ImageNow Envoy

Getting Started Guide

ImageNow Version: 6.7.x

perceptivesoftware

Written by: Product Documentation, R&D
Date: September 2016

Table of Contents

Getting Started with ImageNow Envoy

This document introduces ImageNow Envoy Agent provides an overview of its requirements and web services. Also included is a high-level summary of the steps to follow in writing your own application using ImageNow Envoy. This document is written for developers. For an overview on ImageNow that is geared towards developers, refer to the *ImageNow Overview for Developers* document. This document, as well as other developer content, is available on the Customer Portal of the Perceptive Software website at www.perceptivesoftware.com.

What is ImageNow Envoy?

ImageNow Envoy provides capabilities you can use to facilitate automated, back-end integration of ImageNow with your business applications. Through Envoy, you can send unsolicited, web service-based requests to a remote system with minimal programming. With this capability, you can further automate your business processes. You can also use Envoy to enable ImageNow to make remote service operations available to ImageNow users.

You can use Envoy to automatically create or update records in a business application, such as PeopleSoft, SAP, in-house software, or a custom-built software package. For example, suppose your employees import expense receipts into ImageNow. With Envoy, you can extract this data from the ImageNow documents and then automatically deliver the data to your company's time and expense business application through remote services. This automation ensures that your system properly reflects the delivery data of the documents. If you couple this enhanced automation with a tight integration to ImageNow, you realize increased return on investment as well as add the ability to further extend document management for your existing Service-Oriented Architecture (SOA) framework.

You use an iScript object to invoke your configured web service adaptors. You can launch scripts from within a workflow queue or a form as well as through INTool, and anywhere you use iScript in ImageNow products. You must have an installed Envoy license and an iScript feature license to use this product. Also, using the Management Console, you can add and delete remote services, and activate or deactivate specific remote service operations.

The functions and methods provided by the Envoy object, INRemoteService, contain the abilities to:

- insert, update, or delete a remote service adaptor
- return all of the operations of a remote service
- activate and de-activate operations to designate them as available or not to ImageNow users
- or make calls to remote services

Additionally, the ImageNow database stores the remote service configurations and properties.

In common usage, the term remote service refers to clients and servers that communicate using XML messages that follow the SOAP standard. In this type of communication, there is often a machine-readable description of the operations offered by the service written in the Web Services Description Language (WSDL). For help using the Envoy object, refer to the INRemoteService topic in the iScript help.

Add an Envoy service

You can complete this procedure only if you are a user with management privileges, a manager, or the owner.

An important note about Envoy capabilities involves the transcoders which are supported. For example, if you attempt to add a remote service from a site that uses an 'ISO-8859-1 transcoder, Envoy will return an exception. At this time, only the following transcoders are supported:

- `<tcdr:transcoder tcdr:class="WASP_PlainTextTranscoder"/>`
- `<tcdr:transcoder tcdr:class="WASP_UTF8Transcoder" pt:name="UTF-8" xmlns:pt="urn:UTF-8"/>`
- `<tcdr:transcoder tcdr:class="WASP_UTF16Transcoder" pt:name="UTF-16LE" pt:littleEndian="1" xmlns:pt="urn:UTF-16"/>`
- `<tcdr:transcoder tcdr:class="WASP_UTF16Transcoder" pt:name="UTF-16BE" pt:littleEndian="0" xmlns:pt="urn:UTF-16"/>`

Follow the steps below to add a remote service to Envoy:

1. In the **Management Console**, in the left pane, click **Envoy Services**.
2. On the **Envoy** tab, in the right pane, click **New**.
3. In the **Definition** page, fill out the following options.
    - In the **Name** box, type a name to identify the remote service throughout ImageNow.
    - Optional. In the **Description** box, type a description for the remote service.
    - In the **URI** box, provide the actual URI you want to use for the remote service.

    Note In computing, a Uniform Resource Identifier (URI) consists of a string of characters used to identify or name a resource on the Internet. You might classify a URI as a locator (URL), or a name (URN), or both.

    - In the **Authentication** list, select one of the following options:
        - **None** If you do not require users to authenticate before using the remote service.
        - **SSL** If you require users to authenticate from a client to a server using SSL certificates. Note that this authentication method is clientside SSL, not serverside.
        - **HTTP Basic** If you require users to authenticate using a user name and password using an HTTP request.
        - **WS-Security** If you require users to authenticate using attached signatures and encryption headers to SOAP messages.
        - Select the **Enable interceptor logging** check box if you want to enable interceptor logging for the remote service.

