Outbound SOAP API

SOAP Web Service

Note: *This article applies to Fuji. For more current information, see SOAP Web Service* [1] *at* http://docs.servicenow.com The ServiceNow Wiki is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Overview

The SOAP Web Services provided by ServiceNow are WS-I compliant, as outlined in the WS-I Basic Profile 1.0 [2]. Inbound SOAP web services comply with SOAP1.1 standards.

Concepts and Terminology

- Provider - publishes web service for clients to invoke (consume)
- Consumer - invokes / consumes published web service
- Standards
  - WSDL
    - Web Service Description Language
    - XML document describing functions, arguments, data schema, and endpoint (where / how to invoke the service, URL)
    - WSDL only necessary when generating SOAP envelope programmatically
  - SOAP
    - Simple Object Access Protocol
    - XML document usually HTTP posted to web service endpoint described in WSDL
    - SOAP:Envelope / SOAP:Header / SOAP:Body
  - HTTP
    - Hyper-Text Transfer Protocol
    - POST vs GET - Web Service is POSTed
    - XML vs. Form POST - Web Service is XML

Web Service Provider

ServiceNow publishes its underlying table structures and associated data with the following Web Service methods:

- Direct Web Services: Use a URL query to request a ServiceNow table's WSDL.
- Web Service Import Sets: Use import tables and transform maps to automate Web Service requests to ServiceNow tables.
- Scripted Web Services: Use custom Javascript to execute Web Services requests.

Note: *SOAP messages are sent with the assumption that the recipient is XML compliant. No encoding is applied to the SOAP message.*

Web Service Consumer

Consuming external web services is achieved using Javascript objects that represent the web service SOAP [3] envelope and the subsequent SOAP HTTP request that submits the request. Web Service Consumer documents these programmatic constructs as well as examples of how to invoke web services.

WSDL

All ServiceNow tables and import sets dynamically generate WSDL XML documents that describe its table schema and available operations. You can get a WSDL format by issuing a URL targeting a ServiceNow table with the **WSDL** parameter, for example:

```
https://myinstance.service-now.com/incident.do?WSDL
```

All dynamically generated and served ServiceNow WSDL accessible via HTTP is available for use under the terms defined in the Open Source Initiative OSI - Apache License, Version 2.0 license agreement.

OSI - Apache License, Version 2.0

```
Apache License, Version 2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the
copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other
entities that control, are controlled by, or are under common control
with that entity. For the purposes of this definition, "control" means
(i) the power, direct or indirect, to cause the direction or management
 of such entity, whether by contract or otherwise, or (ii) ownership of
 fifty percent (50%) or more of the outstanding shares, or (iii)
beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising
permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.
```

Direct Web Services

A SOAP direct web service is available for any table in the system provided the correct access control is setup. The supported format of the incoming message is document style literal XML SOAP documents (Document/Literal). To retrieve the direct web service WSDL [4] description and XML schema, point to the relative URL of ***<tablename>*.do?WSDL**. For example, to retrieve the WSDL for the Incident table on the online demo system, use the following URL:

https://<instance name>.service-now.com/incident.do?WSDL

Web Service Import Sets

Web Service Import Sets provide a web service interface to import set tables. This type of web service will transform the incoming data synchronously based on the associated transform maps by default.

SOAP Roles

To use SOAP web services, you must have the appropriate role for the operation you want to perform. Additionally, you must have any other roles required to access the target tables.

Role	Description
soap	Can perform all SOAP operations.
soap_create	Can insert new records.
soap_delete	Can delete existing records.
soap_ecc	Can query, insert, and delete records on the Queues [ecc_queue] table.
soap_query	Can query record information.
soap_query_update	Can query record information and update records.
soap_script	Can run scripts that specify a .do endpoint. This role is required for running Scripted Web Services.
soap_update	Can update records.

Overriding the SOAP Endpoint

The SOAP End-point address (where the SOAP message is posted) is consistent with the endpoint of the WSDL. In some cases, however, the WSDL may reference an incorrect endpoint URL. If this happens, it is necessary to over-ride the generated URL by creating the property **com.glide.soap_address_base_url** to contain the new URL. You may have to add the property manually as this is not an out-of-box property.

For instance, a generated SOAP Endpoint may look like this:

```
https://instance.service-now.com/incident.do?SOAP
```

You can specify a property to define the endpoint such that it goes through a proxy by setting the property:

```
com.glide.soap_address_base_url = "https://myproxy.mycompany.com/service-now/"
```

This will result in the endpoint being generated to appear as:

```
https://myproxy.mycompany.com/service-now/incident.do?SOAP
```

Enabling HTTP Compression

By default, the SOAP request is accepted un-compressed and the result of the request is returned un-compressed. To enable HTTP compression using [gzip [5]] when sending in your SOAP request, set the following HTTP header:

```
Content-Encoding: gzip
```

To receive the SOAP response compressed using [gzip [5]] send in your SOAP request with the following HTTP header:

```
Accept-Encoding: gzip
```

Debugging Incoming SOAP Envelope

To capture incoming SOAP envelope XML in the system log, add the property `glide.processor.debug.SOAPProcessor` with a value of **true**.

When enabled, this property adds the incoming SOAP envelope in the **Message** field of the system log. Disable this debugging feature as soon as you are finished so that the log is not overwhelmed with excessive and unnecessary debugging information.

Preventing Empty Elements in SOAP Messages

By default, ServiceNow does not omit empty elements, elements with NULL or NIL values, from SOAP messages.

To prevent SOAP responses from containing empty elements, an administrator can create a system property called **glide.soap.omit_null_values** and set the value to **true**. This behavior is compliant with the WSDL as all elements in a SOAP message have a **minOccurs=0** attribute and are therefore optional. In addition, this behavior prevents the instance from creating inefficient SOAP messages containing a large number of empty elements.

Set this property to **false** to allow SOAP messages to search for existing fields with empty values. For example, if an administrator wants to search for incidents with an empty **Assigned to** field from a SOAP message, then the SOAP message must be able to send an empty value for this field.

Note: *Changing the value of this property may cause unintended actions in existing web service integrations. Administrators are strongly encouraged to carefully test the new behavior to ensure that existing integrations process empty elements as expected.*

Enhancements

Fuji

- The property `glide.soap.request_processing_timeout` has a default value of 60 seconds.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/integrate/inbound-soap/concept/c_SOAPWebService.html
[2] http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html
[3] http://www.w3.org/TR/soap/
[4] http://www.w3.org/TR/wsdl
[5] http://en.wikipedia.org/wiki/Gzip

SOAP Direct Web Service API

Note: *This article applies to Fuji and earlier releases. For more current information, see SOAP Direct Web Service API Functions* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Direct SOAP API

Please note that the default return for any query is limited to 250 records.

The standard SOAP API is a set of small, globally defined functions that can be performed on a targeted resource. The targeted resource (or table) is defined in the URL by the format **https://<instance name>.service-now.com/<table name>.do**.

Data Retrieval API

Method Summary	Description
getKeys	Query the targeted table by example values and return a comma delimited list of **sys_id**.
getRecords	Query the targeted table by example values and return all matching records and their fields.
get	Query a single record from the targeted table by **sys_id** and return the record and its fields.
aggregate	Query using and aggregate functions SUM, COUNT MIN, MAX and AVG. To enable the aggregate functions, activate the Aggregate Web Service Plugin.

Data Modification API

Method Summary	Description
insert	Creates a new record for the table targeted in the URL.
insertMultiple	Creates multiple new records for the table targeted in the URL. To enable multiple inserts, activate the **Web Service Insert Multiple Plugin**.
update	Updates a existing record in the targeted table in the URL, identified by the mandatory **sys_id** field.
deleteRecord	Delete a record from the targeted table by supplying its **sys_id**.
deleteMultiple	Delete multiple records from the targeted table by example values.

Data Retrieval API

getKeys

Query the targeted table by example values and return a comma delimited list of **sys_id**.

Input Fields

Any field value in the targeted table.

Output Fields

A SOAP response element **sys_id** that contains a comma delimited list of sys_ids

Sample SOAP Messages

Sample SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inc="http://www.service-now.com/incident">

   <soapenv:Header/>

   <soapenv:Body>

      <inc:getKeys>

         <category>hardware</category>

      </inc:getKeys>

   </soapenv:Body>

</soapenv:Envelope>
```

Sample SOAP response

```
<soapenv:Envelope xmlns:inc="http://www.service-now.com/incident" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

   <soapenv:Header/>

   <soapenv:Body>

      <getKeysResponse>


         <sys_id>46e18c0fa9fe19810066a0083f76bd56,46e57642a9fe1981000b96a5dca501ff,46f1784ba9fe19810018aa27fbb23482</sys_id>

         <count>7</count>

      </getKeysResponse>

   </soapenv:Body>

</soapenv:Envelope>
```

For language-specific **getKeys** examples, see:

- Perl SOAP::Lite

- Java Apache Axis2
- Microsoft .NET
- Python

getRecords

Query the targeted table by example values and return all matching records and their fields.

Input Fields

Any field value in the targeted table.

Output Fields

The **getRecordResponse** element may contain 1 or more **getRecordsResult** elements that encapsulate elements representing the field values of records matching the query.

Sample SOAP Messages

Sample SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inc="http://www.service-now.com/incident">

   <soapenv:Header/>

   <soapenv:Body>

      <inc:getRecords>

         <number>INC0000002</number>

      </inc:getRecords>

   </soapenv:Body>

</soapenv:Envelope>
```

Sample SOAP request using an encoded query to filter where incident number is INC0000001 or INC0000002

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inc="http://www.service-now.com/incident">

   <soapenv:Header/>

   <soapenv:Body>

      <inc:getRecords>

         <__encoded_query>number=INC0000001^ORnumber=INC0000002</__encoded_query>

      </inc:getRecords>

   </soapenv:Body>

</soapenv:Envelope>
```

Sample SOAP response containing 1 record

```
<soapenv:Envelope xmlns:inc="http://www.service-now.com/incident" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

   <soapenv:Header/>

   <soapenv:Body>

      <getRecordsResponse>

         <getRecordsResult>

            <caller_id>5137153cc611227c000bbd1bd8cd2007</caller_id>

            <caller_id.email>david.loo@service-now.com</caller_id.email>

            <closed_at/>

            <number>INC0000002</number>

            <opened_at>2009-12-14 23:07:12</opened_at>

            <short_description>Can't get to network file shares</short_description>
         </getRecordsResult>

      </getRecordsResponse>

   </soapenv:Body>

</soapenv:Envelope>
```

Sample SOAP response that contains more than 1 record

```
<soapenv:Envelope xmlns:inc="http://www.service-now.com/incident" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

   <soapenv:Header/>

   <soapenv:Body>

      <getRecordsResponse>

         <getRecordsResult>

            <caller_id>5137153cc611227c000bbd1bd8cd2006</caller_id>

            <caller_id.email>rick.berzle@yourcompany.com</caller_id.email>

            <closed_at>2009-12-17 22:55:16</closed_at>

            <number>INC0000009</number>

            <opened_at>2009-12-16 22:50:23</opened_at>

            <short_description>Reset my password</short_description>

         </getRecordsResult>

         <getRecordsResult>

            <caller_id>5137153cc611227c000bbd1bd8cd2005</caller_id>

            <caller_id.email>fred.luddy@yourcompany.com</caller_id.email>

            <closed_at>2009-12-15 22:54:55</closed_at>

            <number>INC0000010</number>

            <opened_at>2009-12-10 22:53:02</opened_at>

            <short_description>Need Oracle 10GR2 installed</short_description>

         </getRecordsResult>

      </getRecordsResponse>

   </soapenv:Body>

</soapenv:Envelope>
```

For language-specific **getRecords** examples, see:

- Perl SOAP::Lite
- Java Apache Axis2
- Microsoft .NET
- Python

get

Query a single record from the targeted table by **sys_id** and return the record and its fields

Input Fields

An element **<sys_id>** identifying the **sys_id** of the record to be retrieved.

Output Fields

A getResponse element encapsulating all field values for the record retrieved.

Sample SOAP Messages

Sample SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inc="http://www.service-now.com/incident">

   <soapenv:Header/>

   <soapenv:Body>

      <inc:get>

         <sys_id>46e18c0fa9fe19810066a0083f76bd56</sys_id>

      </inc:get>

   </soapenv:Body>

</soapenv:Envelope>
```

