSFT");
param[0] = param1;

FexInfo report = new FexInfo();
report.setMREdomain("u0cn8nx2/u0cn8nx2.htm");
report.setMREfolder("#soapadapterl");
report.setName("app/getquotes.fex");
report.setIBIWS_arrayvalues(param);

LogOnInfo logon = wfs.webQueryLogOn("","","RepUser","RepPass");
WebQueryReturn ret = wfs.webQueryRunFex(logon,report);

String newOutput=ret.getOutput();

File tempfile = new File("c:\\temp\\GetQuotes.htm");

FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);
out.println(newOutput);
out.close();
    }
catch (Throwable t)
 {
 System.err.println(t);
 t.printStackTrace();
 System.exit(1);
 }
```

Finding the Parameters of a Web Query Report

Function Name: WebQueryFexReflection

Purpose: To retrieve the parameters of a Web Query report.

Input:

Description	Type
Web Query cookie information.	*LogOnInfo*

Description	Type
Web Query reporting run information.	*FexInfo*

Output:

Description	Type
Structure that contains information about running the Web Query report. The parameters are retrieved into the ValuesArrayEntry structure that is a sub-structure of the FexInfo structure.	*FexInfo*

Example: Finding the Parameters of a Web Query Report in Visual Basic .NET

In the following example, the parameter information from a Web Query report named CAR54 is written to the Parameters.txt file in the c:\temp directory.

```
Dim wfs As New WQ.WebQuery
Dim logon As New WQ.LogOnInfo
Dim fexinfoIn As New WQ.FexInfo
Dim fexinfoOut As New WQ.FexInfo
Dim newOutput As String = ""
Dim tempfile As String
Dim I As Integer

report.MREdomain = "u0cn8nx2/u0cn8nx2.htm"
report.MREfolder = "#soapadapterl"
report.name = "app/getquotes.fex"

logon = wfs.WebQueryLogOn("","","RepUser","RepPass")
fexinfoOut = wfs.WebQueryFexReflection(logon, fexinfoIn)


tempfile = "c:\temp\Parameters.txt"
FileOpen(1, tempfile, OpenMode.Output)

For I = 0 To fexinfoOut.IBIWS_arrayvalues.Length - 1
       newOutput = fexinfoOut.IBIWS_arrayvalues(I).prompt + "    " _
                   + fexinfoOut.IBIWS_arrayvalues(I).name + "    " _
                   + fexinfoOut.IBIWS_arrayvalues(I).defaultVal
       PrintLine(1, newOutput)

Next I

FileClose(1)
```

Note: WQ is the name of the Web Reference.

Example: Finding the Parameters of a Web Query Report in Java

In the following example, the parameter information from a Web Query report named CAR54 is written to the Parameters.txt file in the c:\temp directory.

```
try {
WebQuery WFservice = new WebQueryLocator();
WebQuerySoap_PortType wfs = WFservice.getWebQuerySoap();
FexInfo fexinfoIn = new FexInfo();
ValuesArrayEntry[] Values;
report.setMREdomain("u0cn8nx2/u0cn8nx2.htm ");
report.setMREfolder("#soapadapterl");
report.setName("app/getquotes.fex");

LogOnInfo logon = wfs.webQueryLogOn("","","RepUser","RepPass");
FexInfo fexinfoOut = wfs.webQueryFexReflection(logon,fexinfoIn);

String newOutput = null;
File tempfile = new File("c:\\temp\\Parameters.txt");
FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);


Values = fexinfoOut.getIBIWS_arrayvalues();

for ( int I=0; i<Values.length; I++ )
    {
    newOutput = Values[i].getPrompt() + "    "
                + Values[i].getName() + "    "
                + Values[i].getDefaultVal();
    out.println(newOutput);
    }
out.close();
    }
catch (Throwable t)
 {
 System.err.println(t);
 t.printStackTrace();
 System.exit(1);
 }
```

Running Links Brought Back in a Web Query Report

Function Name: WebQueryLink

Purpose: To run the links brought back in a Web Query report, such as drill-down information and information to obtain a graph created by a Web Query report. If the URL is being captured within a program when you click a link within a Web Query report, you should use the WebQueryDrill function instead.

Input:

Description	Type
Web Query cookie information.	*LogOnInfo*
LinkArrayEntry structure obtained in the *WebQueryReturn* structure.	LinkArrayEntry

Output:

Description	Type
Structure containing the output from a Web Query report.	*WebQueryReturn*

When you use the LinkArrayEntry structure for drill-down information, the array of links are brought back in pairs within the WebQueryReturn structure if the output format is HTML. The first array entry would have a type of urlstart and the second array entry would have a type of fexdrill. You can only pass link types of fexdrill as a parameter to WebQueryLink for a drill-down report.

Example: Running Links Brought Back in a Web Query Report in Visual Basic .NET

In the following example, the output of the result of a drill-down report from a Web Query report named CENTPL_ACTBUD is written to the Centpl_ActBud.htm file in the c:\temp directory. The first line of the report is used in the drill-down.

```
Dim wfs As New WQ.WebQuery
Dim logon As New WQ.LogOnInfo
Dim ret As New WQ.WebQueryReturn
Dim retdrill As New WQ.WebQueryReturn
Dim report As New WQ.FexInfo
Dim newOutput As String = ""
Dim tempfile As String

report.MREdomain = "samplehj/samplehj.htm"
report.MREfolder = "#carreportsa4"
report.name = "app/carmain.fex"

logon = wfs.WebQueryLogOn("","","RepUser","RepPass")
ret = wfs.WebQueryRunFex(logon, report)
retdrill = wfs.WebQueryLink(logon, ret.links(3))

newOutput = retdrill.output

tempfile = "c:\temp\Centpl_ActBud.htm"

FileOpen(1, tempfile, OpenMode.Output)
Print(1, newOutput)
FileClose(1)
```

Note: WQ is the name of the Web Reference.

Example: **Running Links Brought Back in a Web Query Report in Java**

In the following example, the output of the result of a drill-down report from a Web Query report named CENTPL_ACTBUD is written to the Centpl_ActBud.htm file in the c:\temp directory. The first line of the report is used in the drill-down.