        Note The logging files produced by the Interceptor are useful for troubleshooting issues with SOAP requests and responses.
4. Click **Next**.

5. If you selected **SSL**, in the **SSL** page, perform the following actions:
    1. Under **SSL Certificate**, click the ellipsis … button, browse to the folder that contains your certificate file, and then double-click the filename in the **Open** dialog box. In the **Password**box, type the password.
    2. Under **Private Key**, click the ellipsis … button, browse to the folder that contains your private key file, and then double-click the filename in the **Open** dialog box. In the **Password** box, type the password.
    3. Click **Next**.
6. If you selected **HTTP Basic**, in the **HTTP Basic** page, type in a user name and password, and then click **Next**.
7. If you selected **WS-Security**, in the **WS-Security** page, type in a user name and password. if you want to use password digest, select the **Use password digest** check box, and then click**Next**.

    Note Depending on the size of the remote service, it might take a few minutes to load the following **Operations** page.
8. In the **Operations** page, select the check box for each remote service operation you want to activate or clear the check box for any services you want to deactivate, and then click **Finish**.

    Note You can expand or collapse a service to make the process easier by clicking the expand or collapse button. Also, you can right-click anywhere on an operation and choose **Select All**,**Deselect All**, **Collapse All**, or **Expand All**.

INRemoteService

Use the INRemoteService object to handle all remote services related actions. You can create, update, or delete a remote service from ImageNow. You can also use this object to send requests from iScript to a remote web service and return a response from the remote web service to ImageNow. This object is only available if you have ImageNow Envoy. Note that an Envoy license is required to use Envoy. If you need an Envoy license, contact your Perceptive Software representative.

Another important note about Envoy capabilities involves the transcoders which are supported. For example, if you attempt to add a remote service from a site that uses an 'ISO-8859-1 transcoder, Envoy returns an exception. At this time, only the following transcoders are supported:

- `<tcdr:transcoder tcdr:class="WASP_PlainTextTranscoder" />`
- `<tcdr:transcoder tcdr:class="WASP_UTF8Transcoder" pt:name="UTF-8" xmlns:pt="urn:UTF-8"/>`
- `<tcdr:transcoder tcdr:class="WASP_UTF16Transcoder" pt:name="UTF-16LE" pt:littleEndian="1" xmlns:pt="urn:UTF-16"/>`
- `<tcdr:transcoder tcdr:class="WASP_UTF16Transcoder" pt:name="UTF-16BE" pt:littleEndian="0" xmlns:pt="urn:UTF-16"/>`

Same Name attributes

Works in all ImageNow versions (element has an attribute but not a value).

```
<element class='xyz'></element>
<class>xyz</class>
```

Works in all ImageNow versions (element has a value but not an attribute).

```
<element>xyz</element>
<class>xyz</class>

<element class='xyz'>xyz</element>
<class>xyz</class>
```

Constant Definitions

```
INRemoteServiceType
   WEBSERVICE = 1

INWSAuthType
   None = 0
   HttpBasic = 1
   HttpDigest = 2
   SSL = 3
   WSSECURITY = 4

INWSSPasswordType
   PlainText = 1
   Digest = 2
```

Constructor Summary

```
INRemoteService()
INRemoteService(string remoteServiceName)
```

Data Members

Type	Name	Description
authType	INWSAuthType	The authentication type used for the remote web service adaptor.
desc	String	The remote service description.
id	String	The unique remote service ID.
name	String	The remote service name.
type	INRemoteServiceType	The remote service type.
wsdl	String	The URL to the associated WSDL file.

Functions

Type	Function Description

array of INRemoteService	`INRemoteService` addWS(string serviceName, string url)
	INRemoteService addWS(string serviceName, string url, string desc)
	INRemoteService addWS(string serviceName, string url, string desc, INWSAuthType/string authType, string username, string password)
	INRemoteService addWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, string username, string password, INWSSPasswordType
	
	Use this function to create a new remote web service in ImageNow. When the ImageNow Server creates a new web service, it automatically finds all services and operations in the WSDL file and adds them into the database. All operations are available when added for the first time unless you choose to make some operations not available by using the ActivateOperation method. There is no http-level authentication for the remote service by default.
	