The resulting response of a **get** function call looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemaap/envelope/"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <getResponse xmlns="http://www.service-now.com/incident">
      <active>1</active>
      <approval>not requested</approval>
      <assigned_to>46c6f9efa9fe198101ddf5eed9adf6e7</assigned_to>
      <caller_id>46b673a6a9fe19810007ab03cbd5849d</caller_id>
      <category>network</category>
      <cmdb_ci>0c43f35dc61122750182c132a29e3243</cmdb_ci>
      <comments>Testing</comments>
      <contact_type>phone</contact_type>
      <due_date>2007-10-28 13:29:45</due_date>
      <escalation>0</escalation>
      <impact>3</impact>
      <incident_state>1</incident_state>
      <knowledge>0</knowledge>
      <location>1081761cc611227501d063fd3475a2de</location>
      <made_sla>1</made_sla>
      <notify>1</notify>
      <number>INC10055</number>
      <opened_at>2007-09-18 00:32:09</opened_at>
      <opened_by>46bac3d6a9fe1981005f299d979b8869</opened_by>
      <priority>0</priority>
      <reassignment_count>0</reassignment_count>
      <severity>0</severity>
    </getResponse>
  </soap:Body>
</soap:Envelope>
```

For language-specific **get** examples, see:

- Perl SOAP::Lite
- Java Apache Axis2
- Microsoft .NET
- Python

aggregate

Functionality described here requires the **Aggregate Web Service** plugin.

Query a table using an aggregate function like SUM, COUNT, MIN, MAX, AVG.

Input Fields

Any element of the target table. In addition one or more of the **aggregate** functions (SUM, AVG etc...). A **group by** and a **having** clause may also be added.

Output Fields

A **aggregateResponse** element encapsulating all field values for the record retrieved.

Sample SOAP Messages

Sample SOAP request using **COUNT** aggregate function.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:m="http://www.service-now.com"
                   xmlns:tns="http://www.service-now.com/map"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregate>
      <COUNT>number</COUNT>
      <active>true</active>
    </aggregate>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The resulting response of a **COUNT** aggregate function call looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"

xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                   xmlns:m="http://www.service-now.com"
                   xmlns:tns="http://www.service-now.com/map"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregateResponse>
        <aggregateResult>
          <avg>2.7200</avg>
        </aggregateResult>
    </aggregateResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample SOAP request using **AVG** aggregate function with a **GROUP BY** clause.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:m="http://www.service-now.com"
                   xmlns:tns="http://www.service-now.com/map"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregate xmlns="http://www.service-now.com">
      <GROUP_BY>category</GROUP_BY>
      <active>true</active>
      <AVG>severity</AVG>
    </aggregate>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The resulting response of a **AVG** aggregate function call with a **GROUP BY** clause looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
```

```
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                   xmlns:m="http://www.service-now.com"
                   xmlns:tns="http://www.service-now.com/map"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregateResponse>
        <aggregateResult>
          <avg>1.0000</avg>
          <category>database</category>
        </aggregateResult>
        <aggregateResult>
          <avg>3.0000</avg>
          <category>hardware</category>
        </aggregateResult>
        <aggregateResult>
          <avg>3.0000</avg>
          <category>inquiry</category>
        </aggregateResult>
        <aggregateResult>
          <avg>2.0000</avg>
          <category>network</category>
        </aggregateResult>
        <aggregateResult>
          <avg>2.6923</avg>
          <category>software</category>
        </aggregateResult>
    </aggregateResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample SOAP request using an encoded query to filter the aggregate:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:m="http://www.service-now.com"
                   xmlns:tns="http://www.service-now.com/map"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
```

```
    <aggregate>
      <COUNT>number</COUNT>
      <active>true</active>
      <__encoded_query>number=INC0000001^ORnumber=INC0000002</__encoded_query>
    </aggregate>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sample aggregate request using Having to narrow the results. Having takes four fields each delimited by "^": the aggregate type, the field of the aggregate, the operation type, and the value to compare . More than one Having can be added to the request so you can and Having expressions, but there is no support for OR.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/encoding/"

xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                   xmlns:m="http://www.service-now.com"
                   xmlns:tns="http://www.service-now.com/map"
                   xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <aggregate>
      <COUNT>sys_id</COUNT>
      <GROUP_BY>internal_type</GROUP_BY>
      <HAVING>COUNT^*^>^10</HAVING>
      <HAVING>COUNT^*^<^20</HAVING>
      <COUNT>sys_id</COUNT>
      <active>true</active>
    </aggregate>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Data Modification API

insert

Creates a new record for the table targeted in the URL

Input Fields

All fields from the targeted table, excluding system fields. Fields configured as mandatory in the System Dictionary are reflected in the WSDL with the attribute `minOccurs=1`.

Output Fields

- Regular tables: The **sys_id** field and the display value of the target table are returned.
- Import set tables: The **sys_id** of the import set row, the name of the transformed target table (**table**), the **display_name** for the transformed target table, the **display_value** of the transformed target row, and a **status**

field, which can contain **inserted**, **updated**, or **error**. There can be an optional **status_message** field or an **error_message** field value when *status=error*. When an insert did not cause a target row to be transformed, e.g. skipped because a key value is not specified, the **sys_id** field will contain the sys_id of the import set row rather than the targeted transform table.

- Import set tables with multiple transforms: The response from this type of insert will contain multiple sets of fields from the regular import set table insert wrapped in a multiInsertResponse parent element. Each set will contain a **map** field, showing which transform map created the response.

Sample SOAP Messages

The following example shows an insert that specifies the short description only:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope xmlns:tns="http://www.service-now.com/incident"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:m="http://www.service-now.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <insert xmlns="http://www.service-now.com">
            <short_description xsi:type="xsd:string">This is a test</short_description>
        </insert>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The resulting response looks like this:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:m="http://www.service-now.com"
  xmlns:tns="http://www.service-now.com/incident"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <SOAP-ENV:Body>
        <insertResponse xmlns="http://www.service-now.com">
            <sys_id>6b06494fc611227d00b5f87caf618831</sys_id>
        </insertResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

For language-specific **insert** examples, see:

- Perl SOAP::Lite
- Java Apache Axis2
- Microsoft .NET
- Python

insertMultiple

Creates multiple new records for the table targeted in the URL.

Input Fields

The **insertMultiple** element may contain 1 or more **record** tags that contains all fields from the targeted table, excluding system fields. Limit the number of records inserted in a single operation to no more than 200. You can gradually increase this number with subsequent exports if the increase does not negatively impact instance performance.

Output Fields

The insertMultipleResponse tag followed by 1 or more record tags that contains:

- Regular tables: The **sys_id** field and the display value of the target table are returned.
- Import set tables: The **sys_id** of the import set row, the name of the transformed target table (**table**), the **display_name** for the transformed target table, the **display_value** of the transformed target row, and a **status** field, which can contain **inserted**, **updated**, or **error**. There can be an optional **status_message** field or an **error_message** field value when *status=error*. When an insert did not cause a target row to be transformed, e.g. skipped because a key value is not specified, the **sys_id** field will contain the sys_id of the import set row rather than the targeted transform table.
- Import set tables with multiple transforms: The response from this type of insert will contain multiple sets of fields from the regular import set table insert wrapped in a multiInsertResponse parent element. Each set will contain a **map** field, showing which transform map created the response.

Sample SOAP Messages - regular table

The following example shows an insert that specifies the short description only:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inc="http://www.service-now.com/incident">

   <soapenv:Header/>

   <soapenv:Body>

      <inc:insertMultiple>

         <record>

            <short_description>this is test 1</short_description>

         </record>

         <record>

            <short_description>this is test 2</short_description>

         </record>

         <record>

            <short_description>this is test 3</short_description>

         </record>

      </inc:insertMultiple>

   </soapenv:Body>

</soapenv:Envelope>
```

The resulting response looks like this:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inc="http://www.service-now.com/incident">

   <soapenv:Header/>

   <soapenv:Body>

      <insertMultipleResponse>
```

```
        <insertResponse>

           <sys_id>168160ad4a36231200a89091281dc803</sys_id>

           <number>INC0055180</number>

        </insertResponse>

        <insertResponse>

           <sys_id>1681622e4a36231200a8909115e5c388</sys_id>

           <number>INC0055181</number>

        </insertResponse>

        <insertResponse>

           <sys_id>1681626e4a36231200a89091fa3c0aa8</sys_id>

           <number>INC0055182</number>

        </insertResponse>

     </insertMultipleResponse>

  </soapenv:Body>

</soapenv:Envelope>
```

Sample SOAP Messages - import set table

The following example shows an insert that specifies the short description only:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:imp="http://www.service-now.com/imp_notification">

  <soapenv:Header/>

  <soapenv:Body>

     <imp:insertMultiple>:-->

        <imp:record>

           <imp:message>one</imp:message>

           <imp:uuid>a</imp:uuid>

        </imp:record>

        <imp:record>

           <imp:message>two</imp:message>

           <imp:uuid>b</imp:uuid>

        </imp:record>

        <imp:record>

           <imp:message>three</imp:message>

           <imp:uuid>c</imp:uuid>

        </imp:record>

     </imp:insertMultiple>

  </soapenv:Body>

</soapenv:Envelope>
```

The resulting response looks like this:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:imp="http://www.service-now.com/imp_notification">

  <soapenv:Header/>

  <soapenv:Body>

     <insertMultipleResponse>

        <insertResponse>

           <sys_id>1296b3ab0a0a0b5b73e966fbfab7acde</sys_id>

           <table>incident</table>

           <display_name>number</display_name>
```

```
            <display_value>INC0010033</display_value>
            <status>ignored</status>
            <status_message>No field values changed</status_message>
         </insertResponse>
         <insertResponse>
            <sys_id>1296b48e0a0a0b5b62513bb5974a7d96</sys_id>
            <table>incident</table>
            <display_name>number</display_name>
            <display_value>INC0010034</display_value>
            <status>ignored</status>
            <status_message>No field values changed</status_message>
         </insertResponse>
         <insertResponse>
            <sys_id>1296b58b0a0a0b5b468f534659538b9a</sys_id>
            <table>incident</table>
            <display_name>number</display_name>
            <display_value>INC0010035</display_value>
            <status>ignored</status>
            <status_message>No field values changed</status_message>
         </insertResponse>
      </insertMultipleResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

update

Updates an existing record in the targeted table in the URL, identified by the mandatory **sys_id** field.

Input Fields

All fields from the targeted table, excluding system fields, which will be used for updating the existing record. The **sys_id** field is used to locate the existing record.

Output Fields

Returns the **sys_id** of the record that was updated.

Sample SOAP Messages

Sample SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inc="http://www.service-now.com/incident">
   <soapenv:Header/>
   <soapenv:Body>
      <inc:update>
         <sys_id>46e18c0fa9fe19810066a0083f76bd56</sys_id>
         <short_description>this is updated</short_description>
      </inc:update>
   </soapenv:Body>
</soapenv:Envelope>
```

Sample SOAP response

```
<soapenv:Envelope xmlns:inc="http://www.service-now.com/incident" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

   <soapenv:Header/>

   <soapenv:Body>

      <updateResponse>

         <sys_id>46e18c0fa9fe19810066a0083f76bd56</sys_id>

      </updateResponse>

   </soapenv:Body>

</soapenv:Envelope>
```

For language-specific **update** examples, see:

- Perl SOAP::Lite
- Java Apache Axis2
- Microsoft .NET
- Python

deleteRecord

Delete a record from the targeted table by supplying its **sys_id**.

Input Fields

An element **<sys_id>** identifying the **sys_id** of the record to be retrieved.

Output Fields

A **<count>** element within the deleteRecordResponse parent element indicating the number of records deleted, this will always equal to "1" for deleteRecord.

Sample SOAP Messages

Sample SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inc="http://www.service-now.com/incident">

   <soapenv:Header/>

   <soapenv:Body>

      <inc:deleteRecord>

         <sys_id>46e18c0fa9fe19810066a0083f76bd56</sys_id>

      </inc:deleteRecord>

   </soapenv:Body>

</soapenv:Envelope>
```

Sample SOAP response

```
<soapenv:Envelope xmlns:inc="http://www.service-now.com/incident" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

   <soapenv:Header/>

   <soapenv:Body>

      <deleteRecordResponse>

         <count>1</count>

      </deleteRecordResponse>

   </soapenv:Body>

</soapenv:Envelope>
```

For language-specific **deleteRecord** examples, see:

- Perl SOAP::Lite
- Java Apache Axis2
- Microsoft .NET
- Python

deleteMultiple

Delete multiple records from the targeted table by example values.

Input Fields

All fields from the targeted table, including system fields, are used in query-by-example (QBE) fashion to locate records to be deleted. Query example fields can have special prefixes to constrain the search function.

Output Fields

A **<count>** element within the deleteRecordResponse parent element indicating the number of records deleted

Sample SOAP Messages

Sample SOAP request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:inc="http://www.service-now.com/incident">

   <soapenv:Header/>

   <soapenv:Body>

      <inc:deleteMultiple>

         <category>hardware</category>

      </inc:deleteMultiple>

   </soapenv:Body>

</soapenv:Envelope>
```

Sample SOAP response

```
<soapenv:Envelope xmlns:inc="http://www.service-now.com/incident" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

