KnowWho®
Government Relations Contact Solutions

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

# Elected Officials Communication Service (EOCS) Developer Guide

The KnowWho Elected Officials Communications Service (EOCS) was developed to facilitate electronic communication with elected officials through the most efficient methodology, including e-mail, automatic web form submission, and the new Communicating with Congress service. The EOCS accepts and returns strings of XML and JSON, with the input containing standardized values for electronic message processing and the output containing a summary of the results of the submission(s).

## SUBMITTING INPUT XML

```
<KnowWho>
        <Customer>
                <CustomerCode></ CustomerCode>
                <ExcludePeople></ ExcludePeople >
                <ExcludeMajMin></ ExcludeMajMin>
                <ExcludeParty></ ExcludeParty>
                <ExcludeGender></ ExcludeGender>
                <USExecutive>Y</ USExecutive>
                <USSenate>Y</ USSenate>
                <USHouse>Y</ USHouse>
                <StateExecutive>N</ StateExecutive>
                <StateSenate>N</ StateSenate>
                <StateHouse>N</ StateHouse>
                <CountyExecutive>N</ CountyExecutive>
                <CountyLegislature>N</ CountyLegislature>
                <MunicipalExecutive>N</ MunicipalExecutive>
                <MunicipalLegislature>N</ MunicipalLegislature>
        </Customer>
        <FormData>
                ** see instructions below
        </FormData>
</KnowWho>
```

## SUBMITTING INPUT JSON

```
{
  "KnowWho": {
    "Customer": {
      "CustomerCode": "[value]",
      "ExcludePeople": "[value(s)]",
      "ExcludeMajMin": "[value(s)]",
      "ExcludeParty": "[value(s)]",
      "ExcludeGender": "[value(s)]",
      "USExecutive": "Y",
      "USSenate": "Y",
      "USHouse": "Y",
      "StateExecutive": "N",
      "StateSenate": "N",
      "StateHouse": "N",
      "CountyExecutive": "N",
      "CountyLegislature": "N",
```

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

```
      "MunicipalExecutive": "N",
      "MunicipalLegislature": "N",
    },
    "FormData": {
        ** see instructions below
    }
  }
}
```

**CustomerCode**
You should submit the code given to you at the time of registration. If you have not received a code, please contact our customer support.

**ExcludePeople**
This node accepts one or more PersonID values obtained from either the response from an EODS request or any KnowWho data service to which you are subscribed. The entry can be a single value or multiple values separated by commas without any spaces. Any person identified in this node will be excluded from message processing for that particular call to the service.

**ExcludeMajMin**
This node accepts one of three values: "MAJ", "MIN", or a blank entry. "MAJ" will exclude all identified members of the majority and "MIN" will exclude all identified members of the minority.

**ExcludeParty**
This node accepts one of four values: "R", "D", "O", or a blank entry. "R" will exclude all identified members of the Republican Party, "D" will exclude all identified members of the Democratic Party, and "O" will exclude all identified members of any other party.

**ExcludeGender**
This node accepts one of three values: "M", "F", or a blank entry. "M" will exclude all identified male officials and "F" will exclude all identified female officials.

The exclusion filters are limited to the use of ExcludePeople alone; ExcludeMajMin may be combined with ExcludeGender only; ExcludeParty may be combined with ExcludeGender; all filters may be used individually. Any other combination in the request will return an error.

For each of the other nodes, use "Y" or "N" as the node text to designate whether or not you would like to contact the elected officials represented by that branch/level of government. **Please note: if you submit a "Y" value for a level of government you are not subscribed to, your information will not be submitted to any elected officials at that level.**

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

## FORM DATA FIELDS

The following fields are **required** in the FormData node, with these node names and the described guidelines for the values (where applicable):

1. PrefixCode: integer value from the list of acceptable prefix options (see "Value Options")
2. FirstName
3. LastName
4. Address1
5. City
6. State
7. ZipCode: five-digit zip code (5 digits, no punctuation)
8. Telephone
9. Email
10. Issue: integer value from the list of acceptable issue options (see "Value Options")
11. Subject: brief description of the message content
12. Message: text of the message

The following fields are **optional**, with the same details as above:

1. SuffixCode: integer value from the list of the acceptable suffix options (see "Value Options")
2. Address2
3. Fax
4. Plus4: four-digit zip code extension (4 digits, no punctuation)

## FIELD VALUE OPTIONS

**Prefixes:**

| | |
|---|---|
| Mr. | 01 |
| Mrs. | 02 |
| Ms. | 03 |
| Miss | 04 |
| Hon. | 05 |
| Dr. | 08 |
| Rev. | 09 |
| Prof. | 10 |
| Dean | 11 |
| Pres. | 12 |
| Rabbi | 30 |
| Iman | 31 |

**Suffixes:**

| | |
|---|---|
| Jr. | 01 |
| Sr. | 02 |
| II (Second) | 03 |
| III (Third) | 04 |
| IV (Fourth) | 05 |

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

**Issues:**

| | |
|---|---|
| Abortion | 56 |
| Acquisitions | 57 |
| Aerospace / Space | 40 |
| Agriculture / Food | 2 |
| Animal Rights | 3 |
| Appropriations | 4 |
| Arts / Humanities | 5 |
| Banking | 58 |
| Budget | 59 |
| Elections / Campaigns | 8 |
| Census / Redistricting | 9 |
| Commerce / Business | 7 |
| Communications | 12 |
| Congress | 60 |
| Consumer Affairs | 13 |
| Crime / Law Enforcement | 14 |
| Disabilities / ADA | 16 |
| Disaster / Emergency Management / FEMA | 17 |
| Economic Development / Economic Policy | 55 |
| Education / Schools | 18 |
| Energy / Utilities | 19 |
| Entertainment / Media | 20 |
| Environment / Natural Resources | 21 |
| Family / Children | 10 |
| Finance | 6 |
| Foreign Policy / International Affairs | 22 |
| Gambling | 61 |
| Government Operations / Government Reform | 23 |
| Grants / Interns | 41 |
| Gun Issues | 62 |
| Health / Medicine | 24 |
| Homeland Security / Terrorism | 25 |
| Housing | 26 |
| Human Rights / Civil Rights | 11 |
| Immigration | 27 |
| Insurance | 49 |
| Intelligence / Counterintelligence | 28 |
| Intergovernmental Affairs / State / Local | 53 |
| Internet / Social Media | 69 |
| Judiciary / Law / Courts | 29 |
| Labor / Jobs / Pensions | 30 |
| Land Use / Water / Oceans | 46 |
| Lobbying / Politics | 63 |
| Logistics / Planning | 52 |
| Medicare / Medicaid | 31 |
| Military / Defense | 15 |
| Minority / Ethnic | 33 |

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

| | |
|---|---|
| National Security | 64 |
| Native American Affairs | 65 |
| Privacy / Personal Rights | 70 |
| Public Affairs / Outreach | 54 |
| Public Works / Infrastructure | 32 |
| Recreation / Sports / Parks | 34 |
| Regulation / Regulatory | 50 |
| Religion | 35 |
| Rules / Ethics | 36 |
| Rural Affairs | 37 |
| Science | 38 |
| Seniors / Aging / Elderly | 1 |
| Small Business | 66 |
| Social Security | 39 |
| Tax / Revenue | 42 |
| Technology | 67 |
| Telecommunications | 68 |
| Trade / Exports / Imports | 43 |
| Transportation | 44 |
| Urban Affairs / Urban Development | 51 |
| Veterans Affairs | 45 |
| Welfare / Social Issues | 47 |
| Womens Issues | 48 |

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

## SERVICE OUTPUT XML

The output XML will use the following format:

```
<KnowWho>
        <Customer>
                <CustomerCode></ CustomerCode>
                <RemainingCredits></ RemainingCredits>
                <ReturnCode></ ReturnCode>
        </Customer>
        <USGovernment>
                <ElectedOfficial>
                        <PersonID></ PersonID>
                        <PositionID></ PositionID>
                </ ElectedOfficial>
        </USGovernment>
        <StateGovernment>
                <ElectedOfficial>
                        <PersonID></ PersonID>
                        <PositionID></ PositionID>
                </ ElectedOfficial>
        </StateGovernment>
        <LocalGovernment>
                <ElectedOfficial>
                        <PersonID></ PersonID>
                        <PositionID></ PositionID>
                </ ElectedOfficial>
        </LocalGovernment>
</KnowWho>
```