	When you add a new remote web service, you can specify additional authentication types and any additional information required by the client authentication type. All of these client authentication types are mutually exclusive.
	
	For example, when you want HTTPBASIC client authentication, use the following call:
	
	addWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, string username, string password)
	
	When using WS-Security, you must pass your user_name and password with each call to callOperation the same as you do with HTTP basic authentication. When you want WS-Security client authentication, use the following call:
	
	addWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, string username, string password, INWSSPasswordType /string passwordType)
	
	addWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, sCertFilePath, sCertFilePwd, sPrivateKeyFilePath, sPrivateKeyFilePwd)
	
	Sample Script 1
	
	// Adds a new web service without http-level authentication
	
	`var rs = INRemoteService.addWS("My ERP Remote Service",`
	`    "http://myServer/services/WSDL",`
	`    "This is the primary ERP Web service adaptor");`
	
	Sample Script 2
	
	// Adds a new web service with http-level authentication
	
	`var rs = INRemoteService.addWS("My ERP Remote Service", "`
	`http://myServer/services/WSDL ",`
	`    "This is the primary ERP Web service adaptor", "HttpBasic", "myUser", "myPassword");`
	
	Sample Script 3
	
	// Adds a new web service with WS-Security
	
	`var access_rs = INRemoteService.addWS("access_service", access_wsdl, "desc", "WSSECURITY", "myusername",`
	`"mypassword", "digest");`

Type	Function Description

	Sample Script 4
	// Add a new web service with SSL
	`var access_rs = INRemoteService.addWS("access_service", access_wsdl, "desc", "SSL", "c:\\certs\\client-cert.pem", "", "c:\\certs\\client-key.pem", "clientkeypassword");`
	`/string passwordType)*`
	`INRemoteService addWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, sCertFilePath, sCertFilePwd, sPrivateKeyFilePath, sPrivateKeyFilePwd)*`

array of INRemoteService	`getAllWS()`
	Use this function to get all remote web services in ImageNow Envoy .
	Sample Script
	`var wss = INRemoteService.getAllWS();`
	`if (wss)`
	`{`
	`     for (ii=0; ii<wss.length; ii++)`
	`     {`
	`          printf("Name: %s, Type: %d, Description: %s, WSDL URI: %s, Authentication Type: %s\n", wss[ii].name, wss[ii].type, wss[ii].desc,`
	`               wss[ii].wsdl, wss[ii].authType);`
	`     }`
	`}`

bool	`setLoggingInterceptor` (sRSName, bStatus, sLogPath) Use this function to enable or disable interceptor logging for a remote web service from the ImageNow database. Use interceptor logging to view SOAP requests and responses for troubleshooting and informational purposes. All requests submitted to any of the endpoints that were previously enabled have their SOAP request and response logged in separate text files. These files are named debugXXXIn.txt and debugXXXOut.txt (where XXX is the sequence number of the request and response). These files are located in the logDir directory <path> specified by using the sLogPath parameter. The log files are incrementally numbered when you make calls using workflow scripts. When the Workflow Agent service is restarted, the numbering is also restarted. Note that when you make calls using Intool.exe, all log files contain 1 in the file name instead of incremental numbers. **Note** Using the logging interceptor configuration through iScript supersedes any configuration set up in the XML configuration files.

Type	Function Description
	setLoggingInterceptor Data Members (see below)

setLoggingInterceptor Data Members

Data Type	Data Member Name	Description
String	sRSName	The name of the remote web service to modify.
Boolean	bStatus	Turn interceptor logging on or off.
String	sLogPath	The log path to use for the remote web service interceptor logging. This path must be an existing path.