   <soapenv:Header/>

   <soapenv:Body>

      <deleteMultipleResponse>

         <count>6</count>

      </deleteMultipleResponse>

   </soapenv:Body>

</soapenv:Envelope>
```

For language-specific **deleteMultiple** examples, see:

- Perl SOAP::Lite
- Java Apache Axis2
- Microsoft .NET
- Python

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/integrate/web-services-apis/reference/r_DirectWebServiceAPIFunctions.html

Scripted Web Services

Note: *This article applies to Fuji and earlier releases. For more current information, see Scripted SOAP Web Services* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.'**

Overview

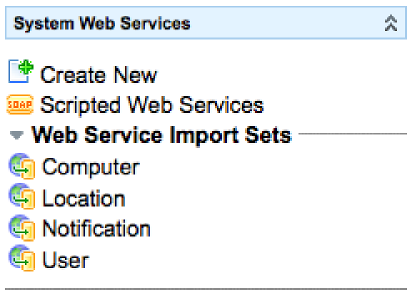
Scripted Web Services allow a ServiceNow administrator to create new web services that are not addressed by the system. You can define input and output parameters for the web service and use JavaScript to perform operations. Though this feature is very powerful, use Direct Web Services or Web Service import sets instead whenever possible since they are simpler to implement and maintain.

Security

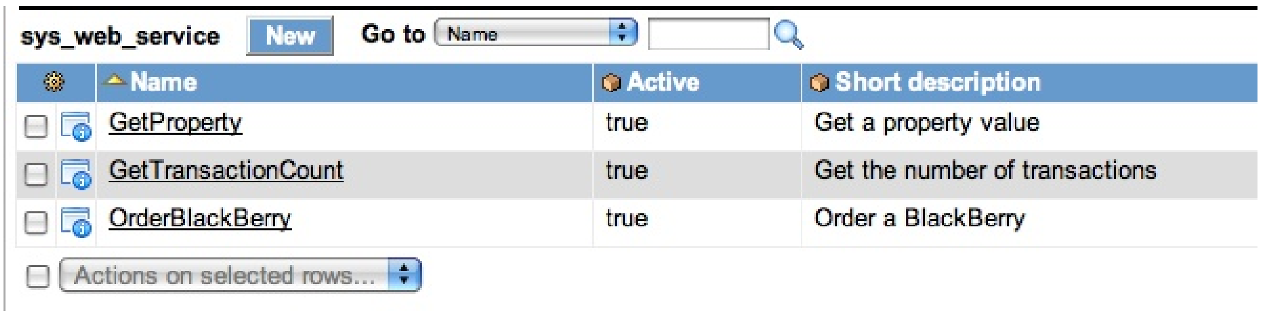
When strict security is enforced on a system, the HTTP authenticated user must have the **soap_script** role to execute the scripted web service.

Creating a new Web Service

When the Web Services Provider - Scripted plugin is activated, a new module **Scripted Web Services** is available under the **System Web Services** application.



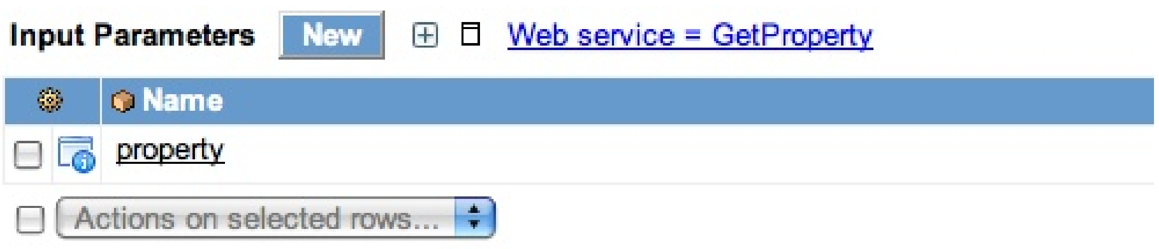
Click the module to display a list of example scripted Web Services.



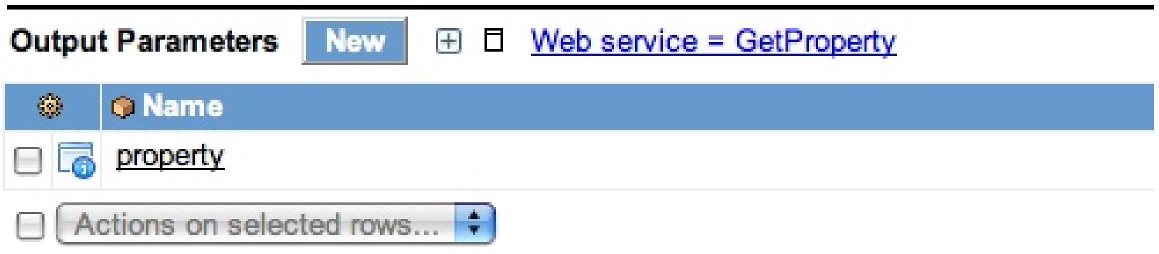
Name	Active	Short description
GetProperty	true	Get a property value
GetTransactionCount	true	Get the number of transactions
OrderBlackBerry	true	Order a BlackBerry

Example 1: Retrieving a System Property

The first step is to define the incoming and return parameters. This is done by adding an entry to the **Input Parameters** and **Output Parameters**. These parameters are used to construct and present a meaningful WSDL, and they do not add to the functionality of processing the actual Web Service itself.



Input Parameters	Web service = GetProperty
Name	property



Output Parameters	Web service = GetProperty
Name	property

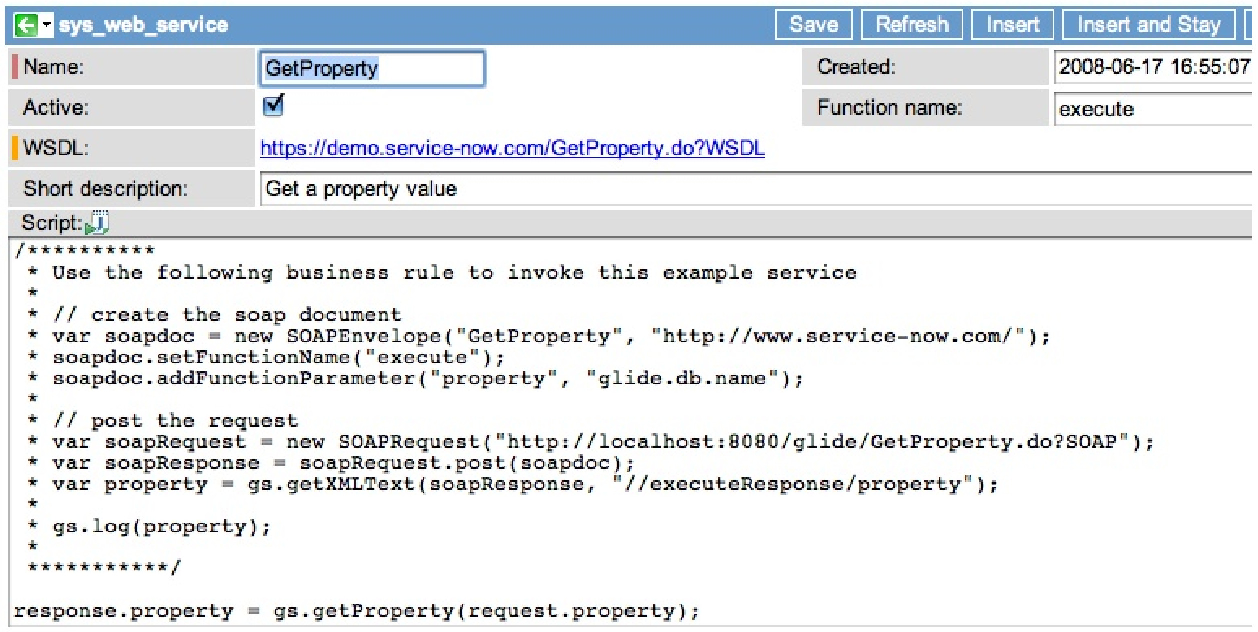
The parameters are referenced in the script of the Web Service. Any of the input parameters are retrieved using the following syntax:

```
var a= request.property;
```

The output parameters are set by using the following syntax:

```
response.property = "ABC";
```

The following example demonstrates how to retrieve a system property and return it as part of the SOAP response. The example shows how to create a custom scripted Web Service to do something specific that the base ServiceNow system direct Web Services cannot.



Name:	GetProperty	Created:	2008-06-17 16:55:07
Active:	✓	Function name:	execute
WSDL:	https://demo.service-now.com/GetProperty.do?WSDL		
Short description:	Get a property value		

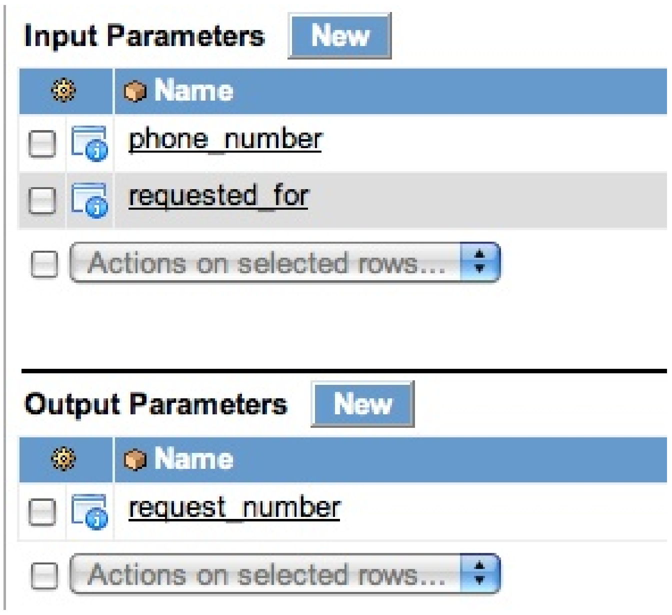
Script:

```
/***********
 * Use the following business rule to invoke this example service
 *
 * // create the soap document
 * var soapdoc = new SOAPEnvelope("GetProperty", "http://www.service-now.com/");
 * soapdoc.setFunctionName("execute");
 * soapdoc.addFunctionParameter("property", "glide.db.name");
 *
 * // post the request
 * var soapRequest = new SOAPRequest("http://localhost:8080/glide/GetProperty.do?SOAP");
 * var soapResponse = soapRequest.post(soapdoc);
 * var property = gs.getXMLText(soapResponse, "//executeResponse/property");
 *
 * gs.log(property);
 *
 ***********/

response.property = gs.getProperty(request.property);
```

Example 2: Ordering a Blackberry

Direct Web Services in ServiceNow operate on tables and their data, while the following example shows how to initiate a business solution, such as ordering a Blackberry, by invoking a scripted Web Service. The following input and output parameters will support the Blackberry example:



This script shows how to use the above parameters to add a Blackberry to the service catalog shopping cart and order it. The request number is returned in the **request_number** field of the SOAP response.

```
var cart = new Cart();
var item = cart.addItem('e2132865c0a8016500108d9cee411699');
cart.setVariable(item, 'original', request.phone_number);

// set the requested for
var gr = new GlideRecord("sys_user");
gr.addQuery("user_name", request.requested_for);
gr.query();
if (gr.next()) {
  var cartGR = cart.getCart();
  cartGR.requested_for = gr.sys_id;
  cartGR.update();
}

var rc = cart.placeOrder();
response.request_number = rc.number;
```

Global Variables

To facilitate custom processing of incoming SOAP requests, the following global variables are available in the script context:

1. **soapRequestDocument:** a Java *org.w3c.dom.Document* object representing the incoming SOAP envelope.
2. **soapRequestXML:** a string object representing the incoming SOAP envelope XML.
3. **request:** a Javascript object containing mapped values (mapped to input parameter names) of the incoming SOAP envelope
4. **response:** a Javascript object which allows the script programmer to customize the response values. See Customized Response

Customized Response

To customize and have control over the XML payload of the SOAP response, follow this example:

1. Create a customized XML document using the XMLDocument script include object
2. set its document element to the variable **response.soapResponseElement** in a scripted web service

For example, the following scripted web service script:

```
var xmldoc = new XMLDocument("<myResponse></myResponse>");

xmldoc.createElement("element_one", "test"); // creates the new element at the document element level if setCurrent is never called

xmldoc.createElement("element_two", "new2 value"); // calling without a value will create a new element by itself


var el = xmldoc.createElement("element_three");

xmldoc.setCurrent(el); // this is now the parent of any new elements created subsequently using createElement()

xmldoc.createElement("newChild", "test child element");


response.soapResponseElement = xmldoc.getDocumentElement();
```

Is used to accept the following request:

```
<soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:tes="http://www.service-now.com/TestCustomResponse">
   <soapenv:Header/>
   <soapenv:Body>
      <tes:execute/>
   </soapenv:Body>
</soapenv:Envelope>
```

Which will respond with the following SOAP response:

```
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:tes="http://www.service-now.com/TestCustomResponse">
   <soapenv:Header/>
   <soapenv:Body>
      <myResponse>
         <element_one>test</element_one>
         <element_two>new2 value</element_two>
         <element_three>
```

```
            <newChild>test child element</newChild>
         </element_three>
      </myResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

WSDL support will need to be created externally. The SOAP endpoint will need to be referred back to the scripted web service in question.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/integrate/inbound-soap/concept/c_ScriptedWebServices.html

AttachmentCreator SOAP Web Service

Note: *This article applies to Fuji. For more current information, see AttachmentCreator SOAP Web Service* [1] *at* http://docs.servicenow.com The Wiki page is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Overview

Attaching documents to records in ServiceNow is achieved by sending a SOAP message targeting the **ecc_queue** table. For example, using the following URL or target end point:

https://instance_name.service-now.com/ecc_queue.do?SOAP

The following fields in **ecc_queue** must be set to trigger the creation of the attachment.

Field Name	Description	Value
agent	The name of the agent sending in the request, this can be any value since it is not used for processing.	AttachmentCreator
topic	The topic of the queue record, this value must be set to "AttachmentCreator" to trigger the attachment creation	AttachmentCreator
name	This field must contain a ":" delimited value of the file name, and its content-type	file_name.xls:application/vnd.ms-excel
source	This field must contain a ":" delimited value of the target table and its **sys_id**	incident:dd90c5d70a0a0b39000aac5aee704ae8
payload	This field must contain the Base 64 [2] encoded string representing the object to be attached	the base 64 encoded string

Sending in the values described in the table above will attach an Excel file to the **incident** table for the record located by the sys_id "dd90c5d70a0a0b39000aac5aee704ae8"

Security

Like all other HTTP based web services available on the platform, the AttachementCreator SOAP web service is required to authenticate using basic authentication [3] by default. The user ID that is used for authentication will be subjected to access control in the same way as an interactive user.

To create attachments, the SOAP user must have any roles required to create Attachment [sys_attachment] records, as well as the soap_create role, and any roles required to read and write records on the target table, such as the itil role to add attachments to incident records. By default there is no single role allowing you to add attachments. You can create a role to explicitly allow adding attachments, then assign this role to the SOAP user.

File Type Security

You can control what file types users can attach by setting the `glide.attachment.extensions` and `glide.security.file.mime_type.validation` properties. These properties apply to the AttachmentCreator web service starting with Fuji Patch 10.

For these properties to apply to the AttachmentCreator web service, the property `glide.attachment.enforce_security_validation` must be set to true. This property is true by default.

Example SOAP Message

The following is an example of a SOAP message that would take a text file "john1.txt" of mime-type: text/plain and attach it to an Incident with a GUID of: e6eed6950a0a3c59006f32c8e3ff3cf9. Note the payload is the base64 encoding of the file itself.

```
<?xml version="1.0" encoding="UTF-8"?>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ecc="http://www.service-now.com/ecc_queue">

    <soapenv:Header />

    <soapenv:Body>

        <ecc:insert>

            <agent>AttachmentCreator</agent>

            <topic>AttachmentCreator</topic>

            <name>john1.txt:text/plain</name>

            <source>incident:e6eed6950a0a3c59006f32c8e3ff3cf9</source>

            <payload>SSB3b25kZXIgaWYgc2hlIGtub3ducyB3aGF0IHNoZSdzIGRvaW5nIG5vdy4K</payload>

        </ecc:insert>

    </soapenv:Body>

</soapenv:Envelope>
```

Example Node.js Script

The following example Node.js [4] script adds an attachment to an incident record. Run this script from a client computer, not a ServiceNow instance.

```
/**
 *
 * Node.js to ServiceNow attachment upload via SOAP
 *
 * Andrew Venables andrew.venables@servicenow.com
 * July 2014
 *
 * Version 1.0
 *
 */

var soap = require('soap'), // https://github.com/vpulim/node-soap
      mime = require('mime'), // https://github.com/broofa/node-mime
      fs = require("fs");

var WSDL_FILENAME = 'ecc_queue.xml'; // Goto
https://instancename.service-now.com/ecc_queue.do?WSDL and save a copy
of the WSDL locally for simplicity
var DIRECTORY_CONTAINING_FILES =
'/Users/andrew.venables/Documents/Uploads'; // Local path to the
directory containing all the files we want to upload
var USERNAME = 'andy.venables'; // An admin user account on the
instance
var PASSWORD = 'MY_PASSWORD'; // Password for above account
var TARGET_TABLE = 'incident'; // Target table to attach the files to
var TARGET_SYS_ID = '9d385017c611228701d22104cc95c371'; // Target
record / sys_id to attach the files to. OOTB INC0000002

var files_to_upload; // Global that will contain our list of files to
be uploaded
var pos = 0; // Global pointer for our position in the files_to_upload
list

// Create a SOAP client to use to post to the instance
soap.createClient(WSDL_FILENAME, function(err, client) { // Node uses
callbacks
      if (err) console.error(err);

      // Set the username and password
      client.setSecurity(new soap.BasicAuthSecurity(USERNAME,
PASSWORD));

      // Read all the files in our source directory, will include . and
 ..
      files_to_upload = fs.readdirSync(DIRECTORY_CONTAINING_FILES);

      console.log('Files to upload: ' + files_to_upload.length + '\n');

      // Start iterating through the list of files to upload
      next(client);

});
```

```
// Process the next file in the files_to_upload array
// This is a neat way of dealing with Node and its expectation of
callbacks
function next(client) {

      // Check we haven't reached the end
      if (pos >= files_to_upload.length) return;

      // Get the next file to upload
      var file_name = files_to_upload[pos];

      // Increment the pointer to the next file
      pos++;

      console.log(pos + '/'+ files_to_upload.length+ ' - Uploading
file: ' + file_name);

      // A blank file is the end of the list
      if (file_name == '') return;

      // Skip to the next file as this one is invalid
      if (file_name == '.' || file_name == '..' ||
file_name.indexOf('.') == 0)
            next(client);

      // Get the file type using an module called mime
      var file_type = mime.lookup(file_name);
      console.log('   of type: ' + file_type);

      var file_payload;
      // Load the file into a buffer
      fs.readFile(DIRECTORY_CONTAINING_FILES + '/' + file_name,
function(err, the_data) {
            if (err) console.error(err);

            // Encode the buffer to base64
            file_payload = new Buffer(the_data,
'binary').toString('base64');

            // Set the parameters before we call the Web Service
            var parameters = {
                  'agent':       'AttachmentCreator',
              'topic':       'AttachmentCreator',
              'name':       file_name+':'+file_type,
              'source':       TARGET_TABLE+':'+TARGET_SYS_ID,
              'payload':       file_payload
```

```
                };

                console.log('       sending...')
                // Make the Web Service call, remember node likes callbacks
                client.insert(parameters, function(err, result) {
                      if (err) console.error(err);

                      console.log(result);

                      // This file is done, next!
                      next(client);
                });
        });
}
```

Example Perl Script

The following example inserts multiple attachments into the ecc_queue table. You can specify the directory to upload using the file_path argument. All files in that directory and all subdirectories are uploaded.