```
try {
WebQuery WFservice = new WebQueryLocator();
WebQuerySoap_PortType wfs = WFservice.getWebQuerySoap();
LinkArrayEntry[] Links;

FexInfo report = new FexInfo();
report.setMREdomain("samplehj/samplehj.htm");
report.setMREfolder("#carreportsa4");
report.setName("app/carmain.fex");

LogOnInfo logon = wfs.webQueryLogOn("","","RepUser","RepPass");
WebQueryReturn ret = wfs.webQueryRunFex(logon,report);

Links = ret.getLinks();
WebQueryReturn retdrill = wfs.webQueryLink(logon,Links[3]);

String newOutput=retdrill.getOutput();

File tempfile = new File("c:\\temp\\Centpl_actBud.htm");

FileOutputStream fos = new FileOutputStream(tempfile);

PrintWriter out=new PrintWriter(fos);
out.println(newOutput);
out.close();
    }
catch (Throwable t)
 {
 System.err.println(t);
 t.printStackTrace();
 System.exit(1);
 }
```

Example: Running Links Brought Back in a Web Query Graph in Visual Basic .NET

In the following example, the output of a Web Query report named CarGraph that produces a graph is written to the CarGraph.htm file in the c:\temp directory.

```
'The following line must be included before the Form Class
Imports System.IO

Dim wfs As New WQ.WebQuery
Dim logon As New WQ.LogOnInfo
Dim ret As New WQ.WebQueryReturn
Dim retdrill As New WQ.WebQueryReturn
Dim report As New WQ.FexInfo
Dim newOutput As String = ""
Dim tempfile As String
Dim lastlink As Integer = 0
Dim link As String
Dim outfile As String

report.MREdomain = "samplehj/samplehj.htm"
report.MREfolder = "#carreportsa4"
report.name = "app/cargraph.fex"
logon = wfs.WebQueryLogOn("","","RepUser","RepPass")
ret = wfs.WebQueryRunFex(logon, report)

link = ret.links(0).link
retdrill = wfs.WebQueryLink(logon, ret.links(0))
```

```
If retdrill.mime = "image/png" Then
     outfile = "c:\temp\CarGraph.png"
     Dim fs As FileStream = New FileStream(outfile,
FileMode.OpenOrCreate)
     Dim w As BinaryWriter = New BinaryWriter(fs)
w.Write(retdrill.binaryData, 0, retdrill.binaryData.Length)
fs.Close()

ElseIf retdrill.mime = "image/gif" Then
     outfile = "c:\temp\CarGraph.jpg"
     Dim fs As FileStream = New FileStream(outfile,
FileMode.OpenOrCreate)
     Dim w As BinaryWriter = New BinaryWriter(fs)
w.Write(retdrill.binaryData, 0, retdrill.binaryData.Length)
fs.Close()
     ElseIf retdrill.mime = "image/jpeg" Then
     outfile = "c:\temp\CarGraph.jpg"
     Dim fs As FileStream = New FileStream(outfile,
FileMode.OpenOrCreate)
     Dim w As BinaryWriter = New BinaryWriter(fs)
     w.Write(retdrill.binaryData, 0, retdrill.binaryData.Length)
     fs.Close()

ElseIf retdrill.mime = "image/svg+xml" Then
     outfile = "c:\temp\CarGraph.svg"
     Dim fs As FileStream = New FileStream(outfile,
FileMode.OpenOrCreate)
     Dim w As BinaryWriter = New BinaryWriter(fs)
     w.Write(retdrill.output, 0, retdrill.output.Length)
     fs.Close()
End If

newOutput = newOutput + Mid(ret.output, 1, ret.links(0).position)
newOutput = newOutput + outfile
lastlink = ret.links(0).position + link.Length
newOutput = newOutput + Mid(ret.output, lastlink + 1)
tempfile = "c:\temp\CarGraph.htm"

FileOpen(1, tempfile, OpenMode.Output)
Print(1, newOutput)
FileClose(1)
```

Note: WQ is the name of the Web Reference.

Example: Running Links Brought Back in a Web Query Graph in Java

In the following example, the output of a Web Query report named CarGraph that produces a graph is written to the CarGraph.htm file in the c:\temp directory.

```
try {
WebQuery WFservice = new WebQueryLocator();
WebQuerySoap_PortType wfs = WFservice.getWebQuerySoap();

File outfile = new File("c:\\temp\\CarGraph.png");
LinkArrayEntry[] Links;
FexInfo report = new FexInfo();
report.setMREdomain("samplehj/samplehj.htm");
report.setMREfolder("#carreportsa4");
report.setName("app/cargraph.fex");

LogOnInfo logon = wfs.webQueryLogOn("","","RepUser","RepPass");
WebQueryReturn ret = wfs.webQueryRunFex(logon,report);

Links = ret.getLinks();
String link = Links[0].getLink();
WebQueryReturn retdrill = wfs.webQueryLink(logon,Links[0]);

if (retdrill.getMime().equals ("image/png"))
{
   byte[] outbytes = retdrill.getBinaryData();
   outfile = new File("c:\\temp\\CarGraph.png");
   FileOutputStream fs = new FileOutputStream(outfile);
   fs.write(outbytes);
   fs.close();
}
```

```
else
{
   if (retdrill.getMime().equals ("image/gif"))
   {
      byte[] outbytes = retdrill.getBinaryData();
      outfile = new File("c:\\temp\\CarGraph.jpg");
      FileOutputStream fs = new FileOutputStream(outfile);
      fs.write(outbytes);
      fs.close();
   }
   else
   {
      if (retdrill.getMime().equals ("image/jpeg"))
      {
         byte[] outbytes = retdrill.getBinaryData();
         outfile = new File("c:\\temp\\CarGraph.jpg");
         FileOutputStream fs = new FileOutputStream(outfile);
         fs.write(outbytes);
         fs.close();
      }
      else
      {
         if (retdrill.getMime().equals ("image/svg+xml"))
         {
         String outstring = retdrill.getOutput();
         outfile = new File("c:\\temp\\CarGraph.svg");
         FileOutputStream fosg = new FileOutputStream(outfile);
         PrintWriter fs = new PrintWriter(fosg);
         fs.println(outstring);
         fs.close();
         }
         else
         {
         }
      }
   }
}
```

```
String newOutput = "";

newOutput=newOutput+ret.getOutput().substring(0, Links[0].getPosition() -

1);
newOutput=newOutput+outfile;
int lastlink = Links[0].getPosition()+ link.length();
newOutput=newOutput+ret.getOutput().substring(lastlink+1);