Sample Script

```
setLoggingInterceptor(sRSName, bStatus, string logPath, string authtype,
 string username, string password)
```

```
setLoggingInterceptor(sRSName, bStatus, string logPath, string authtype,
 string username, string password, int/string passwordType)
```

```
setLoggingInterceptor(sRSName, bStatus, string logPath, string authtype,
 string sCertFilePath, string sCertFilePwd, string sPrivateKeyFilePath,
 string sPrivateKeyFilePwd)
```

Type	Function Description
bool	`setSSLCredentials` (sRSName, sCertFilePath, sCertFilePwd, sPrivateKeyFilePath, sPrivateKeyFilePwd)

To enable SSL with a remote web service, you need to add the remote web service using the addWS function with an authType equal to "SSL". This function can be used to update or change the credentials used for a remote service that is using SSL.

setSSLCertficate Data Members

Data Type	Data Member Name	Description
String	sRSName	The name of the remote web service to modify.
String	sCertFilePath	The path where the file holding a certificate in PEM or PKCS #12 format is located.
String	sCertFilePwd	The password to use to decrypt the certificate, if one is required.
String	sPrivateKeyFilePath	The path where the file holding a private key in PEM or PKCS #12 format is located.
String	sPrivateKeyFilePwd	The password to use to decrypt the private key, if one is required.

Sample Script

// Adds a new web service with SSL authentication

```
var bRet = INRemoteService.setSSLCredentials("access_service",
"C:\\certs\\new-cert.pem",
```

```
"", "C:\\certs\\new-key.pem", "newKeyPassword");
```

Methods

Type	Method Description

bool	`activateOperation` (string: operationName, bool isActive = true)
	
	Activate or de-activate the operation of the remote service. When there are multiple operations with same name for this remote service, the ImageNow Server activates or de-activates all operations associated with the provided operation name. If an operation is de-activated, an ImageNow user cannot make a call to that operation.
	
	Sample Script
	
	// De-activates an operation.
	
	`var rs = INRemoteService ("My ERP Remote Service");`
	
	`rs.activeOperations("VALIDATE_PO", false);`
	
	`activateOperation (OperationObject: operation,  bool isActive = true)`
	
	Activate or de-activate the operation of the remote service. When there are multiple operations matching operation object criteria, ImageNow Server activates or de-activates all operations associated with the name of the provided operation. The data members in the returned objects are defined in the OperationObject Data Member table in this help topic under the getAllServicesAndOperations() method. If an operation is de-activated, an ImageNow user cannot make a call to that operation.
	
	Sample Script
	
	// Activates an operation which has overloading.
	
	`var rs = INRemoteService ("My ERP Remote Service");`
	
	`var op = new Object;`
	
	`op.name = "VALIDATE_PO"; // required`
	
	`op.inputName = "…"; // optional`
	
	`op.outputName = "…"; // optional`
	
	`op.serviceName = "…"; // optional`
	
	`op.serviceNamespace = "…"; // optional`
	
	`op.port = "…"; // optional`
	
	`rs.activeOperations(op, false);`

bool	`callOperation` (string operationName, in ParamValObject inputVal, out ParamValObject outputVal, out ParamValObject fault, string username, string password, in ParamValObject headerVal) **callOperation**(OperationObject op, in ParamValObject inputVal, out ParamValObject outputVal, out ParamValObject fault, string username, string password, in ParamValObject headerVal) Use this method to invoke a remote web service operation. The data members in the return objects are defined in the OperationObject Data Member table in this help topic under the getAllServicesAndOperations() method. When an operation is overloaded, using OperationObject specifies a unique operation. If there are multiple operations with same name or matching the criteria in OperationObject, the ImageNow Server picks the first one. If the operation is using polymorphism, set the optional wsdlType and wsdlTypeNS parameters to specify which type is used for the input object.

Type	Method Description
	The data members in the return objects are defined in the ParamValObject Data Member table in this help topic under the getParamInfo() method. Username, password, and headerVal are optional parameters. Username and password are only used if the remote web service requires http-level authentication. If there are multiple operations with same name or matching the criteria in OperationObject, the ImageNow Server picks the first operation. When there is a failure, the method returns false and the ImageNow Server sends an error message. If the remote service returns an exception defined in the WSDL file, a fault object will be set. When setting the inputVal object, the data member name and type are defined by the WSDL file. The inputVal Object Data Members table contains the fields you may need to set, the fields to set depends on your use.