This script requires a before business rule on inserts to the ecc_queue table to avoid having to make multiple calls for each attachment import.

```
use warnings;
use File::Basename;
use File::Find qw(finddepth);
use ServiceNow::SOAP;
use MIME::Base64;
use MIME::Types;

# Handle command line arguments to get the destination folder
$numArgs = $#ARGV + 1;
if( $numArgs != 6)
{
      print "ERROR: Insufficient Command Line Arguements.\n\n";
      print_usage();
      exit -1;
}

$SNC_HOST = $ARGV[0];
$username = $ARGV[1];
$password = $ARGV[2];
$file_path = $ARGV[3];
$target_table = $ARGV[4];
$correlation_field = $ARGV[5];

my $base64;
my $buf;
my $mimetype;
```

```
#SOAP CONFIG stuff
my $CONFIG = ServiceNow($SNC_HOST, $username, $password);
my $sn_table = $CONFIG->table('ecc_queue');

# traverse the specified file directory and stores files in the @files
array
my @files;
finddepth(sub {
     return if($_ eq '.' || $_ eq '..');
     push @files, $File::Find::name;
}, $file_path);

# process all the files in our array
foreach $file (@files) {
      $full_path = "$file";
      print "\nFull Path: ". $file;

      #Test the file type
      if( -d $file)
      {
            print "\nWe don't send directories only files: ".
$full_path;
            next;
      }

      open( FILE, $file) or die "$!";
      binmode FILE; #preserves file formatting on Windows
      my $file_contents = "";
      while ( read( FILE, $buf, 60 * 57 ) ) {
            $file_contents .= encode_base64($buf);
      }

      my $filename = basename($full_path);
      my $mimetype = MIME::Types -> new;


  #Get Target Record by Correlation Field - $number should retrieve the
 matching correlation value from the filename
  #  In this case, the correlation value is the first 5 characters of
the filename
  my $number = substr $filename, 0, 5;


       my $sysid = $sn_table->insert(
      agent => 'AttachmentCreator',
      topic => 'AttachmentCreator',
      name => $filename.':'.$mimetype->mimeTypeOf($filename),
      source => $target_table.':ka'.$number.':'.$correlation_field,
      payload => $file_contents
  );



  # print results or faults
  print " Created Attachment record: " . $sysid . "\n";
}

# subroutine to print out the USAGE of this script
sub print_usage {
      print "===============================\n";
      print "sn_attachment_import.pl\n";
      print "===============================\n";
      print "\nUSAGE:\n\n";
      print "sn_attachment_import.pl <SN instanceUrl> <SN username> <SN password> <root_data_path>\n";
      print "\n...Where:\n";
      print "\t<SN instance name>: The name of the target ServiceNow instance, i.e.
instance (for instance.service-
now.com)\n";
      print "\t<username>: ServiceNow user name with rights to add records to the
 target table\n";
      print "\t<password>: ServiceNow user password\n";
      print "\t<root_data_path>: The full path and name of the directory that contains
the attachment data.\n";
      print "\t<target_table>: The ServiceNow table the attachment should be
associated with.\n";
      print "\t<correlation_field>: The ServiceNow field on the target table that matches
the correlation value of the attachments\n";
      print "\nEXAMPLE:\n\n";
      print "sn_attachment_import.pl myinstancename myusername
mypassword ./attachment_root_folder sn_table_name correlation_field\n";
      print "\n-------------------------------\n\n";
}
```

Before Business Rule

Create the following before business rule to manage inserting multiple attachments.

Field	Value
Table	ecc_queue
When	Before
Insert	true
Condition	current.source.indexOf(':ka') > -1
Script	

```
onBefore(current, previous) {
   //This function will be automatically called when this rule is
processed.
      var log = [];
      log.push('Starting Attachment Correlation\nAgent: '+
current.agent);
      var correlation_field =
current.source.split(':ka')[1].split(':')[1]+'';
      var correlation_id =
current.source.split(':ka')[1].split(':')[0]+'';
      var target_table = current.source.split(':ka')[0]+'';
      log.push('Field: '+ correlation_field +'\nID: '+ correlation_id
+'\nTable: '+ target_table);
      var gr = new GlideRecord(target_table);
      if(correlation_field != '' && correlation_id != ''){
            gr.addQuery(correlation_field, correlation_id);
            gr.query();
            gr.next();
            current.source = target_table + ':' + gr.sys_id;
            log.push('Source: '+ current.source);
      }
      gs.log(log.join('\n'));
}
```

FAQ

- How many attachments can you send in one message
  - Only one attachment per message
- Is there size limit on the attachment in the message
  - Attachments are limited to a maximum of 5MB.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/integrate/inbound-soap/reference/r_AttachmentCreatorSOAPWebService.html
[2] http://en.wikipedia.org/wiki/Base64
[3] http://www.w3.org/Protocols/HTTP/1.0/draft-ietf-http-spec.html#BasicAA
[4] http://nodejs.org/

ECC Queue Retry Policy

Overview

Define retry policies for outbound Web Services that are executed via the **ECC Queue** table. Retry policies specify a matching error condition for ECC Queue *input* records that are a result or response of an *output* queue record, the interval for retry, and the maximum number of retries. Because it matches on the *input* queue record, the retry policies only work when an *input* ECC Queue record is expected, and therefore requires that the outbound messages are queued on the ECC Queue table as well. Advanced matching criteria may be specified using script.

Activating the Plugin

Click the plus to expand instructions for requesting a plugin.

1. Navigate to [HI [1]].
2. Click **Service Catalog**.
3. Click **Request Plugin Activation**.
   - [Required] In **Target Instance**, select the instance on which to activate the plugin.
   - [Required] In **Plugin Name**, enter the name of the plugin to activate.
   - [Optional] In **Date and time would you like the plugin to be enabled?**, specify a date and time at least 12 hours in the future. Leave this field empty if you want the plugin activated as soon as possible.

   Note: Plugins are generally activated during business hours in the Pacific time zone, but can be scheduled for a different time with advance notice.

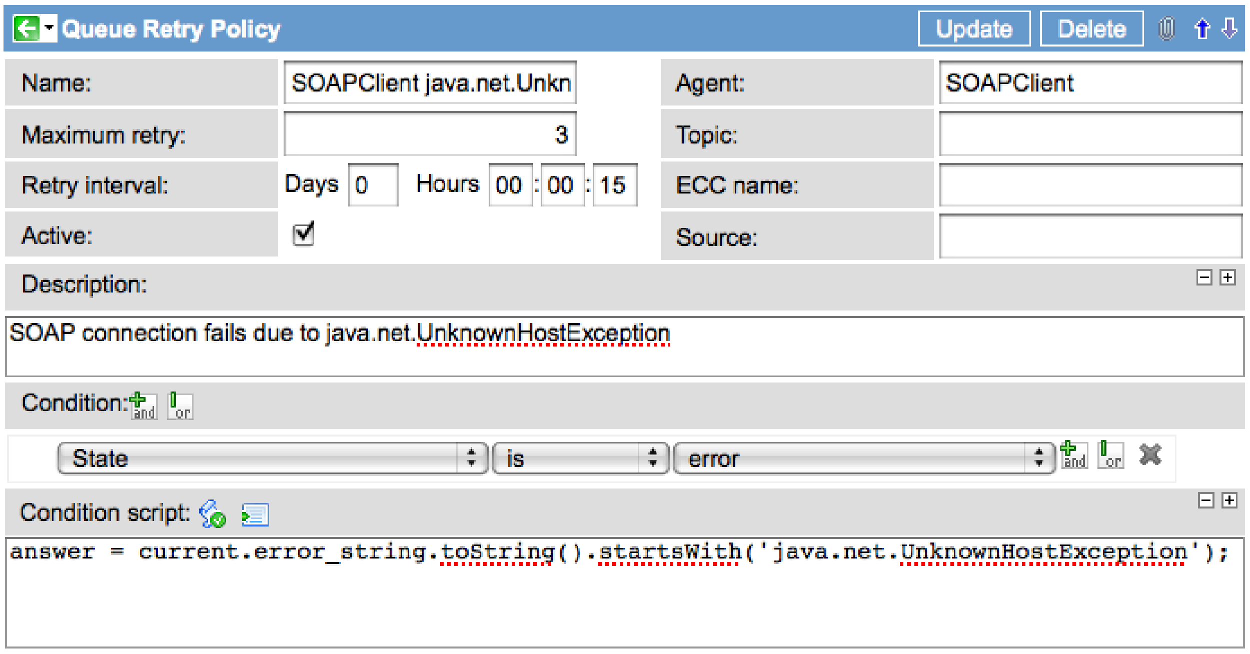
   - [Optional] In **Reason/Comments**, add any information that would be helpful for the ServiceNow technical support engineer activating the plugin.
4. Click **Submit**.

The plugin installs the following new modules under the **ECC** application:

- Queue Retry Policy
- Queue Retry Activity

Retry Policy

The following is a default retry policy that defines the matching criteria when an *UnknownHostException* response is received during a **SOAPClient** (outbound SOAP message invocation) call.



Name:	SOAPClient java.net.Unkn	Agent:	SOAPClient
Maximum retry:	3	Topic:	
Retry interval:	Days 0 Hours 00 : 00 : 15	ECC name:	
Active:	✓	Source:	

Description:

SOAP connection fails due to java.net.UnknownHostException

Condition:

State | is | error

Condition script:

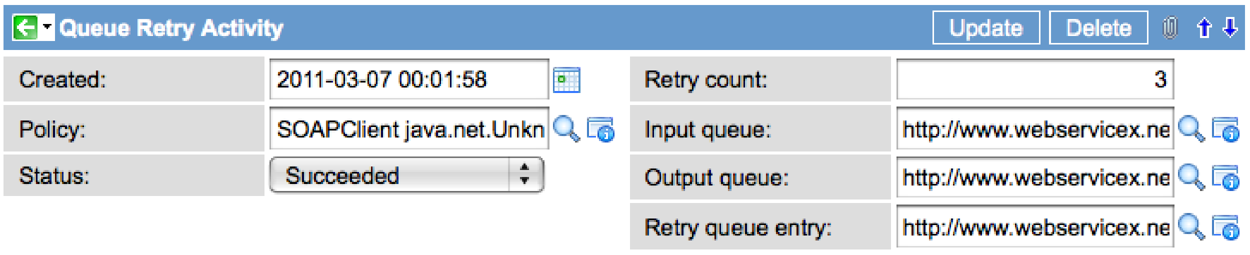
```
answer = current.error_string.toString().startsWith('java.net.UnknownHostException');
```

The policy will only match if the **Agent**, **Topic**, **ECC Name** (matches the **Name** field), and **Source** fields match that of an *input* ECC queue record. Additionally, the **Condition** and **Condition script** criteria will also have to match "State == error" and the **error_string** field contains the text *java.net.UnknownHostException*.

When these conditions are met, and an *input* record is matched, it will retry the originating *output* queue record after 15 seconds, and for a maximum of 3 times.

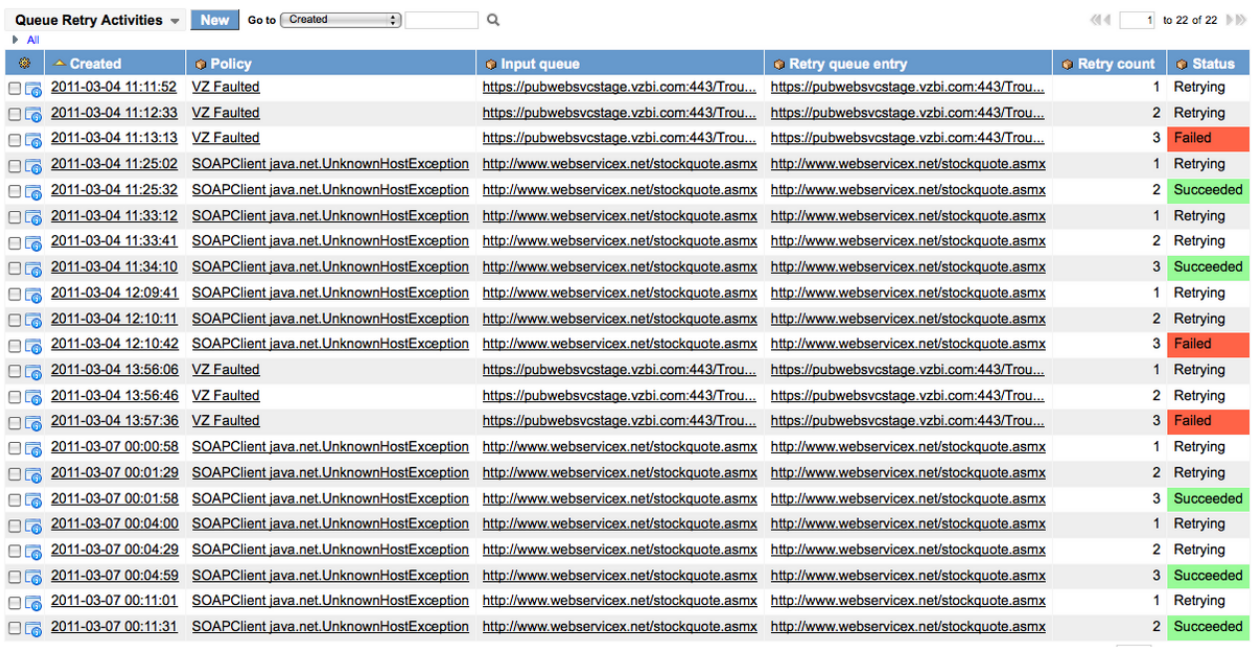
Retry Activity

The retry activity records document each attempt to retry the *output* queue record. When the current policy is being retried, the **State** of the activity will indicate **Retrying**. When all retries are exhausted, e.g. the maximum number of retries have occurred and the *output* queue still failed, the **State** field will indicate a **Failed** state. Otherwise, at anytime during the retry, if the request then becomes successful, the **State** will indicate **Succeeded**.