File tempfile = new File("c:\\temp\\CarGraph.htm");
FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);
out.println(newOutput);
out.close();
}
catch (Throwable t)
 {
 System.err.println(t);
 t.printStackTrace();
 System.exit(1);
 }
```

Passing a Drill-Down URL to Web Query

Function Name: WebQueryDrill

Purpose: To run a drill-down Web Query report when the URL is captured within a program if a link within a Web Query report is clicked on.

Input:

Description	**Type**
Web Query cookie information.	*LogOnInfo*
Edited link captured within a program when you have clicked a link in a Web Query report. You must remove the http:// and the machine name or TCP/IP address from the URL before you pass it. When you use Web Query, you must either associate the Web Query report with a specific application, or you must set SET BASEURL (for example, to SET BASEURL = 'http://localhost') in the Web Query report. This will only work if the output type of the Web Query report is HTML.	String

Output:

Description	Type
Structure that contains the output from a Web Query report.	*WebQueryReturn*

Example: Passing Links With Drill Down Information in Visual Basic .NET

In the following example, the output of a drill-down report from a Web Query report named CENTPL_ACTBUD is written to the Centpl_ActBud.htm file in the c:\temp directory. The example begins with the result URL that is clicked on.

```
Dim wfs As New WQ.WebQuery
Dim logon As New WQ.LogOnInfo
Dim ret As New WQ.WebQueryReturn
Dim retdrill As New WQ.WebQueryReturn
Dim report As New WQ.FexInfo
Dim newOutput As String = ""
Dim tempfile As String
Dim URL As String
Dim newURL As String

report.MREdomain = "samplehj/samplehj.htm"
report.MREfolder = "#carreportsa4"
report.name = "app/carmain.fex"

logon = wfs.WebQueryLogOn("","","RepUser","RepPass")
ret = wfs.WebQueryRunFex(logon, report)

URL = "http://localhost/webservice?IBIF_webapp=/webquery&IBIC_server=
EDASERVE &IBIWF_msgviewer=OFF&&IBIMR_drill=X,efremvhj/efremvhj.htm
&IBIF_ex=app/Cardrill.fex&CLICKED_ON=&COUNTRY=ENGLAND"

newURL = Replace(URL, "http://localhost", "")
retdrill = wfs.WebQueryDrill(logon, newURL)
newOutput = retdrill.output
tempfile = "c:\temp\Centpl_ActBud.htm"

FileOpen(1, tempfile, OpenMode.Output)
Print(1, newOutput)
FileClose(1)
```

Note: WQ is the name of the Web Reference.

Example: Passing Links With Drill Down Information in Java

In the following example, the output of a drill-down report from a Web Query report named CENTPL_ACTBUD is written to the Centpl_ActBud.htm file in the c:\temp directory. The example begins with the result URL that is clicked on.

```
try {
WebQuery WFservice = new WebQueryLocator();
WebQuerySoap_PortType wfs = WFservice.getWebQuerySoap();
String URL;
String newURL;

FexInfo report = new FexInfo();
report.setMREdomain("samplehj/samplehj.htm");
report.setMREfolder("#carreportsa4");
report.setName("app/carmain.fex");

LogOnInfo logon = wfs.webQueryLogOn("","","RepUser","RepPass");
WebQueryReturn ret = wfs.webQueryRunFex(logon,report);

URL = “http://localhost/webservice?IBIF_webapp=/webquery&IBIC_server=
EDASERVE&IBIWF_msgviewer=OFF&&IBIMR_drill=X,efremvhj/efremvhj.htm
&IBIF_ex=app/Cardrill.fex&CLICKED_ON=&COUNTRY=ENGLAND” ;

newURL = URL.substring(16);

WebQueryReturn retdrill = wfs.webQueryDrill(logon,newURL);

String newOutput=retdrill.getOutput();

File tempfile = new File("c:\\temp\\Centpl_actBud.htm");

FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);
out.println(newOutput);
out.close();
    }

catch (Throwable t)
 {
 System.err.println(t);
 t.printStackTrace();
 System.exit(1);
 }
```

Listing Values for a Column

Function Name: WebQueryFieldValues

Purpose: To retrieve a list of values for a particular column within a table.

Input:

Description	**Type**
Web Query cookie information.	*LogOnInfo*
Table name.	String
Web Query code that is used as a preprocess before the values are retrieved. This value must be set to null.	String
Column name or virtual field name.	String
Requested format of returned values. This value must be set to null.	String
Selection criteria. This value must be set to null.	String

Output:

Description	**Type**
Structure that contains output for Web Query reports and certain Web Query functions. The list of values is returned to the values array of WebQueryReturn. values(0) contains an array of values.	*WebQueryReturn*

Example: Listing Values for a Column in Visual Basic .NET

In the following example, the list of values for the column COUNTRY within the CAR table is written to the Values.txt file in the c:\temp directory. COUNTRYN has been defined as the COUNTRY column name surrounded by brackets. The values have been reformatted to A15 from the original size of A12. Only values where the Dealer Cost is greater than 5000 will be selected.

```
Dim wfs As New WQ.WebQuery
Dim logon As New WQ.LogOnInfo
Dim ret As New WQ.WebQueryReturn
Dim newOutput As String = ""
Dim tempfile As String
Dim I As Integer
Dim FocCode As String
Dim FC1 As String
Dim FC2 As String
Dim FC3 As String
Dim CRLF As String
Dim Selct As String


CRLF = vbCrLf
FC1 = "DEFINE FILE CAR"
FC2 = "COUNTRYN/A12 = '(' | COUNTRY | ')';"
FC3 = "END"

FocCode = FC1 + CRLF + FC2 + CRLF + FC3

Selct = "IF DEALER_COST GT 5000"

logon = wfs.WebQueryLogOn("","","RepUser","RepPass")
ret = wfs.WebQueryFieldValues(logon, "CAR", FocCode, "COUNTRYN", "A15",
Selct)

tempfile = "c:\temp\Values.txt"

FileOpen(1, tempfile, OpenMode.Output)

For I = 0 To ret.values(0).Length - 1
       newOutput = ret.values(0)(I)
        PrintLine(1, newOutput)

Next I

FileClose(1)
```

Note: WQ is the name of the Web Reference.

Example: Listing Values for a Column in Java

In the following example, the list of values for the column COUNTRY within the CAR table is written to the Values.txt file in the c:\temp directory. COUNTRYN has been defined as the COUNTRY column name surrounded by brackets. The values have been reformatted to A15 from the original size of A12. Only values where the Dealer Cost is greater than 5000 will be selected.