inputVal Object Data Members

iScript Data Type	Data Member Nam	ImageNow Envoy Type and Description
String	<name in ParameterObject>	WASP_Dll_String
Number	<name in ParameterObject>	WASP_Dll_Int, WASP_Dll_Float, WASP_Dll_Double
boolean	<name in ParameterObject>	WASP_Dll_Boolean
Date	<name in ParameterObject>	WASP_Dll_DateTime
Object	<name in ParameterObject>	WASP_Dll_Structure
Array	<name in ParameterObject>	WASP_Dll_Array
String	wsdlType	Optional. If undefined, the ImageNow Server uses the type defined in operation schema. When the operation is using polymorphism, wsdlType and wsdlTypeNS provide a way to uniquely define a derived WSDL type other than the type in the operation parameter definition.
String	wsdlTypeNS	Optional. If undefined, ImageNow Server uses the type defined in operation schema. When the operation is using polymorphism, wsdlType and wsdlTypeNS provide a way to uniquely define a derived WSDL type other than the type in the operation parameter definition.

Type	Method Description
	Sample Script 1

// Normal operation without overloading and polymorphism, using the operation name when invoking the remote service operation.

```
var rs = INRemoteService ("My ERP Remote Service");
var input = new Object;
input.parameter = new Object;
input.parameter.INVOICE_NUM = "12345";
input.parameter.ERP_USER_NAME = userName;
input.parameter.PASSWORD = password;
input.parameter.PO_NUM = "A05847";
input.parameter.VENDOR_NAME = "Acme Office Supplies";
input.parameter.INVOICE_AMOUNT = 7500.00;
var output = "";
var faults = "";
var ret = rs.callOperation("VALIDATE_PO", input, output, faults);
if (ret)
{
     var poObject = output.parameter.PURCHASE_ORDER;
     if (poObject != null)
     {
           printf(”r;Purchase Order is valid.\n”);
           printf("PO ID: %s\n", poObject.PO_ID);
           printf(”r;PO Date: %s\n”, poObject.PO_DATE);
     }
}
```

Sample Script 2

```
// Overloaded operation, use the operation Object to define the operation.
var rs = INRemoteService ("Overload Remote Service");
var op = new Object;
op.name = "getList";
op.inputName = "getList_1_input";
op.outputName = "getList_1_output";
var input = new Object;
input.p0 = 125; // input is integer
var output = null;
```

Type	Method Description

	`var ret = rs.callOperation(op, input, output);`
	`if (ret)`
	`     printf(" getList 1 Return [%s]\n", output.response);`
	` else`
	`     printf("Failed to complete request - %s\n", getErrMsg());`
	
	Sample Script 3
	`// 2nd call to the overloaded operation.  Note the input is a string here.`
	`var rs = new INRemoteService("Overload Remote Service");`
	`var op = new Object;`
	`op.name = "getList";`
	`op.inputName = "getList_3_input";`
	`op.outputName = "getList_3_output";`
	`var input = new Object;`
	`input.p0 = "309"; // input is string`
	`var output = null;`
	`var ret = rs.callOperation(op, input, output);`
	
	Sample Script 4
	`// Polymorphic implementation example 1.`
	`var rs = new INRemoteService("Polymorphic Remote Service");`
	`var operationName = ”r;echoObject”;`
	`var circle = new Object;`
	`circle.wsdlType = "Circle";`
	`circle.wsdlTypeNS = "http://systinet.com/wsdl/default/";`
	`circle.name = "My Circle"; // from base type`
	`circle.center = new Object; // center, radius are from Circle Object`
	`circle.center.x = 2;`
	`circle.center.y = 3;`
	`circle.radius = 23;`
	`var input = new Object;`
	`input.p0 = circle;`
	`var output;`
	`var faults;`
	`var ret = rs.callOperation(operationName, input, output, faults);`

Type	Method Description

	Sample Script 5
	`// Polymorphic implementation example 2.`
	`var rs = new INRemoteService("Polymorphic Remote Service");`
	`var operationName = ”r;echoObject”;`
	`var triangle = new Object;`
	`triangle.wsdlType = "Triangle";`
	`triangle.wsdlTypeNS = "http://systinet.com/wsdl/default/";`
	`triangle.name = "My triangle";  // from base type`
	`triangle.a = new Object; // a, b, c are points from the Triangle Object`
	`triangle.a.x = 1;`
	`triangle.a.y = 1;`
	`triangle.b = new Object;`
	`triangle.b.x = 4;`
	`triangle.b.y = 4;`
	`triangle.c = new Object;`
	`triangle.c.x = 8;`
	`triangle.c.y = 8;`
	`var input = new Object;`
	`input.p0 = triangle;`
	`var output;`
	`var faults;`
	`var ret = rs.callOperation(operationName, input, output, faults);`
	Sample Script 5
	`// Example of how to use the header parameter.`
	`var input = new Object();`
	`var output = new Object();`
	`var faults = new Object();`
	`var header = new Object();`
	`input.COMPANY = "PSI";`
	`input.LOCATION = "Shawnee";`
	`header.UsernameToken = new Object();`
	`header.UsernameToken.Username="test1";`
	`header.UsernameToken.Password="ImageNow!";`
	`// call op`
	`var rtn = rs.callOperation("EchoHeader", input, output,`
	`faults,"","",header);`