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

## SERVICE OUTPUT JSON

The output JSON will use the following format:

```
{
   "KnowWho": {
      "Customer": {
         "CustomerCode": "",
         "RemainingCredits": "",
         "ReturnCode": ""
      }
      "USGovernment": {
         "ElectedOfficial": {
            "PersonID": "",
            "PositionID": ""
         }
      },
      "StateGovernment": {
         "ElectedOfficial": {
            "PersonID": "",
            "PositionID": ""
         }
      },
      "LocalGovernment": {
         "ElectedOfficial": {
            "PersonID": "",
            "PositionID": ""
         }
      }
   }
}
```

## SERVICE OUTPUT FIELDS

The CustomerCode node will reiterate the submitted CustomerCode.

The RemainingCredits node will display the customer's credit balance after processing the request.

The ReturnCode will display a status to summarize the service request. Possible values are:

- "NOT AUTHORIZED": An error occurred while validating the customer. Possible causes include an inactive account, access from an IP address that has not been submitted to KnowWho, and an incorrect CustomerCode value. Please contact our customer support for a resolution.
- "INSUFFICIENT CREDITS": The customer does not have enough credits to complete the requested form submission(s). Please contact our customer support to increase your credits.
- "NOT FOUND": The address values (Address1, Address2, City, State, ZipCode, and Plus4) were not able to be resolved to an address or the officials for the address could not be determined.
- "INCOMPLETE": Only a portion of the attempted form submissions were completed successfully. This could be caused by an incorrect zip code and/or 4-digit extension.
- "OK": All elected officials were found successfully and all possible form submissions were sent.

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

# INTEGRATION WITH THE SERVICE

**URL for service access:**
http://www.knowwho.info/Services/ElectedOfficialsCommunicationsService.asmx

**Developer's notes:**
To maintain standardization across languages, the service accepts an object of type String and returns an object of type String. The returned XML String contains root attributes of version 1.0 and char-set utf-8.

# XML CODE SAMPLES

**From C#:**

- In Visual Studio, Add Web Reference to http://www.knowwho.info/Services/ElectedOfficialsCommunicationsService.asmx

```
// declare and build String containing XML
string xmlString = "<KnowWho><Customer> … </FormData></KnowWho>";

// declare new instance of the service
Namespace.ReferenceName.ElectedOfficialsCommunicationsService
serviceInstance = new Namespace.ReferenceName.
ElectedOfficialsCommunicationsService();


// call the appropriate function, passing the string
string returnedXml = serviceInstance.GetService(xmlString);

// if all was successful, returnedXml now contains a string
//  representation of the returned XML
```

**From VB.NET:**

- In Visual Studio, Add Web Reference to http://www.knowwho.info/Services/ElectedOfficialsCommunicationsService.asmx

```
// declare and build String containing XML
Dim xmlString As String
xmlString = "<KnowWho><Customer> … </FormData></KnowWho>";

// declare new instance of the service
Dim serviceInstance As New
Namespace.ReferenceName.ElectedOfficialsCommunicationsService()

// call the appropriate function, passing the String
Dim returnedXml As String
returnedXml = serviceInstance.GetService (xmlString)

// if all was successful, returnedXml now contains a string
//  representation of the returned XML
```

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

**From PHP:**

- Use PHP's SoapClient to establish a connection to the web service

```
// declare and build String containing XML
$xmlString = "<KnowWho><Customer> … </FormData></KnowWho>";

// declare new instance of the service
$client = new SoapClient(http://www.knowwho.info/Services/
ElectedOfficialsCommunicationsService.asmx);

// call the appropriate function, passing the string
$returnedXml = $client->GetService(array('InputString' => $ xmlString));

// if all was successful, $returnedXml now contains a string
//  representation of the returned XML
```