```
try {
WebQuery WFservice = new WebQueryLocator();
WebQuerySoap_PortType wfs = WFservice.getWebQuerySoap();

WebQueryReturn ret;
String[][] StringArray;

String CRLF = System.getProperty("line.separator");
String FC1 = "DEFINE FILE CAR";
String FC2 = "COUNTRYN/A12 = '(' | COUNTRY | ')';";
String FC3 = "END";

String FocCode = FC1 + CRLF + FC2 + CRLF + FC3;

String Selct = "IF DEALER_COST GT 5000";

LogOnInfo logon = wfs.webQueryLogOn("","","RepUser","RepPass");
ret = wfs.webQueryFieldValues(logon,"CAR",FocCode,"COUNTRYN","A15",Selct);


String newOutput = null;
File tempfile = new File("c:\\temp\\Values.txt");
FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);

StringArray = ret.getValues();

for ( int I=0; i<StringArray[0].length; I++ )
    {
      newOutput = StringArray[0][I];
      out.println(newOutput);
    }
out.close();
    }
catch (Throwable t)
 {
 System.err.println(t);
 t.printStackTrace();
 System.exit(1);
 }
```

Getting a List of Domains for a Particular User

Function Name: MREGetUserDomains

Purpose: To retrieve a list of domains that the user can access.

Input:

Description	Type
Web Query cookie information.	*LogOnInfo*

Output:

Description	Type
Structure that contains XML for various Web Query functions.	*MREReturn*

Example: **Getting a List of Domains for a Particular User in Visual Basic .NET**

In the following example, a list of domains that the user can access is retrieved and written to the GetUserDomains.xml file in the c:\temp directory.

```
Dim wfs As New MR.WebQuery
Dim logon As New MR.LogOnInfo
Dim retMR As New MR.MREReturn
Dim newOutput As String = ""
Dim tempfile As String

logon = wfs.WebQueryLogOn("","","RepUser","RepPass")
retMR = wfs.MREGetUserDomains(logon)

tempfile = "c:\temp\GetUserDomains.xml"

newOutput = retMR.xml

FileOpen(1, tempfile, OpenMode.Output)
Print(1, newOutput)
FileClose(1)
```

Note: MR is the name of the Web Reference.

Example: Getting a List of Domains for a Particular User in Java

In the following example, a list of domains that the user can access is retrieved and written to the GetUserDomains.xml file in the c:\temp directory.

```
try {
WebQuery WFservice = new WebQueryLocator();
WebQuerySoap_PortType wfs = WFservice.getWebQuerySoap();

LogOnInfo logon =
wfs.webQueryLogOn("","","RepUser","RepPass");
MREReturn retMR = wfs.MREGetUserDomains(logon);

String newOutput = retMR.getXml();

File tempfile = new File("c:\\temp\\GetUserDomains.xml");

FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);
out.println(newOutput);
out.close();
    }
catch (Throwable t)
 {
 System.err.println(t);
 t.printStackTrace();
 System.exit(1);
 }
```

Opening a Domain

Function Name: MREOpenDomain

Purpose: To open a domain and obtain a list of all Web Query reports within Standard Reports, Reporting Objects, and Other Files folders.

Input:

Description	Type
Web Query cookie information.	*LogOnInfo*
HREF of the domain. You can obtain the HREF in the output of MREGetUserDomains.	String

Output:

Description	Type
Structure that contains XML for various Web Query functions.	*MREReturn*

Example: Opening a Domain in Visual Basic .NET

In the following example, a domain named "New Domain" is opened. The XML output from the function is written to the OpenDomain.xml file in the c:\temp directory.

- The HREF for "New Domain" is obtained from the output of MREGetUserDomains:

```
<HREF flgs="none" href="xgzr36o2/xgzr36o2.htm" desc="New Domain"
imag="" />

Dim wfs As New MR.WebQuery
Dim logon As New MR.LogOnInfo
Dim retMR As New MR.MREReturn
Dim newOutput As String
Dim tempfile As String

logon = wfs.WebQueryLogOn("","","RepUser","RepPass")

retMR = wfs.MREOpenDomain(logon, "xgzr36o2/xgzr36o2.htm")

tempfile = "c:\temp\OpenDomain.xml"

newOutput = retMR.xml

FileOpen(1, tempfile, OpenMode.Output)
Print(1, newOutput)
FileClose(1)
```

Note: MR is the name of the Web Reference.

Example: Opening a Domain in Java

In the following example, a domain named "New Domain" is opened. The XML output from the function is written to the OpenDomain.xml file in the c:\temp directory.

- The HREF for "New Domain" is obtained from the output of MREGetUserDomains:

```
<HREF flgs="none" href="xgzr36o2/xgzr36o2.htm" desc="New Domain"
imag="" />

try {
WebQuery WFservice = new WebQueryLocator();
WebQuerySoap_PortType wfs = WFservice.getWebQuerySoap();

LogOnInfo logon=
wfs.webQueryLogOn("","","RepUser","RepPass");

MREReturn retMR = wfs.MREOpenDomain(logon,"xgzr36o2/xgzr36o2.htm");

String newOutput = retMR.getXml();

File tempfile = new File("c:\\temp\\OpenDomain.xml");