Type	Method Description
array of Object	`getAvailableOperations()` Use this method to get all available operations for the remote service from the database. The data members in the return objects are defined in the OperationObject Data Member table in this help topic under the getAllServicesAndOperations() method. **Sample Script**

```
var rs = INRemoteService ("My ERP Remote Service");
var ops = rs.getAvailableOperations();
if (ops)
{
     for (i=0; i<ops.length; i++)
         printOperation(ops[i]);
}
```

Type	Method Description
array of Object	`getAllServicesAndOperations()` Use this method to get all of services and their operations for a remote web service from the ImageNow database. Each object in the result array contains the following data members.

getAllServicesAndOperations Data Members

Data Type	Data Member Name	Description
String	serviceName	Service name in the WSDL file.
String	serviceNamespace	Service namespace in the WSDL file.
String	port	Port name in the WSDL file.
String	desc	Description for service in the WSDL file.
array of Operation Object	operations (See Operation Object table for data members.)	Operations associated with this service and port in the WSDL file.

OperationObject Data Members

Data Type	Data Member Name	Description
String	id	Defined by ImageNow, not by the WSDL file.
String	name	Operation name in the WSDL file.
String	desc	Documentation under operation in the WSDL file.

Type	Method Description

String	inputName	Operation input name (optional) in the WSDL file. The inputName and outputName are optional data members. However, you must define these data members for an overloaded operation. For an overloaded operation, use inputName and outputName to uniquely decide which operation you want.
String	outputName	Operation output name (optional) in the WSDL file. The inputName and outputName are optional data members. However, you must define these data members for an overloaded operation. For an overloaded operation, use inputName and outputName to uniquely decide which operation you want.
String	serviceName	Service name to which the operation belongs.
String	serviceNamespace	Service namespace.
String	port	Port name.
boolean	isActive	Defined by ImageNow, not by the WSDL file. True: Allow ImageNow user to invoke the operation. False: Do not allow ImageNow user to invoke the operation.

Sample Script

```
var rs = INRemoteService ("My ERP Remote Service");
var services = rs.getAllServicesAndOperations();
for (var ii=0; ii<services.length; ii++)
    {
          printf("ServiceName: %s, ServiceNamespace: %s\n",
                  services[ii].serviceName,
services[ii].serviceNamespace);
          printf("Port:%s\n", services[ii].port);
          printf("Service Description:%s\n", services[ii].desc);
          if (services[ii].operations)
          {
              printf("Number of Operations: %d\n",
services[ii].operations.length);
              for(var jj=0; jj<services[ii].operations.length; jj++)
                  printOperation(services[ii].operations[jj]);
          }
    }
function printOperation(op)
{
```

Type	Method Description

	`printf("Operation Name: %s,   ID: %s,  Is Active: %s\n", op.name,`
	`op.id, op.isActive);`
	`printf("   Input Name: %s\n", op.inputName);`
	`printf("   Output Name: %s\n", op.outputName);`
	`printf("   Description: %s\n", op.desc);`
	`printf("   Service Name: %s\n", op.serviceName);`
	`printf("   Service Namespace: %s\n", op.serviceNamespace)`
	`printf("   Port: %s\n", op.port);`
	`}`

array of OperationObject	`getOperationsByName` (string operationName)
	
	Use this method to get all operations with the given name for the remote service. Because WSDL allows operation overloading, this method may return more than one object. However, the operationName input and the operationName output must be different. If the operationName as input or output is not provided, the method returns the first operation if there is more than one operation in the database. The data members in the return objects are defined in the OperationObject Data Member table in this help topic under the getAllServicesAndOperations() method.
	