Created:	2011-03-07 00:01:58	Retry count:	3
Policy:	SOAPClient java.net.Unkn	Input queue:	http://www.webservicex.ne
Status:	Succeeded	Output queue:	http://www.webservicex.ne
		Retry queue entry:	http://www.webservicex.ne

You may also display the current list of retry activities and their states by selecting the **Queue Retry Activity** module.


Queue Retry Activities New Go to Created

All

1 to 22 of 22

Created	Policy	Input queue	Retry queue entry	Retry count	Status
2011-03-04 11:11:52	VZ Faulted	https://pubwebsvcstage.vzbi.com:443/Trou...	https://pubwebsvcstage.vzbi.com:443/Trou...	1	Retrying
2011-03-04 11:12:33	VZ Faulted	https://pubwebsvcstage.vzbi.com:443/Trou...	https://pubwebsvcstage.vzbi.com:443/Trou...	2	Retrying
2011-03-04 11:13:13	VZ Faulted	https://pubwebsvcstage.vzbi.com:443/Trou...	https://pubwebsvcstage.vzbi.com:443/Trou...	3	Failed
2011-03-04 11:25:02	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	1	Retrying
2011-03-04 11:25:32	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	2	Succeeded
2011-03-04 11:33:12	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	1	Retrying
2011-03-04 11:33:41	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	2	Retrying
2011-03-04 11:34:10	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	3	Succeeded
2011-03-04 12:09:41	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	1	Retrying
2011-03-04 12:10:11	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	2	Retrying
2011-03-04 12:10:42	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	3	Failed
2011-03-04 13:56:06	VZ Faulted	https://pubwebsvcstage.vzbi.com:443/Trou...	https://pubwebsvcstage.vzbi.com:443/Trou...	1	Retrying
2011-03-04 13:56:46	VZ Faulted	https://pubwebsvcstage.vzbi.com:443/Trou...	https://pubwebsvcstage.vzbi.com:443/Trou...	2	Retrying
2011-03-04 13:57:36	VZ Faulted	https://pubwebsvcstage.vzbi.com:443/Trou...	https://pubwebsvcstage.vzbi.com:443/Trou...	3	Failed
2011-03-07 00:00:58	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	1	Retrying
2011-03-07 00:01:29	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	2	Retrying
2011-03-07 00:01:58	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	3	Succeeded
2011-03-07 00:04:00	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	1	Retrying
2011-03-07 00:04:29	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	2	Retrying
2011-03-07 00:04:59	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	3	Succeeded
2011-03-07 00:11:01	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	1	Retrying
2011-03-07 00:11:31	SOAPClient java.net.UnknownHostException	http://www.webservicex.net/stockquote.asmx	http://www.webservicex.net/stockquote.asmx	2	Succeeded

References

[1] http://hi.service-now.com

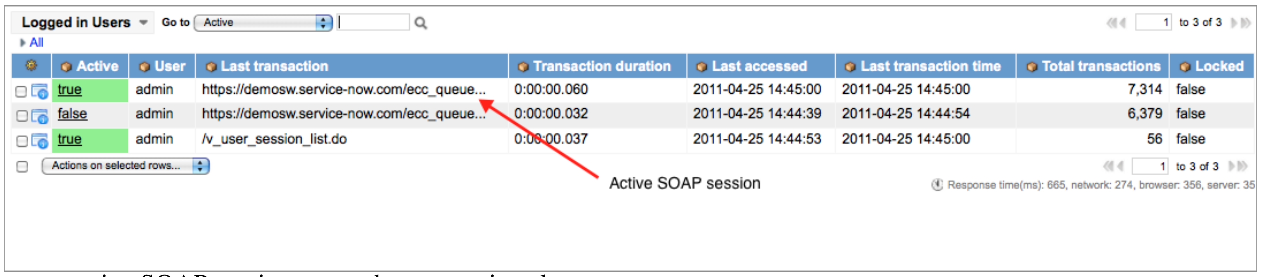
SOAP Session Management and Reporting

Overview

A SOAP session is a Glide session established with a ServiceNow instance by any external SOAP client, such as a customer-developed Web Services client application, a ServiceNow MID Server, or the ServiceNow ODBC driver. SOAP sessions are reported in the **Logged-in Users** (active user sessions) report. SOAP sessions are identified by their **?SOAP** URLs. Idle SOAP sessions are timed out and removed if the **glide.soap.invalidate_session_timeout** interval expires without a new request arriving. The **glide.soap.invalidate_session_timeout** Glide property is an optional property whose value defaults to 60 seconds.

Viewing a SOAP Session Report

1. Navigate to *User Administration > Logged in users*.


Logged in Users Go to Active

All

1 to 3 of 3

Active	User	Last transaction	Transaction duration	Last accessed	Last transaction time	Total transactions	Locked
true	admin	https://demosw.service-now.com/ecc_queue...	0:00:00.060	2011-04-25 14:45:00	2011-04-25 14:45:00	7,314	false
false	admin	https://demosw.service-now.com/ecc_queue...	0:00:00.032	2011-04-25 14:44:39	2011-04-25 14:44:54	6,379	false
true	admin	/v_user_session_list.do	0:00:00.037	2011-04-25 14:44:53	2011-04-25 14:45:00	56	false

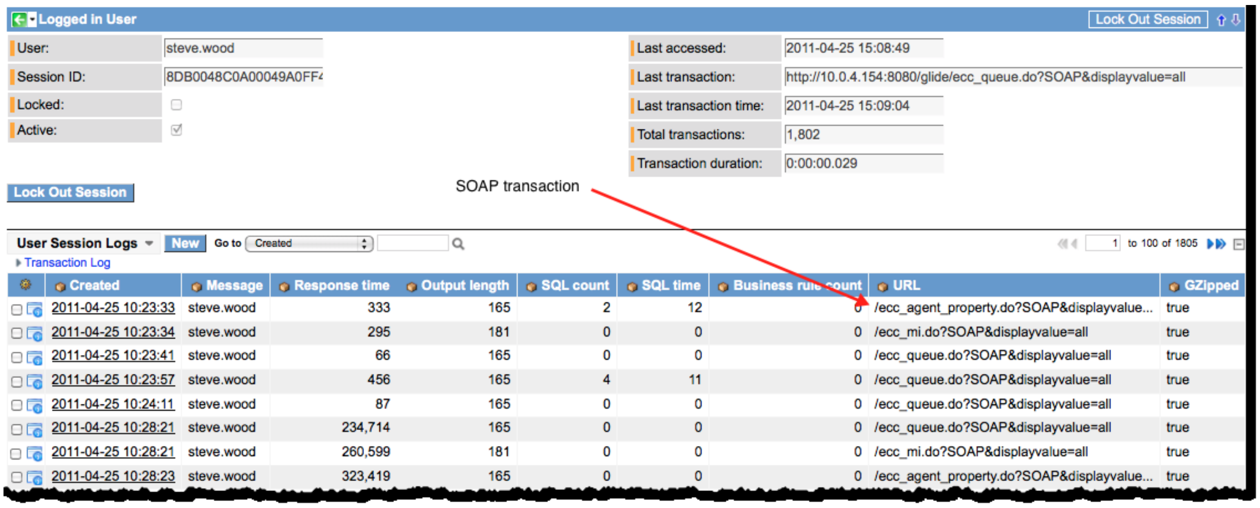
Actions on selected rows...

1 to 3 of 3

Active SOAP session

Response time(ms): 665, network: 274, browser: 356, server: 35

2. Open an active SOAP session to see the transactions log.



User:	steve.wood	Last accessed:	2011-04-25 15:08:49
Session ID:	8DB0048C0A00049A0FF4	Last transaction:	http://10.0.4.154:8080/glide/ecc_queue.do?SOAP&displayvalue=all
Locked:	☐	Last transaction time:	2011-04-25 15:09:04
Active:	✓	Total transactions:	1,802
		Transaction duration:	0:00:00.029

Created	Message	Response time	Output length	SQL count	SQL time	Business rule count	URL	GZipped
2011-04-25 10:23:33	steve.wood	333	165	2	12	0	/ecc_agent_property.do?SOAP&displayvalue...	true
2011-04-25 10:23:34	steve.wood	295	181	0	0	0	/ecc_mi.do?SOAP&displayvalue=all	true
2011-04-25 10:23:41	steve.wood	66	165	0	0	0	/ecc_queue.do?SOAP&displayvalue=all	true
2011-04-25 10:23:57	steve.wood	456	165	4	11	0	/ecc_queue.do?SOAP&displayvalue=all	true
2011-04-25 10:24:11	steve.wood	87	165	0	0	0	/ecc_queue.do?SOAP&displayvalue=all	true
2011-04-25 10:28:21	steve.wood	234,714	165	0	0	0	/ecc_queue.do?SOAP&displayvalue=all	true
2011-04-25 10:28:21	steve.wood	260,599	181	0	0	0	/ecc_mi.do?SOAP&displayvalue=all	true
2011-04-25 10:28:23	steve.wood	323,419	165	0	0	0	/ecc_agent_property.do?SOAP&displayvalue...	true

The SOAP session is marked as inactive within 60 seconds of the last transaction.

Long-Running SOAP Request Support

Note: *This article applies to Fuji and earlier releases. For more current information, see Long-Running SOAP Request Support* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Overview

The ServiceNow platform supports long-running SOAP requests by preventing socket timeouts due to inactivity of the network connection while the requests are in process. This functionality improves the efficiency of the ODBC driver when requesting large numbers of records, doing aggregate queries, or using *order by* expressions that require sorting.

By default, ServiceNow provides timeout protection for Web Services clients provided by ServiceNow such as the ODBC driver and the MID Server. You can add time timeout protection to customer developed Web Services with the parameters described below.

Timeout Protection

Web Services clients receive a **307-Temporary Redirect** [2] to keep long sessions alive and prevent a timeout due to socket inactivity. A 307-Temporary Redirect causes Web Services clients which support the status code to repeat their last request to the location specified in the HTTP location header. The value of the location header sent by ServiceNow is the same URL that the Web Services client originally specified. The use of 307-Temporary Redirects is WS-I compliant.

A Web Service request that exceeds the timeout limit (`glide.soap.request_processing_timeout`) can only receive a 307-Temporary Redirect when **all** of these are met:

- The value of `glide.soapprocessor.allow_long_running_threads` is true
- The request includes a `redirectSupported=true` URL parameter
- The Request is session-aware (supports HTTP cookies)

If any of these conditions is not met, the Web Service client receives a 408 Request Timeout error.

Note: *To ensure that applications experience a socket timeout rather than a 408 Request Timeout, set the* `glide_soap_request_processing_timeout` *property to a value larger than the shortest socket timeout setting in effect for the connection between the application and the ServiceNow instance (300 seconds for hosted instances).*

Properties

The following properties control long-running SOAP processes:

Property	Description
glide.http.connection_timeout	Specify the maximum number of milliseconds an outbound HTTP request (such as Web Services) has to finish processing before the connection times out. • **Type:** integer • **Default value:** 100000 (100 seconds) • **Location:** Add to the System Properties [sys_properties] table
glide.http.timeout	(Web Service Consumer Plugin) Specifies the maximum number of milliseconds to wait before an outbound transaction times out. • **Type:** integer • **Default value:** 175000 (175 seconds) • **Location:** Add to the System Properties [sys_properties] table
glide.soap.request_processing_timeout	Specify the maximum number of seconds an inbound SOAP request has to finish processing before the connection times out. If undefined, this property computes a default value from the value of the property `glide.http.timeout` divided by 1000. You might have network infrastructure, such as proxy servers, which implement a shorter timeout. In this case, a socket timeout may occur unless this property is set to a shorter value. In general, you should set this property to a value several seconds less than the shortest socket inactivity timeout in effect anywhere in the network path between the client application and the ServiceNow instance. • **Type:** integer • **Default value:** 60 for new instances starting on the Fuji release. 175 prior to Fuji. • **Location:** System Property [sys_properties] table
glide.soapprocessor.allow_long_running_threads	Enables or disables a 307-Temporary Redirect. The default setting is *true* (enabled).
glide.soapprocessor.max_long_running_threads	Controls the maximum number of long-running SOAP threads which can run at any one time. The default value for this property is determined dynamically based on the number of SOAP semaphores configured. It should not be necessary to change this value.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/integrate/inbound-soap/reference/r_LongRunningSOAPRequestSupport.html
[2] http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/2175ec7d-b2a9-4f2c-821a-b6539b88902e.mspx?mfr=true

Web Services Integrations Best Practices

Overview

These are some of the most important best practices for Web Services integrations.

Best Practices

Choice of Web Services client technology

When choosing amongst alternatives for creating web services clients, keep in mind that while simple scripting alternatives may seem appealing, they can have hidden costs. For example, if the library does not have configuration options for maintaining sessions, or performing preemptive authorization, performance will be terrible. Solving this may then require writing additional code to manage HTTP session cookies or to create custom HTTP headers. The time and effort spent to diagnose these problems and write the additional code typically erases the benefits sought by using the scripting approach. When possible, use a robust Java or .NET SOAP stack, as opposed to a less robust scripting approach. Use Sun/Oracle or Microsoft web services libraries, or a very well known and well supported open-source library such as Axis 2.x. Avoid out-of-date libraries such as Axis 1.4.

Client Connection Setting Recommendations

To ensure acceptable performance, carefully observe the following requirements for the client-side of all Web Services integrations:

- Use HTTP 1.1 and ensure that the client does not accidentally disable Keep-Alive by sending a *Connection: close* header or by setting up an entirely new connection for every request to ServiceNow
- Reuse the connection with ServiceNow for multiple requests
- Ensure that the client is session-aware - that it observes Set-Cookie headers in ServiceNow responses and responds with a Cookie header on subsequent requests.
- Ensure that the client performs preemptive authorization - that it sends a Basic Authorization or OAuth HTTP header on every request. (Does not apply to WS-Security signed requests.)