FileOutputStream fos = new
FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);
out.println(newOutput);
out.close();
    }
catch (Throwable t)
 {
 System.err.println(t);
 t.printStackTrace();
 System.exit(1);
 }
```

Example: Viewing XML Output Generated By MREOpenDomain

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <ibwfrpc name="MR_OPEN_DOMAIN">
  <RETURN_TYPE_VERSION>1</RETURN_TYPE_VERSION>
  <DOMAIN_COUNT>33</DOMAIN_COUNT>
  <MR_HELP flgs="none" href="app/help.htm" size="195" time="1067353378"
desc="Help" />
- <MR_OTHER_SECTION>
  <MR_HREF_PROPERTIES flgs="ddmap" href="app/webservi.kmd" size="42"
time="1072795840" desc="Web Services Domain" imag="" />
  </MR_OTHER_SECTION>
- <MR_STD_REPORT>
- <MR_STD_REPORT_FOLDER flgs="none" href="#newfoldersrf"
name="newfoldersrf" desc="HR Reports" imag="">
  <MR_HREF_PROPERTIES href="app/centempt.htm" size="1957"
time="1089656067" desc="centempt" imag="" />
  <MR_HREF_PROPERTIES href="app/hrranknm.fex" size="5151"
time="1090003822" desc="hrranknM" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/ranking.fex" size="4511"
time="1089655726" desc="Ranking" imag="" />
  </MR_STD_REPORT_FOLDER>
- <MR_STD_REPORT_FOLDER flgs="none" href="#soapadapterk"
name="soapadapterk" desc="SOAP Adapter" imag="">
  <MR_HREF_PROPERTIES flgs="none" href="app/adhoc.fex" size="385"
time="1082137354" desc="Adhoc" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/baseball.fex" size="2140"
time="1089807365" desc="BaseballRoster" imag="" />
  <MR_HREF_PROPERTIES href="app/gethoros.fex" size="1406"
time="1072885852" desc="GetHoroscope" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/getplaye.fex" size="1454"
time="1076538938" desc="GetPlayers" imag="" />
```

```
  <MR_HREF_PROPERTIES href="app/nflnews.fex" size="1439"
time="1072795646" desc="NFLNews" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/testheb.fex" size="427"
time="1091544691" desc="TestHeb" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/webquery.fex" size="593"
time="1091460016" desc="WebQueryDBInfo" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/g665d93p.fex" size="641"
time="1091132635" desc="WebQueryFieldValues" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/zl20xysv.fex" size="694"
time="1091460397" desc="WebQueryGetFexText" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/o0xasxff.fex" size="737"
time="1091120255" desc="WebQueryListApps" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/r6wao1xy.fex" size="834"
time="1091468774" desc="WebQueryListFexs" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/tzzw715l.fex" size="582"
time="1091132843" desc="WebQueryListServers" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/aw0165u3.fex" size="755"
time="1091469080" desc="WebQueryMasterInfo" imag="" />
  <MR_HREF_PROPERTIES flgs="none" href="app/localtim.fex" size="2199"
time="1091117884" desc="Zip Code Information" imag="" />
  </MR_STD_REPORT_FOLDER>
  </MR_STD_REPORT>
- <MR_REPORT_OBJECT>
- <MR_REPORT_OBJECT_FOLDER flgs="none" href="#zipcodejoins"
name="zipcodejoins" desc="Zip Code Joins" imag="">
```

```
<MR_HREF_PROPERTIES flgs="suffix" href="mrv/localtim.fex" size="318"
time="1093439281" desc="Local Time to Temperature" imag="" />
  </MR_REPORT_OBJECT_FOLDER>
  </MR_REPORT_OBJECT>
  <MR_FILE_INFO href="webservi/webservi.htm" size="2782" type="DOMAIN"
look="0" time="1093439281" />
  <MR_GET_USERS
flgs="admin,shared,rcadmin,robot,library,email=efrem_litwin@ibi.com"
href="admin.htm" name="admin" desc="Default Administrator" imag="" />
- <IBIMR_MYREPORTS_TABLE flgs="none" href="webservi/webservi.htm"
desc="Web Services">
- <IBIMR_MYREPORTS_TABLE_FOLDER flgs="none" href="#.olapcustomreports"
desc="Custom Reports" imag="">
  <MR_HREF_PROPERTIES flgs=",myreport,rassist,shared"
href="admin/haveaquo.fex" size="474" time="1092921850" desc="Have a
Quote" imag="" />
  <MR_HREF_PROPERTIES flgs="shared" href="admin/testrepo.fex" size="45"
time="1093384265" desc="Test Report" imag="" />
  </IBIMR_MYREPORTS_TABLE_FOLDER>
- <IBIMR_MYREPORTS_TABLE_FOLDER href="#.olapdeferred" desc="Deferred
Reports" imag="">
  <MR_HREF_PROPERTIES flgs="1093438271,fexinfo=IBIMR_domain
    %3Dwebservi%2Fwebservi.htm%2CIBIMR_folder%3D
    %23soapadapterk%2CIBIMR_fex%3Dapp%2Fgetplaye.fex
    %2CIBIMR_sub_action%3DMR_STD_REPORT%2CIBIC_user%3D
    %2CIBIMR_time%3D1076538938288%2C%2CIBIMR_report_type
    %3D%2CIBIMR_checkboxcount%3D0%2C,"
    href="#2004-08-25-08-30-24cmrpip000012_edaserve" size="0" time="0"
    desc="GetPlayers" imag="" />

<MR_HREF_PROPERTIES flgs="1093017713,fexinfo=IBIMR_domain
    %3Dwebservi%2Fwebservi.htm%2CIBIMR_folder
    %3D%23soapadapterk%2CIBIMR_fex%3Dapp%2Fzoh9uv5k.fex
    %2CIBIMR_sub_action%3DMR_STD_REPORT%2CIBIC_user%3D
    %2CIBIMR_time%3D1088775483508%2C%2CIBIMR_report_type
    %3D%2CIBIMR_checkboxcount%3D0%2CTICKER%3DSIRI%2C,"
    href="#2004-08-20-10-01-08cmrpip000017_edaserve" size="0" time="0"
    desc="Get Stock Quotes" imag="" />
  </IBIMR_MYREPORTS_TABLE_FOLDER>
  </IBIMR_MYREPORTS_TABLE>
  <MR_FILE_INFO
flgs="admin,shared,rcadmin,robot,library,email=efrem_litwin@ibi.com"
href="admin.htm" size="4467" type="USER_ID" look="0" time="1093438271" />
  <RETURNCODE>1000</RETURNCODE>
  </ibwfrpc>
```

Report Broker Functions

In this section:

This topic describes the Report Broker functions contained within Web Query Web Services.

Logging on to Report Broker

Function Name: logon

Purpose: Accepts a user name and a password. It returns a string representing a security token. This token is then set in a corresponding Authenticate object.

Input:

Description	Type
User Name.	String
Password.	String

Output:

Description	Type
Represents the security token associated with this logon.	String

Example: Logging on to Report Broker in Visual Basic .NET

In the following example, a user is logged on to Report Broker and the security token is set. The security token is then used as part of the Authenticate structure. The Authenticate structure is the first parameter to all other Report Broker Web Services functions. The security token is written to the RCsecToken.txt file.

```
Try
  Dim RCLogon As New LogonManager.LogonManagerWSService
  Dim SecToken As String
  Dim tempfile As String
  Dim newOutput As String

  SecToken = RCLogon.logon("admin", "")

  tempfile = "d:\RCtemp\RCsecToken.txt"
  FileOpen(1, tempfile, OpenMode.Output)

  newOutput = SecToken

  PrintLine(1, newOutput)
  FileClose(1)

  Catch x As Exception

     MsgBox(x.Message, MsgBoxStyle.OKOnly, "Error Message")

  End Try
```