	Sample Script
	`var rs = INRemoteService ("My ERP Remote Service");`
	`var ops = rs.getOperationsByName("VALIDATE_PO");`
	`if (ops)`
	`{`
	`for (i=0; i<ops.length; i++)`
	`printOperation(ops[i]);`
	`}`

bool	`getParamInfo` (string operationName, out Object inputParm, out Object outputParm, out array of Object faults, string username, string password) `getParamInfo` (OperationObject operation, out ParameterObject inputParm, out ParameterObject outputParm, out array of ParameterObject faults, string username, string password) Use this method to get parameter input, output, and faults definition for an operation. Use this information to set input value and read output value, as well as handle any exceptions/faults. The data members in the return objects are defined in the OperationObject Data Member table in this help topic under the getAllServicesAndOperations() method. When an operation is overloaded, using OperationObject specifies a unique operation. If there are multiple operations with same name or matching the criteria in OperationObject, the ImageNow Server picks the first one. Refer to the ParameterObject Data Members table for details. Username and password are optional parameters. Username and password are only used if the remote web service requires http-level authentication.

Type	Method Description
	ParameterObject Data Members

Data Type	Data Member Name	Description
String	name	Input, output, or fault message part name in the WSDL file. Fault name is required in the WSDL file, other names may not be defined in the WSDL file.
String	message	Message name in the WSDL file.
String	type	The data type of the associated parameter. Possible values include: WASP_Dll_String, WASP_Dll_Boolean, WASP_Dll_Int, WASP_Dll_Date, WASP_Dll_Structure, WASP_Dll_Array
String	desc	The optional detailed description (annotation) associated with the operation. This is defined in wsdl <element><annotation><documentation> …desc … </documentation></annotation></element>
Boolean	isOptional	Indicates if the parameter is optional or required. True if the Nillable=true or minOccurs = 0 attributes occur in the WSDL file for the element.
String	wsdlType	The wsdltype for the parameter as defined in the WSDL file.
String	wsdlTypeNS	The namespace of the wsdltype as defined in the WSDL file.
String	kind	The specific parameter type, based upon the element definition within the associated WSDL file. Possible values include: PRIMITIVE, ALL, SEQUENCE, CHOICE, UNION, LIST, ENUMERATION, UNKNOWN, ANY, or GROUP
FacetObject	facet	Refer to the Available Facet Types table shown below for possible facet members. Not all the members of that table are required to be defined for a given parameter.

Type	Method Description

Array of String	enumeration	Returns acceptable values for the name data member that is passed in. Note this is only available for parameters that are defined as an ENUMERATION (see ”r;kind” attribute”).
Array of ParameterObject	members	Contains all sub-elements definition of a complex type parameter, as defined in the associated WSDL file.

Available Facet Types

Data Member Name	Description
Length	The number of units of length. Units vary according to the base type. The simpleType must be this number of units of length. For example, if xsd:string is the base type, you might specify 5 as the length if you know that each value is a code that always has five characters.
MinLength	The minimum number of units of length. Units vary according to the base type. The length of the instances of this simpleType must be equal to or more than this number of lengths.
MaxLength	The maximum number of units of length. Units vary according to the base type. The length of the instances of this simpleType must be less than or equal to this number of lengths.
Pattern	A regular expression. The values of the simpleType must be literals that match this regular expression.
Enumeration	One allowable value. An enumeration facet is added for each allowable value.
MaxInclusive	The inclusive upper bound of the range of values allowed for this simpleType. The value of the simpleType must be less than or equal to the value of maxInclusive.
MinInclusive	The inclusive lower bound of the range of values allowed for this simpleType. The value of the simpleType must be equal to or more than the value of minInclusive.
MaxExclusive	The exclusive upper bound of the range of values allowed for this simpleType. The value of the simpleType must be less than the value of maxExclusive.
MinExclusive	The exclusive lower bound of the range of values allowed for this simpleType. The value of the simpleType must be more than the value of minExclusive.

Type	Method Description

WhiteSpace	May be defined in the WSDL file, but not returned as a data member. Specify one of the following values: "preserve" indicates that no normalization is done. The value is not changed. "replace" indicates that each tab, line feed, and return is replaced with a space. "collapse" indicates that the processing specified by the replace is done, and then contiguous sequences of spaces are collapsed into one space.
TotalDigits	The maximum number of digits that are allowed in values of simpleTypes that are derived from xsd:decimal.
FractionDigits	The maximum number of digits that are allowed in the fractional portion of values of simpleTypes that are derived from xsd:decimal.