These are simple objectives to accomplish if you are using the latest integration middleware or SOAP library, such as Java JDK JAX-WS, to write the client. Typically, this is just a matter of setting a few properties. If these steps are overlooked, the following can occur:

- If the TCP connection is not maintained, an SSL handshake occurs on every request, which creates significant delays.
- If the Web Services client is not session-aware, large numbers of new sessions are created in the ServiceNow instance, slowing everything down and wasting memory.
- If the Web Services client does not perform preemptive authorization, every message is sent twice (once without credentials, a second time with credentials, following a challenge response from ServiceNow).

Handling SOAP Message Response

It's recommended that whenever a SOAP Request is executed (i.e. **posted**), that you evaluate if it is necessary to get an instant response back. Consider one of these two options for handling the SOAP Message Response:

1. Asynchronously: If the SOAP Message is not set to use a MID Server, then when calling the **post()** method, you can change this to **post(true)**. This will send the SOAP Request and Response through the ECC Queue. You can then create a sensor (business rule on the ECC Queue) to observe the response and process the SOAP Response asynchronously.

```
var s = new SOAPMessage('TemperatureConvert',
'TempConvertSoap.FahrenheitToCelsius');
s.setParameter('temperatureF', '100');
s.post(true);
```

2. Synchronously: If you're posting the SOAP Message synchronously, then ensure you don't loop through the SOAP Response, as your script will already wait for it and get an almost instant response back. Consider this example:

```
var message = new SOAPMessage('TemperatureConvert',
'TempConvertSoap.FahrenheitToCelsius');
message.setParameter('temperatureF', '100');
message.post();

var k = 1;
var r = message.getResponse();
while(r == null) {
  gs.log("waiting ... " + k + " seconds");
  r = message.getResponse(1000);
  k++;

  if (k > 30) {
    break;
  }
}
```

Because `message` has been posted synchronously, you don't need to create any unnecessary loops to wait for the SOAP response:

```
var message = new SOAPMessage('TemperatureConvert',
'TempConvertSoap.FahrenheitToCelsius');
message.setParameter('temperatureF', '100');
var response = message.post();
```

Remember, since `response` is an XML string, you will need to convert it to an actual XML document so that you can use XPATH operations on it.

SOAP Timeout Value

If you do expect to process large numbers of SOAP records in a single transaction (such as during an ODBC request), set the system property `glide.soap.request_processing_timeout`. Start off with a modest value of 60 (60 seconds) and increase the value by 60 as needed. ServiceNow does not recommend setting the time out value to an arbitrarily large value such as 30000 because of the chance of wasting database resources.

For example, suppose a runaway SOAP query for ODBC runs for hours before someone finally cancels the client connection. Then someone repeats the same runaway SOAP process and again cancels the client when they get no response the second time. With a large timeout value, the instance wastes hours of database resources processing requests for cancelled client transactions. With a smaller timeout value, the instance recovers from the closing of a client connection sooner and wastes less database resources.

Where possible create multiple batches of SOAP requests that complete processing in three minutes or less. If you cannot break up the SOAP request in multiple requests, increment the timeout value by minutes rather than set a large timeout value.

Compression

Enable compression of HTTP messages. This is the default behavior with the latest, robust SOAP libraries, but make sure to verify it. This is done by ensuring that each HTTP request carries an Accept header that includes **gzip**.

Queries

The ServiceNow WSDL defines a number of SOAP operations, including *getKeys* and *getRecords*. Use sequences of one *getKeys* request followed by a *getRecords* request to accomplish queries and retrieval of large numbers of records.

Do not retrieve batches of records in a single request using *getRecords* with an encoded query or using a query-by-example approach. A direct request using *getRecords* with a query can potentially return many megabytes of data and overwhelm the client application. An empty or unqualified *getRecords* request against a table will match all records. See **Direct Web Services** for instructions on how to use these APIs.

Testing

Test new applications carefully. Perform the following:

- **Examine wire-level traffic for anomalies and to verify correct and efficient behavior.** Introduce a reverse proxy at the customer site during testing or troubleshooting so that the client application can be configured to use HTTP instead of HTTPS for testing purposes and to capture unencrypted request/response messages *on the wire*. Also, the client application must be configured to disable compression so that the payload of each request and response is readable instead of compressed. In this scenario, the reverse proxy is configured to repeat each request from the client application to Service-now using SSL (All communication between the customer and ServiceNow must be SSL). Since the traffic between the source client application and the customer-premises reverse proxy is unencrypted, traffic can be captured, logged, and examined. This practice allows you to verify that preemptive Basic Authorization is happening (as opposed to constant challenge/response behavior) and that the JSESSIONID values sent by ServiceNow in Set-Cookie: headers are being noticed by your client application and repeated back in Cookie: headers on subsequent requests, etc. The Apache Web Server, which is bundled in most Linux distributions and with Mac OS X, can easily be configured to do this.
- **Test with large datasets of the kind that will be encountered in production, rather than relatively tiny test data sets.** Use realistic test data and event rates so that applications which work perfectly in a tiny test instance do not surprise anyone when the message rate skyrockets in a production deployment scenario.

Message Rates

Service-now uses a gating approach with semaphores, which allows the customer to control the rate at which SOAP messages are processed in a ServiceNow instance. See **Long-Running SOAP Request Support** for details.

Use Version-Specific REST APIs

When accessing a ServiceNow REST API, such as the Table API or Aggregate API, use a version-specific URL instead of the default URL. A version-specific URL includes a version number; the default URL for a REST API endpoint does not include a version number and always directs to the latest version of that API. Using a version-specific URL allows you to preserve existing functionality when a new version of an API is released. This ensures that web service behavior remains consistent until you decide to migrate to a newer version of the API.

For example, *https://<instance name>.service-now.com/api/now/v1/table/incident* is a version-specific URL. The **v1** portion of this URL indicates that requests to this API will always use version 1 of the API, even when newer versions are released. You can change this URL to *https://<instance name>.service-now.com/api/now/v2/table/incident* to access version 2 of the API.

Note: *Refer to the product documentation for each REST API for information about which versions of that API are available.*

Pull a New SOAP WSDL after Upgrading

Upgrading an instance may add or change columns that are provided in WSDLs. Client applications that use an outdated WSDL may encounter errors. After upgrading, pull new WSDLs using any client applications that access the instance through SOAP and rebuild the stubs. This action updates the client application to use the upgraded WSDL.

Outbound SOAP Web Service

Note: *This article applies to Fuji and earlier releases. For more current information, see Outbound SOAP Web Service* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Overview

The SOAP Message module can be used to develop, prototype, and save outbound SOAP messages that can be reused in business rules and scripts. Outbound SOAP web services comply with SOAP1.1 standards.

You can use outbound SOAP messages in scripts using the SOAPMessageV2 API and the SOAPResponseV2 API, starting with the Fuji release. Examples detailing how to script outbound SOAP are available. For versions prior to Fuji, refer to previous version documentation.

Video Tutorial

The following video tutorial demonstrates how to configure outbound SOAP web service messages to consume third-party web services from the ServiceNow platform. Applies to all supported releases as of Fuji.

How to Consume Third-Party SOAP Web Services

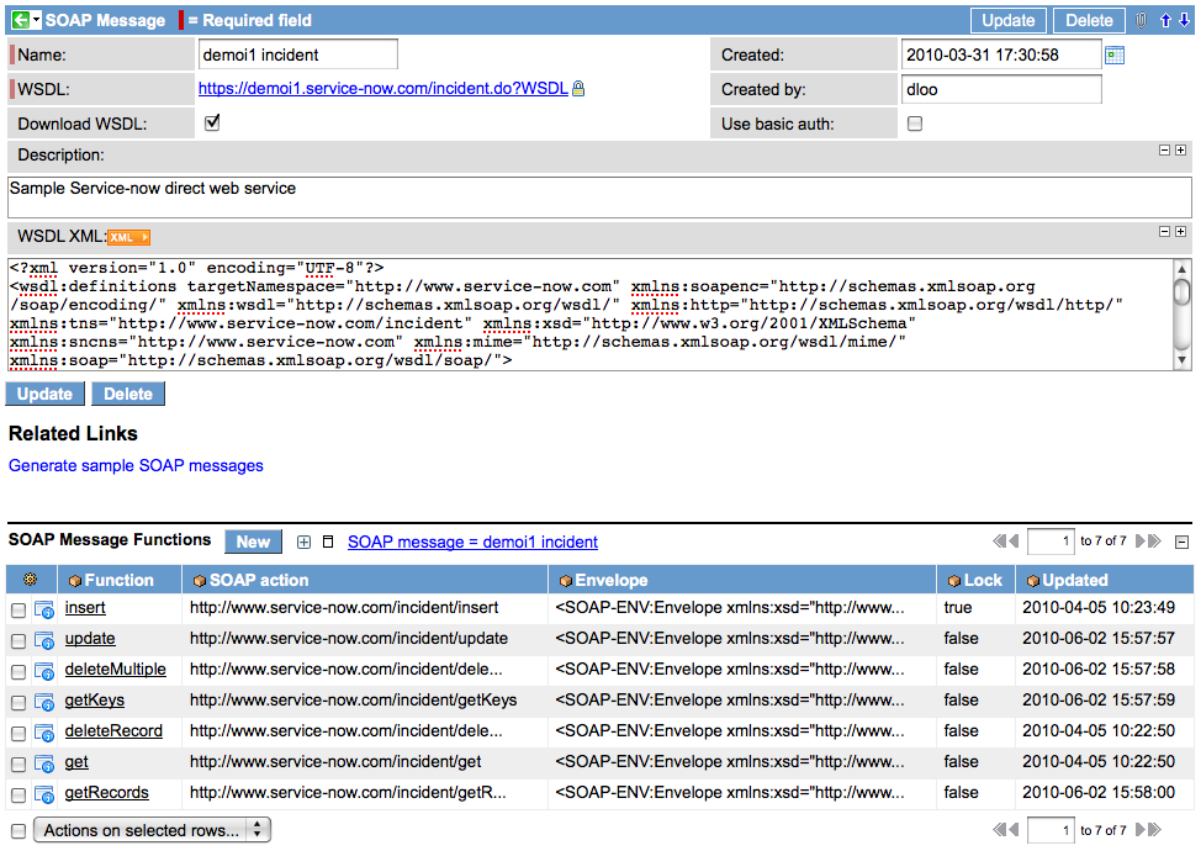
SOAP Message

Information needed to send SOAP requests is stored in SOAP message records. Each record specifies an endpoint for the request, the required format of the request as a web services description language (WSDL) file, authentication information, and a list of functions that can run against the endpoint.

Creating a SOAP Message

1. Navigate to **System Web Services > SOAP Message**.
2. Click **New**.
3. Enter a **Name** to identify the SOAP message.
4. Specify a WSDL using one of these options:
   - To download and use an online WSDL source, select the **Download WSDL** check box and enter the URL for the WSDL in the **WSDL** field
   - To enter the WSDL directly, clear the **Download WSDL** check box, and then copy and paste the WSDL XML into the **WSDL XML** field.
5. If the endpoint is protected by basic authentication, select the **Use basic auth** check box and enter the credentials.
6. If the endpoint requires mutual authentication, select the **Enable mutual authentication** check box and select a **Protocol profile** to use for mutual authentication (starting with the Fuji release).
7. Click **Submit**.

This image shows an example of a SOAP message that connects to a demo instance of ServiceNow.



Name:	demoi1 incident	Created:	2010-03-31 17:30:58
WSDL:	https://demoi1.service-now.com/incident.do?WSDL	Created by:	dloo
Download WSDL:	✓	Use basic auth:	☐

Description:

Sample Service-now direct web service

WSDL XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://www.service-now.com" xmlns:soapenc="http://schemas.xmlsoap.org
/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:tns="http://www.service-now.com/incident" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sncns="http://www.service-now.com" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
```

Related Links

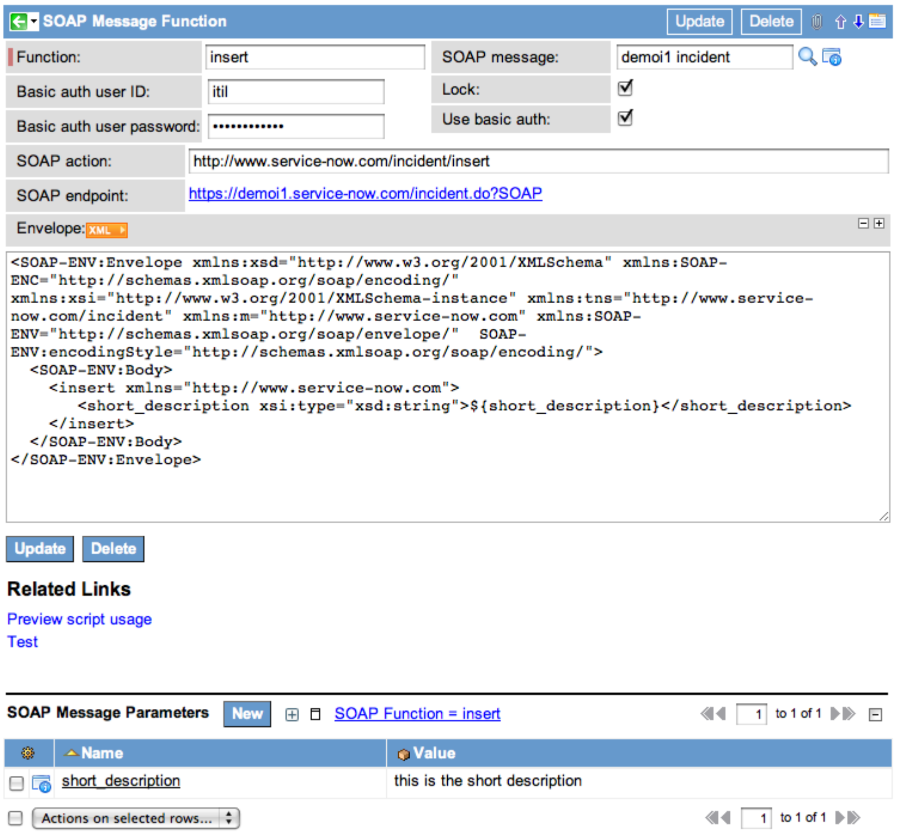
Generate sample SOAP messages

SOAP Message Functions — New — SOAP message = demoi1 incident — 1 to 7 of 7

	Function	SOAP action	Envelope	Lock	Updated
☐	insert	http://www.service-now.com/incident/insert	<SOAP-ENV:Envelope xmlns:xsd="http://www...	true	2010-04-05 10:23:49
☐	update	http://www.service-now.com/incident/update	<SOAP-ENV:Envelope xmlns:xsd="http://www...	false	2010-06-02 15:57:57
☐	deleteMultiple	http://www.service-now.com/incident/dele...	<SOAP-ENV:Envelope xmlns:xsd="http://www...	false	2010-06-02 15:57:58
☐	getKeys	http://www.service-now.com/incident/getKeys	<SOAP-ENV:Envelope xmlns:xsd="http://www...	false	2010-06-02 15:57:59
☐	deleteRecord	http://www.service-now.com/incident/dele...	<SOAP-ENV:Envelope xmlns:xsd="http://www...	false	2010-04-05 10:22:50
☐	get	http://www.service-now.com/incident/get	<SOAP-ENV:Envelope xmlns:xsd="http://www...	false	2010-04-05 10:22:50
☐	getRecords	http://www.service-now.com/incident/getR...	<SOAP-ENV:Envelope xmlns:xsd="http://www...	false	2010-06-02 15:58:00

SOAP Message Functions

After you create a SOAP message record, you can click **Generate sample SOAP messages** to populate the **SOAP Message Functions** related list. The instance creates these functions by reading the supplied WSDL definition.



Function:	insert	SOAP message:	demoi1 incident
Basic auth user ID:	itil	Lock:	✓
Basic auth user password:	•••••••••••••	Use basic auth:	✓
SOAP action:	http://www.service-now.com/incident/insert		
SOAP endpoint:	https://demoi1.service-now.com/incident.do?SOAP		

Envelope:

```
<SOAP-ENV:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:tns="http://www.service-
now.com/incident" xmlns:m="http://www.service-now.com" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"  SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <insert xmlns="http://www.service-now.com">
      <short_description xsi:type="xsd:string">${short_description}</short_description>
    </insert>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Related Links

Preview script usage
Test

SOAP Message Parameters — New — SOAP Function = insert — 1 to 1 of 1

	Name	Value
☐	short_description	this is the short description

The **SOAP action**, **SOAP endpoint**, and **Envelope** fields should be populated automatically based on the WSDL definition. The **Envelope** defines the message to send to the endpoint. In this example, the **Envelope** values have this format:

```
...
<!-- optional -->
<short_description xsi:type="xsd:string">String</short_description>
...
```

To submit a specific value, enter the value directly in the appropriate XML tag. In this example, to set the **Short description** for a record, enter:

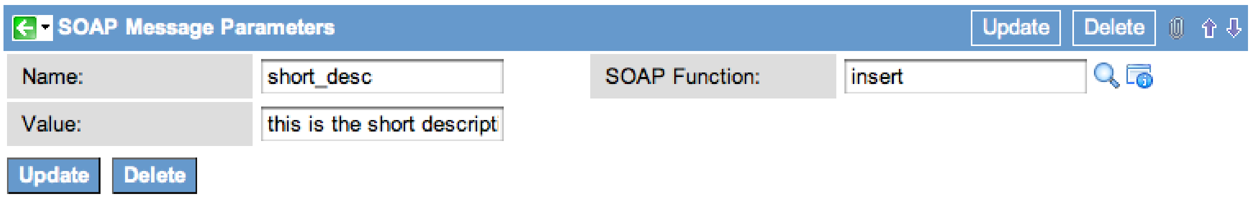
```
...
<short_description xsi:type="xsd:string">This is the short description</short_description>
...
```

Variable Substitution

To use variable substitution, use the format `${<variable_name>}` instead of defining a specific value.

```
...
<short_description xsi:type="xsd:string">${short_desc}</short_description>
...
```

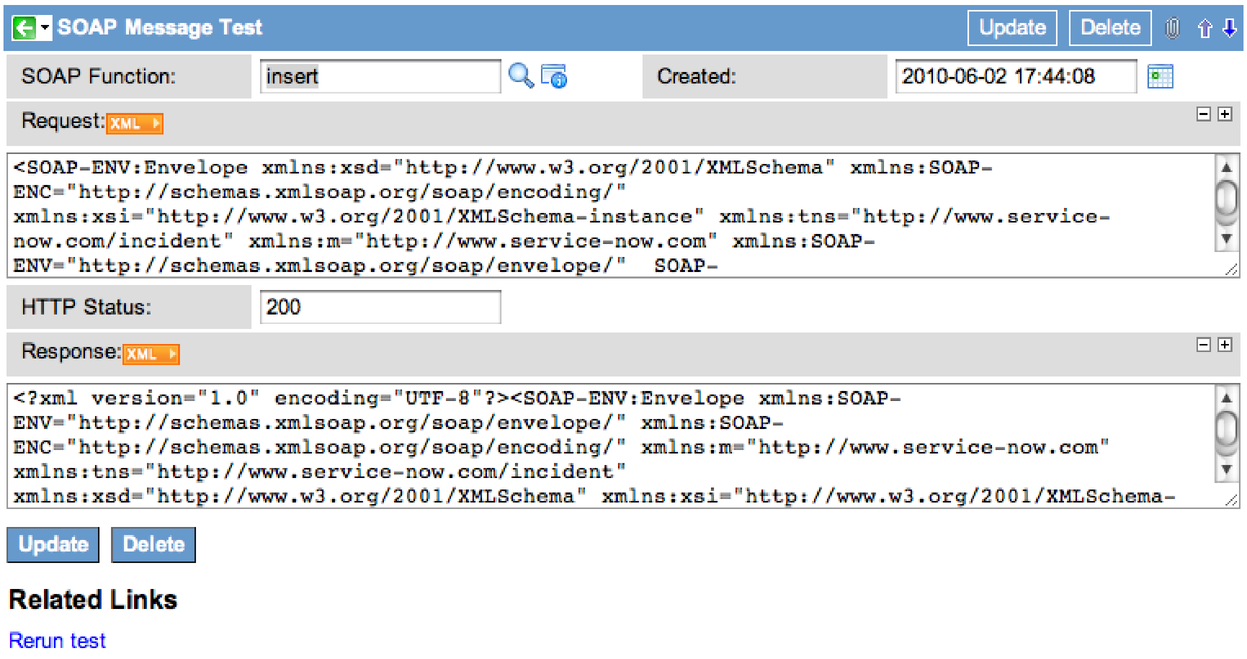
To test variable substitution after you have modified the SOAP envelope with the variables, define values for the variables in the **SOAP Message Parameters** related list. For example, click **New** and enter the following information:



Name:	short_desc	SOAP Function:	insert
Value:	this is the short descript		

Testing

To test the SOAP message, click the **Test** related link. You are redirected to a test result form as shown below.



SOAP Function: insert

Created: 2010-06-02 17:44:08

Request:

```
<SOAP-ENV:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:tns="http://www.service-
now.com/incident" xmlns:m="http://www.service-now.com" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"  SOAP-
```

HTTP Status: 200

Response:

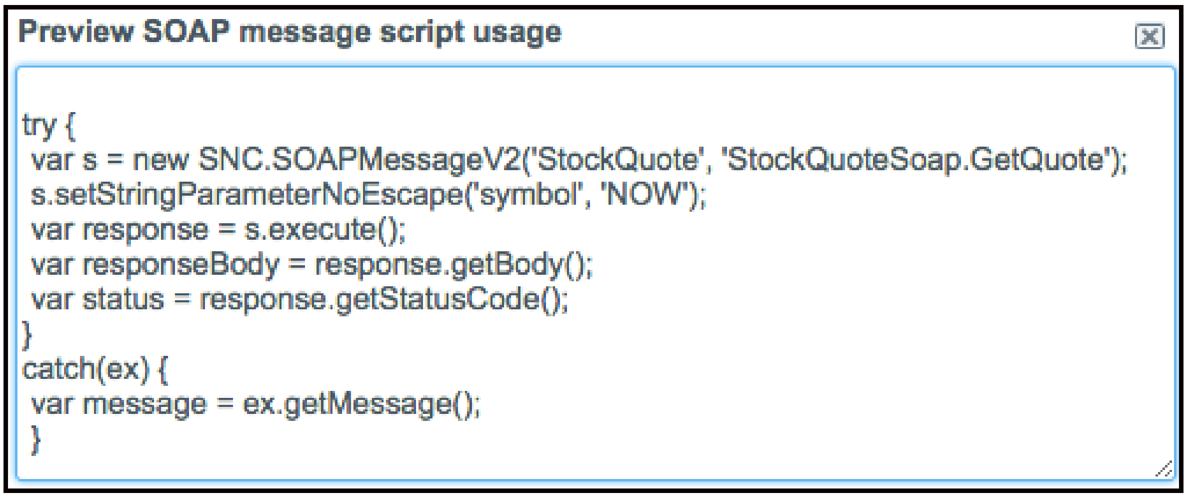
```
<?xml version="1.0" encoding="UTF-8"?><SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:m="http://www.service-now.com"
xmlns:tns="http://www.service-now.com/incident"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
```

You can see the original SOAP request message, the resulting HTTP status code, and the SOAP response in this screen. You can also click the **Rerun test** related link to resubmit the SOAP request.

Note: *A test SOAP message will time out after 60 seconds if a response is not received.*

Using a SOAP Message in a Script

After you have developed and tested the SOAP message, click the **Preview script usage** related link in the SOAP Message Function form. The dialog box displays an example of how you can invoke the SOAP message from a script. See SOAPMessageV2 API for a list of available methods, or Scripting Outbound SOAP for detailed scripting examples.



```
try {
 var s = new SNC.SOAPMessageV2('StockQuote', 'StockQuoteSoap.GetQuote');
 s.setStringParameterNoEscape('symbol', 'NOW');
 var response = s.execute();
 var responseBody = response.getBody();
 var status = response.getStatusCode();
}
catch(ex) {
 var message = ex.getMessage();
 }
```

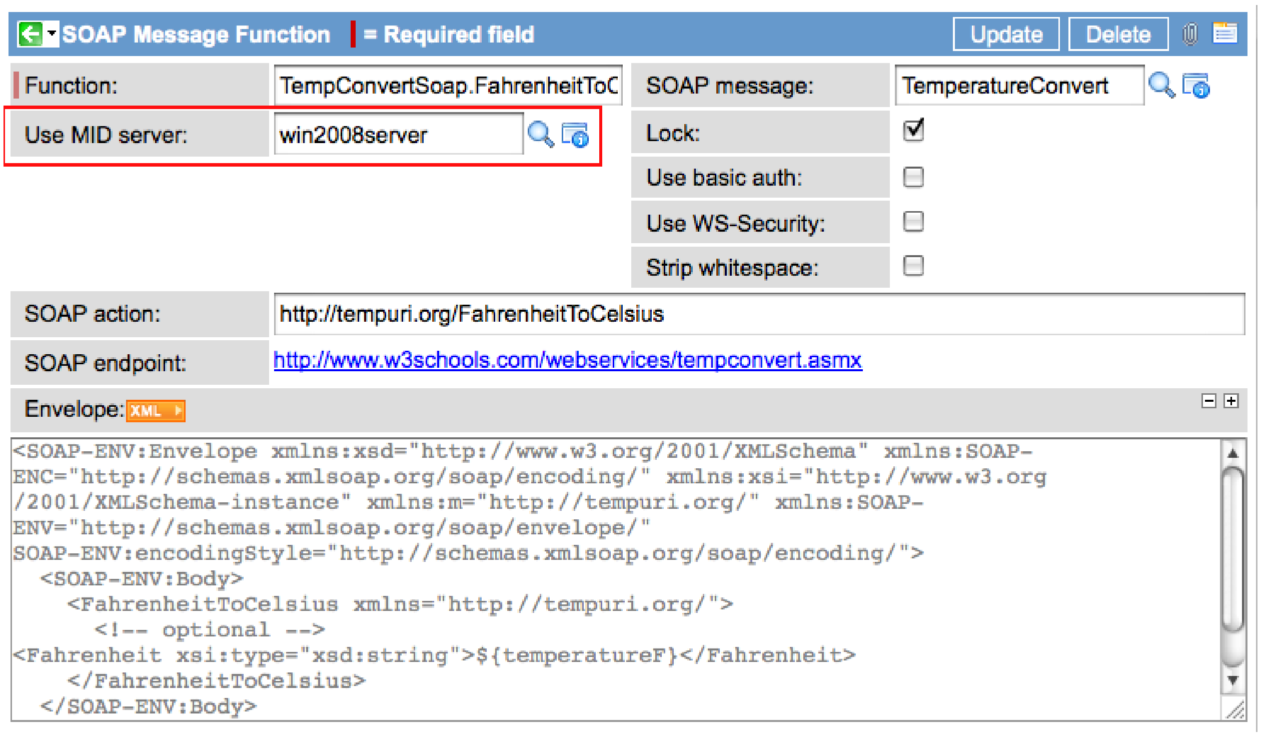
You can manipulate the resulting XML response body with XMLDocument or with `gs.getXMLText` and `gs.getXMLNodeList`.

Demonstration Video

The following video shows examples of creating and sending SOAP messages. The script example in this video uses the version one SOAPMessage API.

Sending a SOAP Message Through a MID Server

When creating SOAP message functions, you can configure the function to be sent through a MID Server selected in the **Use MID Server** field. There must be a running MID Server associated with your ServiceNow instance to use this functionality. All SOAP messages sent through a MID Server are performed asynchronously.



Function:	TempConvertSoap.FahrenheitToC	SOAP message:	TemperatureConvert
Use MID server:	win2008server	Lock:	✓
		Use basic auth:	☐
		Use WS-Security:	☐
		Strip whitespace:	☐
SOAP action:	http://tempuri.org/FahrenheitToCelsius		
SOAP endpoint:	http://www.w3schools.com/webservices/tempconvert.asmx		

Envelope: XML

```
<SOAP-ENV:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org
/2001/XMLSchema-instance" xmlns:m="http://tempuri.org/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <FahrenheitToCelsius xmlns="http://tempuri.org/">
      <!-- optional -->
<Fahrenheit xsi:type="xsd:string">${temperatureF}</Fahrenheit>
    </FahrenheitToCelsius>
  </SOAP-ENV:Body>
```

By specifying a MID Server, all SOAP requests that use this SOAP message are sent through that MID Server. You can override the selected MID Server by using the `setMIDServer(mid server)` API call in a script.

Connectivity Details

For the ServiceNow-initiated SOAP requests to successfully communicate with the web service provider inside a remote network, the ServiceNow instance must have HTTP or HTTPS access to the SOAP endpoint at the provider. Like any integration, such as LDAP, web services, or JDBC, the SOAP endpoint may reside behind a firewall that is blocking inbound communication from the ServiceNow instance. If this is the case, you need to make network changes to allow this connectivity into your network. You can either modify the firewall and ACL rules to allow the ServiceNow IP address, configure the SOAP message to use a MID Server, or implement a VPN tunnel to allow the ServiceNow communication into your network.