Example: Logging on to Report Broker in Java

In the following example, a user is logged on to Report Broker and the security token is set. The security token is then used as part of the Authenticate structure. The Authenticate structure is the first parameter to all other Report Broker Web Services functions. The security token is written to the RCsecToken.txt file.

```
try
{
LogonManagerWSService LogonService = new LogonManagerWSServiceLocator();
LogonManagerWS RCLogon = LogonService.getLogonService();
String sectoken = RCLogon.logon("admin","");
File tempfile = new File("d:\\RCtemp\\RCsecToken.txt");
FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter (fos);
String newOutput = sectoken;
out.println(newOutput);
out.close();
}
catch (Throwable t)
{
  System.err.println(t);
  t.printStackTrace();
  System.exit(1);
}
```

Retrieving an Existing Schedule From the Report Broker Repository

Function Name: getSchedule

Purpose: Retrieves an existing schedule from the Report Broker repository based on a given schedule identifier. The schedule ID uniquely identifies a schedule and can be used to retrieve all schedule data including any information pertaining to tasks associated with the schedule. This function is available to the administrator and the schedule owner only.

Input

Description	Type
Authentication information.	Authenticate
ID that uniquely identifies the schedule in the repository.	String

Output

Description	Type
Schedule information.	Schedule

Example: Retrieving the Existing Schedule From the Report Broker Repository in Visual Basic .NET

In the following example, schedule information for a specific schedule ID is retrieved. In this example, the schedule retrieved schedules a Standard Report from Managed Reporting to be distributed to the Report Broker Library and it is to be run once. The schedule description, distribution type (for example, DistributionEmail/DistributionPrint/StorageLibrary), Storage Library Category, Time Type (for example, TimeInfoOnce/TimeInfoDay/TimeInfoWeek), Start Time, Task Type (for example, TaskWQServerProcedure/ TaskStandardReport), procedure name, and the report parameters are written to the RCrunSchedule.txt file.

```
Try
   Dim RCLogon As New LogonManager.LogonManagerWSService
   Dim S As New SManager.ScheduleManagerWSService
   Dim SecToken As String
   Dim SAuthenticate As New SManager.Authenticate
   Dim mySchedule As New SManager.Schedule
   Dim tempfile As String
   Dim newOutput As String
   Dim DistributionType As System.Type
   Dim TimeType As System.Type
   Dim TaskType As System.Type
   Dim i As Integer

   SecToken = RCLogon.logon("admin", "")
   SAuthenticate.securityToken = SecToken

   mySchedule = S.getSchedule(SAuthenticate, "S10259bneq04")

   DistributionType = mySchedule.distribution.GetType()
   TimeType = mySchedule.timeInfo.GetType
   TaskType = mySchedule.taskList(0).GetType

   Dim SLdistribution As New SManager.StorageLibrary
   SLdistribution = mySchedule.distribution

   Dim RunOnce As New SManager.TimeInfoOnce
   RunOnce = mySchedule.timeInfo

   Dim MREtask As New SManager.TaskStandardReport
   MREtask = mySchedule.taskList(0)

   tempfile = "d:\RCtemp\RCrunSchedule.txt"
   FileOpen(1, tempfile, OpenMode.Output)

   newOutput = mySchedule.description + "   " + _
               DistributionType.Name + " " + _
               SLdistribution.category + "   " + _
               TimeType.Name + "   " + _
               RunOnce.startTime + " " + _
               TaskType.Name + "   " + _
               MREtask.procedureName
   PrintLine(1, newOutput)

   For i = 0 To MREtask.parameterList.Length - 1
         newOutput = MREtask.parameterList(i).name + "  " + _
                     MREtask.parameterList(i).value
         PrintLine(1, newOutput)
   Next i

   FileClose(1)



   Catch x As Exception

       MsgBox(x.Message, MsgBoxStyle.OKOnly, "Error Message")

End Try
```

Example: Retrieving the Existing Schedule From the Report Broker Repository in Java

In the following example, schedule information for a specific schedule ID is retrieved. In this example, the schedule retrieved schedules a Standard Report from Managed Reporting to be distributed to the Report Broker Library and it is to be run once. The schedule description, distribution type (for example, DistributionEmail/DistributionPrint/StorageLibrary), Storage Library Category, Time Type (for example, TimeInfoOnce/TimeInfoDay/TimeInfoWeek), Start Time, Task Type (for example, TaskWQServerProcedure/ TaskStandardReport), procedure name, and the report parameters are written to the RCrunSchedule.txt file.

```
try
{
LogonManagerWSService LogonService = new LogonManagerWSServiceLocator();
LogonManagerWS RCLogon = LogonService.getLogonService();

ScheduleManagerWSService ScheduleService = new ScheduleManagerWSServiceLocator();
ScheduleManagerWS S = ScheduleService.getScheduleService();

String sectoken = RCLogon.logon("admin","");
Authenticate Authobj = new Authenticate();
Authobj.setSecurityToken(sectoken);

Schedule mySchedule = S.getSchedule(Authobj,"S10259bneq04");

String distributionType =
mySchedule.getDistribution().getClass().getName();
StorageLibrary SLdistribution =
(StorageLibrary)mySchedule.getDistribution();

String timeType = mySchedule.getTimeInfo().getClass().getName();
TimeInfoOnce runOnce = (TimeInfoOnce)mySchedule.getTimeInfo();

String taskType = mySchedule.getTaskList()[0].getClass().getName();
TaskStandardReport MREtask = (TaskStandardReport)mySchedule.getTaskList()[0];

File tempfile = new File("d:\\RCtemp\\RCrunSchedule.txt");
FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);
SimpleDateFormat sdf = new SimpleDateFormat();

String newOutput = mySchedule.getDescription() + "   " +
                   distributionType + "   " +
                   SLdistribution.getCategory()+ "   " +
                   timeType + "   " +
                   sdf.format(runOnce.getStartTime().getTime()) + "   " +
                   taskType + "   " +
                   MREtask.getProcedureName();

out.println(newOutput);
for ( int i=0; i<MREtask.getParameterList().length; i++ )
{
         newOutput = MREtask.getParameterList()[i].getName() + "  " +
                     MREtask.getParameterList()[i].getValue();

         out.println(newOutput);
}

out.close();

}

catch (Throwable t)
{
  System.err.println(t);
  t.printStackTrace();
  System.exit(1);
}
```

Retrieving a List of Schedule Information From the Report Broker Repository

Function Name: getScheduleInfoListByCaller

Purpose: Retrieves a list of schedule information from the Report Broker repository that is owned by the current logon user. This is designed just to retrieve information for each schedule for performance reasons. Schedule information has all its properties populated with data with the exception of the task information, which are all NULL. An administrator will receive all schedules, and an end user will receive only those schedules owned by him.