Sample Script 1

```
var rs = INRemoteService ("My ERP Remote Service");
     var inputInfo, outputInfo, faults;
if (!rs.getParamInfo(”r;VALIDATE_PO”, inputInfo, outputInfo, faults))
// error, return


    if (inputInfo)
    {
            printf("Input Message: %s, Name: %s\n", inputInfo.message, 
inputInfo.name);
            printParamInfoList(inputInfo.members);
    }
    if (outputInfo)
    {
           printf("Output Message: %s, Name: %s\n", outputInfo.message, 
outputInfo.name);
           printParamInfoList(outputInfo.members);
    }
    if (faults)
    {
        for (ii=0; ii<faults.length; ii++)
        {
            printf("Fault Name: %s, Message: %s\n", faults[ii].name, 
            printParamInfoList(faults[ii].members);
        }
```

Type	Method Description

	```
	    }
	}
	```
	Sample Script 2
	`// getParmInfo for an overloaded operation.`
	`var rs = new INRemoteService("My ERP Remote Service");`
	`// set operation object.  name is required, others are optional`
	`var op2 = new Object;`
	`op2.name = "VALIDATE_PO"; // required`
	`op2.inputName = "ValidatePO_2_input";`
	`op2.outputName = "ValidatePO_2_output";`
	`op2.serviceName = "…";`
	`op2.serviceNamespace = "…";`
	`op2.port = "…"`
	
	`var inputInfo, outputInfo, faults;`
	`var username = ″r;MyUser″;`
	`var password = ″r;MyPassword″;`
	
	`// User name & password are required only using http level authentication`
	`rs.getParamInfo(op2, inputInfo, outputInfo, faults, username, password);`

bool	`remove()`
	Use this method to delete an existing remote service.
	Sample Script
	`var rs = INRemoteService ("My ERP Remote Service");`
	`rs.remove();`

bool	`updateWS` (string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, string username, string password) updateWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, string username, string password, INWSSPasswordType/string passwordType)* updateWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, sCertFilePath, sCertFilePwd, sPrivateKeyFilePath, sPrivateKeyFilePwd)* Use this method to update an existing remote web service. If you leave the optional parameters empty, the value in database remains the same. When you add a new remote web service, you specify the authentication type and any additional information required by the client authentication type. All of these client authentication types are mutually exclusive. For example, when you want HTTPBASIC client authentication, use the following call.

Type	Method Description

	updateWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, string username, string password)
	
	When using WS-Security, you must pass your user_name and password with each call to callOperation the same as you do with HTTP basic authentication. When you want WS-Security client authentication, use the following call.
	
	updateWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, string username, string password, INWSSPasswordType/string passwordType)
	
	updateWS(string newServiceName, string newWsdl, string newDesc, INWSAuthType/string authentication, sCertFilePath, sCertFilePwd, sPrivateKeyFilePath, sPrivateKeyFilePwd)
	
	Sample Script 1
	
	// Updates the remote service name, where the description does not change.
	
	`var rs = INRemoteService ("My ERP Remote Service");`
	
	`rs.update("Updated ERP Remote Service");`
	
	Sample Script 2
	
	`// Updates the remote service name, where the description does change.`
	
	`var rs = INRemoteService ("My ERP Remote Service");`
	
	`rs.update("Updated ERP Remote Service", " http://myServer/services/WSDL ",`
	
	`    "This is the ERP Web service adaptor – updated as of v2.5", INWSAuthType.HttpBasic,`
	
	`    "myUser","myPassword");`
	
	Sample Script 3
	
	`// Update the remote web service with WS-Security`
	
	`var access_rs = INRemoteService.updateWS("access_service", access_wsdl, "desc", "WSSECURITY", "myusername",`
	
	`"mypassword", "digest");`
	
	Sample Script 4
	
	`// Updates the remote web service with SSL`
	
	`var access_rs = INRemoteService.updateWS("access_service", access_wsdl, "desc", "SSL", "c:\\certs\\client-cert.pem", "", "c:\\certs\\client-key.pem", "clientkeypassword");`