**From Java:**

- Use the "wsimport" command to import the WSDL and create the proxy class (http://www.knowwho.info/Services/ElectedOfficialsCommunicationsService.asmx?WSDL)

```
// declare and build String containing XML
string xmlString = "<KnowWho><Customer> … </FormData></KnowWho>";

// declare new instance of the service
Namespace.ElectedOfficialsCommunicationsService serviceInstance = new
Namespace.ElectedOfficialsCommunicationsService();

// call the appropriate function, passing the string
string returnedXml = serviceInstance.GetService(xmlString);

// if all was successful, returnedXml now contains a string
//  representation of the returned XML
```

**From Ruby:**

- Include require 'soap/wsdlDriver'

```
# declare and build String containing XML
_xmlString = "<KnowWho><Customer> … </FormData></KnowWho>";

# declare new instance of the service and call the appropriate
#   function, passing the string
SOAP::WSDLDriverFactory.new('http://knowwho.info/Services/ElectedOfficialsCo
mmunicationsService.asmx?WSDL').GetService({ 'InputString' => _xmlString
}).GetServiceResult;

# call the appropriate function, passing the string
_returnedXml = serviceInstance.GetService(xmlString);

# if all was successful, returnedXml now contains a string
#   representation of the returned XML
```

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

# JSON CODE SAMPLES

**From C#:**

- In Visual Studio, Add Web Reference to http://www.knowwho.info/Services/ElectedOfficialsCommunicationsService.asmx

```
// declare and build String containing JSON
string jsonString = "{ KnowWho: { Customer: { … }}}";

// declare new instance of the service
Namespace.ReferenceName.ElectedOfficialsCommunicationsService
serviceInstance = new Namespace.ReferenceName.
ElectedOfficialsCommunicationsService();

// call the appropriate function, passing the string
string returnedJson = serviceInstance.GetService(jsonString);

// if all was successful, returnedJson now contains a string
//  representation of the returned JSON
```

**From VB.NET:**

- In Visual Studio, Add Web Reference to http://www.knowwho.info/Services/ElectedOfficialsCommunicationsService.asmx

```
// declare and build String containing JSON
Dim jsonString As String
jsonString = "{ KnowWho: { Customer: { … }}}";

// declare new instance of the service
Dim serviceInstance As New
Namespace.ReferenceName.ElectedOfficialsCommunicationsService()

// call the appropriate function, passing the String
Dim returnedJson As String
returnedJson = serviceInstance.GetService (jsonString)


// if all was successful, returnedJson now contains a string
//  representation of the returned JSON
```

P.O. Box 62, Mount Vernon, VA 22121
703.619.1544 (p) | 703.619.1545 (f)
www.KnowWho.com | info@KnowWho.com

**From PHP:**

- Use PHP's SoapClient to establish a connection to the web service

```
// declare and build String containing JSON
$jsonString = "{ KnowWho: { Customer: { … }}}";

// declare new instance of the service
$client = new SoapClient(http://www.knowwho.info/Services/
ElectedOfficialsCommunicationsService.asmx);

// call the appropriate function, passing the string
$returnedJson = $client->GetService(array('InputString' => $ jsonString));

// if all was successful, $returnedJson now contains a string
//  representation of the returned JSON
```

**From Java:**

- Use the "wsimport" command to import the WSDL and create the proxy class (http://www.knowwho.info/Services/ElectedOfficialsCommunicationsService.asmx?WSDL)

```
// declare and build String containing JSON
string jsonString = "{ KnowWho: { Customer: { … }}}";

// declare new instance of the service
Namespace.ElectedOfficialsCommunicationsService serviceInstance = new
Namespace.ElectedOfficialsCommunicationsService();

// call the appropriate function, passing the string
string returnedJson = serviceInstance.GetService(jsonString);

// if all was successful, returnedJson now contains a string
//  representation of the returned JSON
```

**From Ruby:**

- Include require 'soap/wsdlDriver'

```
# declare and build String containing JSON
_jsonString = "{ KnowWho: { Customer: { … }}}";

# declare new instance of the service and call the appropriate
#   function, passing the string
SOAP::WSDLDriverFactory.new('http://knowwho.info/Services/ElectedOfficialsCo
mmunicationsService.asmx?WSDL').GetService({ 'InputString' => _jsonString
}).GetServiceResult;

# call the appropriate function, passing the string
_returnedJson = serviceInstance.GetService(jsonString);

# if all was successful, returnedJson now contains a string
#   representation of the returned JSON
```