Input

Description	Type
Authentication information.	Authenticate

Output

Description	Type
List of schedule information.	Schedule

Example: Retrieving a List of Schedules in Visual Basic .NET

In the following example, a list of schedules containing schedule Information for the authenticated user is retrieved. The task information is not returned in this function. The getSchedule function must be used to return schedule information including the task information. The schedule ID and schedule description are written to RCscheduleListByCaller.txt file.

```
Try
   Dim RCLogon As New LogonManager.LogonManagerWSService
   Dim S As New SManager.ScheduleManagerWSService
   Dim SecToken As String
   Dim SAuthenticate As New SManager.Authenticate
   Dim ScheduleInfo() As SManager.Schedule
   Dim tempfile As String
   Dim newOutput As String
   Dim i As Integer

   SecToken = RCLogon.logon("admin", "")
   SAuthenticate.securityToken = SecToken

   ScheduleInfo = S.getScheduleInfoListByCaller(SAuthenticate)

   tempfile = "d:\RCtemp\RCscheduleListByCaller.txt"
   FileOpen(1, tempfile, OpenMode.Output)

   For i = 0 To ScheduleInfo.Length - 1
         newOutput = ScheduleInfo(i).id + "  " + _
                     ScheduleInfo(i).description

         PrintLine(1, newOutput)
   Next i

   FileClose(1)
   Catch x As Exception

       MsgBox(x.Message, MsgBoxStyle.OKOnly, "Error Message")

End Try
```

Example: Retrieving a List of Schedules in Java

In the following example, a list of schedules containing schedule Information for the authenticated user is retrieved. The task information is not returned in this function. The getSchedule function must be used to return schedule information including the task information. The schedule ID and schedule description are written to RCscheduleListByCaller.txt file.

```
try
{
LogonManagerWSService LogonService = new LogonManagerWSServiceLocator();
LogonManagerWS RCLogon = LogonService.getLogonService();

ScheduleManagerWSService ScheduleService = new
ScheduleManagerWSServiceLocator();
ScheduleManagerWS S = ScheduleService.getScheduleService();

String sectoken = RCLogon.logon("admin","");
Authenticate Authobj = new Authenticate();
Authobj.setSecurityToken(sectoken);

Schedule scheduleInfo[] = S.getScheduleInfoListByCaller(Authobj);

File tempfile = new File("d:\\RCtemp\\RCscheduleListByCaller.txt");
FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);

for ( int i=0; i<scheduleInfo.length; i++ )
{
         String newOutput = scheduleInfo[i].getId() + "   " +
                            scheduleInfo[i].getDescription();

         out.println(newOutput);
}

out.close();

}

catch (Throwable t)
{
  System.err.println(t);
  t.printStackTrace();
  System.exit(1);
}
```

Running the Schedule

Function Name: run

Purpose: After schedule information has been created and its properties populated with data, this function immediately submits the schedule to the Report Broker Distribution Server that runs the schedule. When the schedule runs, the Distribution Server creates a unique job number or job ID for this schedule, which is returned to the caller upon the completion/execution of the job. This function is available to the administrator and the schedule owner only.

Input

Description	**Type**
Authentication information.	Authenticate
Schedule information encapsulating the job that is to be run.	Schedule

Output

Description	**Type**
A job number generated by the Distribution Server.	String

Example: Running the Schedule in Visual Basic .NET

In the following example, a schedule is run. The getSchedule function is run first to retrieve all the schedule information for a specified schedule. The schedule information is used as input to the "run" function. The job ID is written to RCrunSchedule.txt file.

```
Try
   Dim RCLogon As New LogonManager.LogonManagerWSService
   Dim S As New SManager.ScheduleManagerWSService
   Dim SecToken As String
   Dim SAuthenticate As New SManager.Authenticate
   Dim scheduleInfo As New SManager.Schedule
   Dim jobId As String
   Dim tempfile As String
   Dim newOutput As String

   SecToken = RCLogon.logon("admin", "")
   SAuthenticate.securityToken = SecToken

   scheduleInfo = S.getSchedule(SAuthenticate, "S10259bneq04")

   jobId = S.run(SAuthenticate, scheduleInfo)

   tempfile = "d:\RCtemp\RCrunSchedule.txt"
   FileOpen(1, tempfile, OpenMode.Output)

   newOutput = jobId

   PrintLine(1, newOutput)
   FileClose(1)
   Catch x As Exception

       MsgBox(x.Message, MsgBoxStyle.OKOnly, "Error Message")

End Try
```

Example: Running the Schedule in Java

In the following example, a schedule is run. The getSchedule function is run first to retrieve all the schedule information for a specified schedule. The schedule information is used as input to the "run" function. The job ID is written to RCrunSchedule.txt file.

```
try
{
LogonManagerWSService LogonService = new LogonManagerWSServiceLocator();
LogonManagerWS RCLogon = LogonService.getLogonService();

ScheduleManagerWSService ScheduleService = new
ScheduleManagerWSServiceLocator();
ScheduleManagerWS S = ScheduleService.getScheduleService();

String sectoken = RCLogon.logon("admin","");
Authenticate Authobj = new Authenticate();
Authobj.setSecurityToken(sectoken);

Schedule scheduleInfo = S.getSchedule(Authobj,"S10259bneq04");

String jobId = S.run(Authobj,scheduleInfo);

File tempfile = new File("d:\\RCtemp\\RCrunSchedule.txt");
FileOutputStream fos = new FileOutputStream(tempfile);
PrintWriter out=new PrintWriter(fos);

String newOutput = jobId;

out.println(newOutput);

out.close();

}

catch (Throwable t)
{
  System.err.println(t);
  t.printStackTrace();
  System.exit(1);
}
```

IBM

5 Troubleshooting Web Query Web Services

This topic provides information about troubleshooting Web Services.

Topics:

- ❑ Troubleshooting Steps

Troubleshooting Steps

This topic describes the steps you must perform to aid the debugging process of an application that calls Web Query Web Services:

1. **Run Web Services Traces.**

   When running an application that calls Web Query Web Services, the UDDI Trace facility can be invoked to help debug a program. Some of the reasons for this type of tracing include the abending of a program when the Web Query Web Services function is being executed or the output of the Web Services function call is not the expected result.

   a. To access Web Services tracing, log on to the Web Query Administration Console using the following URL:

      `http://target_machine[:port]/webquery_html/wfconsole.htm`

      where:

      `target_machine`

      Is the location where Web Query is installed.

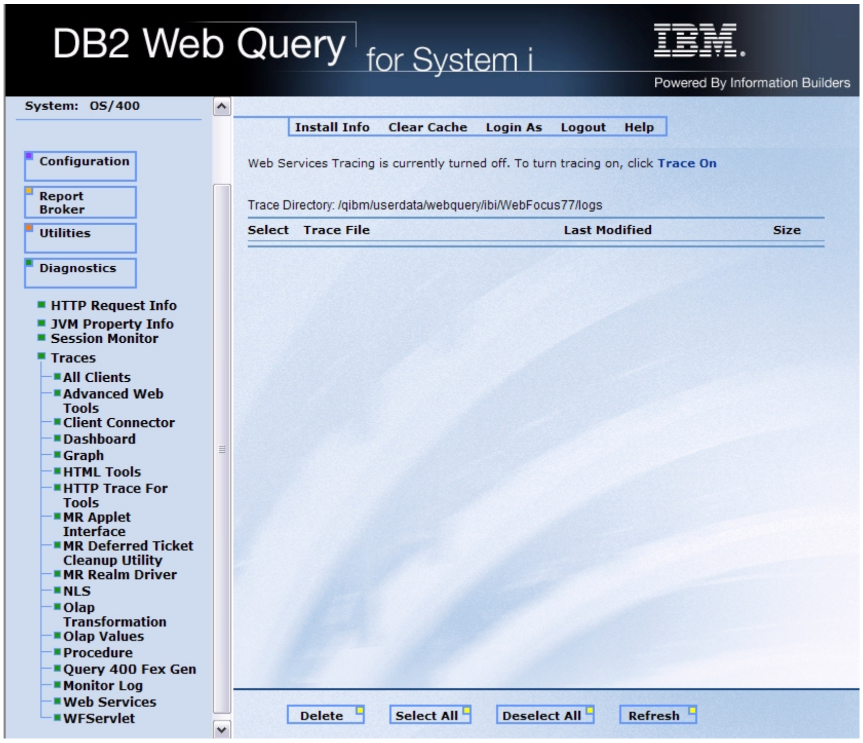
      `port`

      Is the port number used by Web Query.

   b. Click the *diagnostics* button.

   c. Click *Web Services* under the Traces section.

d. To turn on Web Services tracing, click *Trace On* in the right panel.



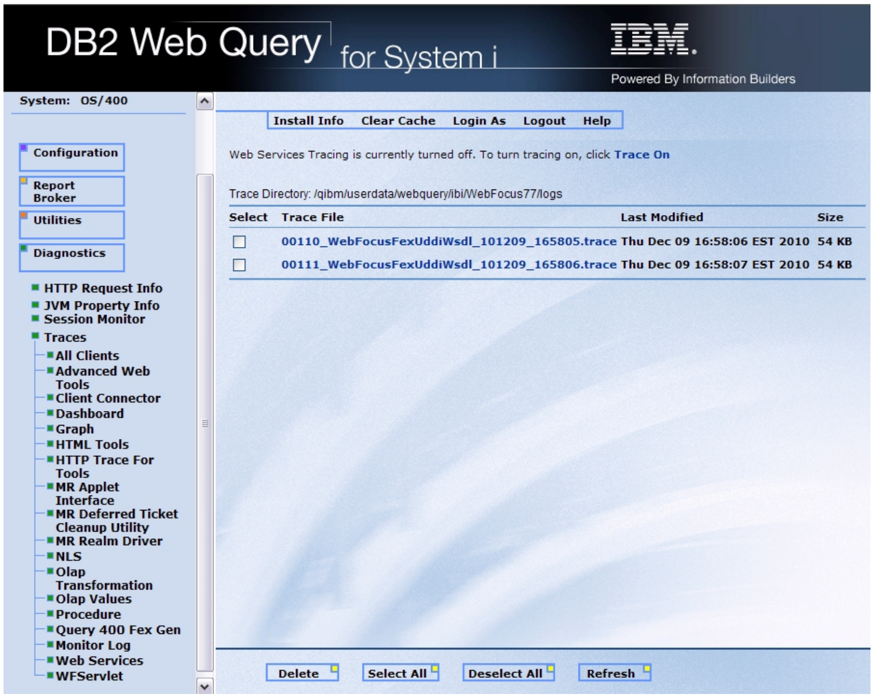
Web Services Tracing is currently turned off. To turn tracing on, click Trace On

Trace Directory: /qibm/userdata/webquery/ibi/WebFocus77/logs

Select	Trace File	Last Modified	Size

e. Run your application.

After your application has either abended or completed executing, go back to the Tracing page and click the *Refresh* button. Note that there is a trace file for each Web Query Web Services function call. To view a trace file, click on the trace file name.



Web Services Tracing is currently turned off. To turn tracing on, click Trace On

Trace Directory: /qibm/userdata/webquery/ibi/WebFocus77/logs

Select	Trace File	Last Modified	Size
☐	00110_WebFocusFexUddiWsdl_101209_165805.trace	Thu Dec 09 16:58:06 EST 2010	54 KB
☐	00111_WebFocusFexUddiWsdl_101209_165806.trace	Thu Dec 09 16:58:07 EST 2010	54 KB

2. To turn off Web Services traces, click *Trace Off* in the right panel.

3. Check authentication.

When you run the WebQueryLogOn function, the authentication status has to be equal to true. You can determine this by interrogating the status in the LogOnInfo structure after this function is run or by looking in the trace file to see whether the authentication was successful.

The following would be the lines in the trace file if the authentication was successful:

```
57:719:|
-- mre rc1000
58:729:
after mre signedOn = true
59:769:
after wf signedOn = true
```

The following would be the lines in the trace file if the authentication was unsuccessful:

```
57:811:
-- mre rc1005
58:821:
after mre signedOn = false
59:831:
after wf signedOn = false
```

4. **Ensure that parameters are passed correctly.**

   Verify with the documentation that the Web Services function is being called correctly.

   When setting the ValuesArrayEntry structure to pass parameters to a Web Query report using the WebQueryRunFex function, ensure that the Name and Value (or StringArray) are set properly for each parameter. Use the WebQueryFexReflection function for determining the current parameters of the Web Query report.

IBM

Index