Note: *A common misconception is that because asynchronous SOAP requests are routed through the ECC queue, they are always sent through a MID Server. This is not the case. Asynchronous SOAP requests only use a MID Server when configured to do so.*

Outbound SOAP Security

You can authenticate outbound SOAP messages using several different security protocols. The security protocol you should use depends on the requirements of the web service provider.

Basic Authentication

If the endpoint requires a user name and password, you can provide these credentials using basic authentication.

1. Navigate to **System Web Services > Outbound > SOAP Message**.
2. Select a SOAP message record.
3. In the **SOAP Message Functions** related list, select a function.
4. Select **Use basic auth**.
5. Enter a user name in the **Basic auth user ID** field.
6. Enter the password for that user in **Basic auth user password**.
7. Click **Update**.

Web Service Security

You can sign outbound SOAP messages using a key store and trusted server certificate saved on the instance.

1. Upload a key store certificate with a **Type** of **Java Key Store** or **PKCS12 Key Store**.
2. Upload the trusted server certificate for the key store certificate.
3. Navigate to **System Web Services > Outbound > SOAP Message**.
4. Select a SOAP message record.
5. In the **SOAP Message Functions** related list, select a function.
6. Select **Use WS-Security**.
7. In the **Key store** field, select the Java or PKCS12 key store you uploaded.
8. In the **Key store alias** field, enter the alias that identifies the public and private key pair.
9. In the **Key store password** field, enter the password you assigned the key store record.
10. In the **Certificate** field, select the trusted certificate for the selected key store.
11. Click **Update**.

Mutual Authentication

ServiceNow supports mutual authentication for outbound web services. Mutual authentication is not available for inbound web services.

Configuring SOAP with a Proxy

The following properties provide support for SOAP requests to use a web proxy server.

Property	Description	Examples
glide.http.proxy_host	The proxy server hostname or IP address	proxy.company.com, 192.168.34.54
glide.http.proxy_port	The port number for the proxy server	8080, 9100
glide.http.proxy_username	If the proxy server is authenticating using user name and password, enter a value for this property	proxyuser
glide.http.proxy_password	If the proxy server is authenticating using user name and password, enter a value for this property	password

Enhancements

Fuji

- The glide.outbound.sslv3.disabled system property can disable the SSLv3 protocol for outbound connections.
- Mutual authentication is available for outbound web services.
- The SOAPMessageV2 API and SOAPResponseV2 API provide enhanced interfaces for sending outbound SOAP messages using scripts.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/integrate/outbound-soap/concept/c_OutboundSOAPWebService.html

Scripting Outbound SOAP

Note: *This article applies to Fuji and earlier releases. For more current information, see Scripting Outbound SOAP* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Overview

You can send outbound SOAP messages using JavaScript.

Note: *The examples on this page use the SOAPMessageV2 API and SOAPResponseV2 API, available starting with the Fuji release. For earlier versions, see Scripting Outbound SOAP - Versions Prior to Fuji.*

Sending a Direct SOAP Message

You can send a SOAP message directly to the endpoint. In this example, the script sends a SOAP message requesting a stock quote and waits for a response. If there is no response from the web service provider, or if the specified SOAP message record is unavailable, the script throws an error, handled in this example by the try-catch block.

```
var requestBody;
var responseBody;
var status;
var sm;
try{
      sm = new sn_ws.SOAPMessageV2("StockQuote",
"StockQuoteSoap.GetQuote");  // Might throw exception if message
doesn't exist or not visible due to scope
      sm.setBasicAuth("admin","admin");
      sm.setStringParameter("symbol", "NOW");
      sm.setStringParameterNoEscape("xml_data","<data>test</data>");
      sm.setHttpTimeout(10000) //In Milli seconds. Wait at most 10
seconds for response from http request.

      response = sm.execute();//Might throw exception if http
connection timed out or some issue with sending request itself because
       of encryption/decryption of password and stuff
      responseBody = response.haveError() ? response.getErrorMessage()
: response.getBody();
      status = response.getStatusCode();
} catch(ex) {
      responseBody = ex.getMessage();
      status = '500';
} finally {
      requestBody = sm ? sm.getRequestBody():null;
}
gs.info("Request Body: " + requestBody);
gs.info("Response: " + responseBody);
gs.info("HTTP Status: " + status);
```

Sending a SOAP Message with no SOAP Message Record

You can use the SOAPMessageV2() constructor with no parameters to define a SOAP message entirely in the script. When using this constructor you must provide an endpoint and SOAP action. In this example, the script creates an empty SOAP message and sets the values needed to insert an incident record.

```
var s = new sn_ws.SOAPMessageV2(); //create an empty SOAP message


s.setBasicAuth('admin','admin');


s.setSOAPAction('http://www.service-now.com/incident/insert'); //set


the SOAP action to perform


s.setEndpoint('http://<instance>.service-now.com/incident.do?SOAP'); //set the


web service provider endpoint
```

```
s.setRequestBody('<soapenv:Envelope xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\" xmlns:inc=\"http://www.service-now.com/incident\"><soapenv:Header/><soapenv:Body><inc:insert><short_description>Test Dynamic SOAP</short_description></inc:insert></soapenv:Body></soapenv:Envelope>');


var response = s.execute();


var xmldoc = new XMLDocument(response.getBody());


var incident_sysid = xmldoc.getNodeText('//sys_id');
```

Sending a SOAP Message Asynchronously

You can send a SOAP message asynchronously. When you send an asynchronous message the instance does not wait for a response before proceeding. You must handle waiting for a response within your code.

```
var requestBody;
var responseBody;
var status;
var sm;
try{
      sm = new sn_ws.SOAPMessageV2("StockQuote",
"StockQuoteSoap.GetQuote");  // Might throw exception if message
doesn't exist or not visible due to scope
      sm.setBasicAuth("admin","admin");
      sm.setStringParameter("symbol", "NOW");
      sm.setStringParameterNoEscape("xml_data","<data>test</data>");
      response = sm.executeAsync(); //Might throw exception if http
connection timed out or some issue with sending request itself because
of encryption/decryption of password

      response.waitForResponse(60);// In Seconds, Wait at most 60
seconds to get response from ECC Queue/Mid Server //Might throw
exception timing out waiting for response in ECC queue

      responseBody = response.haveError() ? response.getErrorMessage()
: response.getBody();
      status = response.getStatusCode();
} catch(ex) {
      responseBody = ex.getMessage();
      status = '500';
} finally {
      requestBody = sm ? sm.getRequestBody():null;
}
gs.info("Request Body: " + requestBody);
gs.info("Response: " + responseBody);
gs.info("HTTP Status: " + status);
```

Sending a SOAP Message Through a MID Server

You can send a SOAP message through a MID Server. By sending the message through a MID Server, you can access endpoints that are behind a firewall or within a private network. All SOAP messages sent through a MID Server are asynchronous.

```
var requestBody;
var responseBody;
var status;
var sm;
try{
      sm = new sn_ws.SOAPMessageV2("StockQuote",
"StockQuoteSoap.GetQuote");  // Might throw exception if message
doesn't exist or not visible due to scope
      sm.setBasicAuth("admin","admin");
      sm.setStringParameter("symbol", "NOW");
      sm.setStringParameterNoEscape("xml_data","<data>test</data>");
      sm.setMIDServer('mid_server_name');
      response = sm.execute();//Might throw exception if http
connection timed out or some issue with sending request itself because
of encryption/decryption of password and stuff

      response.waitForResponse(60);// In Seconds, Wait at most 60
seconds to get response from ECC Queue/Mid Server //Might throw
exception timing out waiting for response in ECC queue

      responseBody = response.haveError() ? response.getErrorMessage()
: response.getBody();
      status = response.getStatusCode();
} catch(ex) {
      responseBody = ex.getMessage();
      status = '500';
} finally {
      requestBody = sm ? sm.getRequestBody():null;
}
gs.info("Request Body: " + requestBody);
gs.info("Response: " + responseBody);
gs.info("HTTP Status: " + status);
```

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/integrate/outbound-soap/reference/r_ScriptingOutboundSOAP.html

SOAPMessageV2 API

This article applies to Fuji. For more current information, see SOAPMessageV2 [1] **at** https:/ / developer. servicenow.com'''

The ServiceNow Wiki is no longer being updated. Please refer to the Developer Portal [2] **for the latest information.**

Overview

The SOAPMessageV2 API allows you to send an outbound SOAP message using JavaScript. Use the SOAPResponseV2 API to manage the response returned by the SOAP provider. For examples demonstrating how to use this API, see Scripting Outbound SOAP.

Note: *The outbound SOAP web service API is significantly enhanced with the Fuji release. This page describes current behavior. For previous version information, see Scripting Outbound SOAP - Versions Prior to Fuji.*

NEW: You can also find this class documented in the easy-to-read API reference on the Developer Program web site: SOAPMessageV2 [3].

Method Summary

Return Type	Method
Not Applicable	SOAPMessageV2() The message constructor.
Not Applicable	SOAPMessageV2(String soapMessage, String soapFunction) The message constructor.
SOAPResponse	execute() Send the SOAP Message to the endpoint.
SOAPResponse	executeAsync() Send the SOAP Message to the endpoint asynchronously.
void	setSOAPAction(String soapAction) Define the SOAP action this SOAP message performs.
void	setHttpTimeout(Number timeoutMs) Set the amount of time the request waits for a response from the web service provider before the request times out.
void	setBasicAuth(String userName, String userPass) Set basic authentication headers for the SOAP message.
void	setMutualAuth(String profileName) Set the mutual authentication protocol profile for the SOAP message.
void	setWSSecurity(String keystoreId, String keystoreAlias, String keystorePassword, String certificateId) Set web service security values for the SOAP message.
void	setStringParameter(String name, String value) Set a variable from the SOAP message record to the specified value.
void	setStringParameterNoEscape(String name, String value) Set a variable from the SOAP message record to the specified value without escaping XML reserved characters.

void	setEccCorrelator(String correlator) Associate outbound requests and the resulting response record in the ECC queue.
void	setEccParameter(String name, String value) Override a value from the database by writing to the SOAP message payload.
void	setRequestHeader(String headerName, String headerValue) Set an HTTP header in the SOAP message to the specified value.
void	setRequestBody(String requestBody) Set the body content to send to the web service provider.
void	setEndpoint(String endpoint) Set the endpoint for the SOAP message.
void	setMIDServer(String midServerName) Configure the SOAP message to be sent through a MID Server.
String	getRequestBody() Get the content of the SOAP message body.
String	getEndpoint() Get the URL of the endpoint for the SOAP message.
String	getRequestHeader(String headerName) Get the value for an HTTP header specified by the SOAP client.
Object	getRequestHeaders() Get name and value for all HTTP headers specified by the SOAP client.

Constructors

SOAPMessageV2()

Instantiates an empty SOAPMessageV2 object. When using an object instantiated this way, you must manually specify a SOAP action and endpoint.

Parameters

- `None`

SOAPMessageV2(String soapMessage, String soapFunction)

Instantiate a SOAPMessageV2 object from a SOAP message record and a function associated with that record. Values such as the endpoint, authentication, or MID Server settings from the SOAP message record apply to this object.

Parameters:

- `soapMessage` - (String) the SOAP message record you want to use as the base for this object.
- `soapFunction` - (String) the SOAP function you want to execute. Available SOAP functions depend on the WSDL supplied by the web service provider.

Method Detail

execute()

Send the SOAP message to the endpoint.

Parameters:

- `None`

Returns:

`SOAPResponse` - the response returned by the SOAP provider. For more information, see SOAPResponseV2 API.

executeAsync()

Send the SOAP message to the ECC queue. SOAP messages in the ECC queue are processed by the **SOAPClient** business rule. By default, this business rule does not run asynchronously. To configure this business rule to run asynchronously, set the **When** value to **Async** and add `current.update()` to the end of the **Script**. The instance does not wait for a response from the web service provider when sending a message through the ECC queue.

Parameters:

- `None`

Returns:

`SOAPResponse` - the response returned by the SOAP provider. For more information, see SOAPResponseV2 API. Attempting to use the SOAP response object before the response has been processed may result in a timeout error.

setSOAPAction(String soapAction)

Define the SOAP action this SOAP message to performs. The WSDL for your web service provider lists SOAP actions you can perform. You must call this method when using the SOAPMessageV2() constructor with no parameters.

Parameters:

- `soapAction` - (String) The SOAP action this SOAP message performs.

Returns:

`void`

setHttpTimeout(Number timeoutMs)

Set the amount of time the SOAP message waits for a response from the web service provider before the request times out.

Parameters:

- `timeoutMs` - (Number) the amount of time to wait for a response from the web service provider, in milliseconds.

Returns:

    void

setBasicAuth(String userName, String userPass)

Set basic authentication headers for the SOAP message. Setting basic authentication headers using this method overrides basic authentication values defined in the SOAP message record.

Parameters:

- `userName` - (String) the username to use when authenticating the SOAP message.
- `userPass` - (String) the password for the specified user.

Returns:

    void

setMutualAuth(String profileName)

Set the mutual authentication protocol profile for the SOAP message. Setting a protocol profile using this method overrides the protocol profile selected for the SOAP message record.

Parameters:

- `profileName` - (String) The **Name** of the protocol profile to use for mutual authentication.

Returns:

    void

setWSSecurity(String keystoreId, String keystoreAlias, String keystorePassword, String certificateId)

Sets web service security values for the SOAP message. Setting security values using this method overwrites web service security values defined for the SOAP message record.

Parameters:

- `keystoreId` - (String) the sys_id of the Java or PKCS12 key store to use.
- `keystoreAlias` - (String) the alias that identifies the public and private keys.
- `keystorePassword` - (String) the password assigned to the key store record.
- `certificateId` - (String) the sys_id of the trusted server certificate.

Returns:

    void

setStringParameter(String name, String value)

Set a variable with the specified name from the SOAP message record to the specified value. XML reserved characters in the value are converted to the equivalent escaped characters.

Parameters:

- `name` - (String) the name of the SOAP message variable.
- `value` - (String) the value to assign to the specified variable.

Returns:

```
void
```

setStringParameterNoEscape(String name, String value)

Set a variable with the specified name from the SOAP message record to the specified value. This method is equivalent to setStringParameter but does not escape XML reserved characters.

Parameters:

- `name` - (String) the name of the SOAP message variable.
- `value` - (String) the value to assign to the specified variable.

Returns:

```
void
```

setEccCorrelator(String correlator)

Associate outbound requests and the resulting response record in the ECC queue. This method only applies to SOAP messages sent through a MID Server. The correlator provided populates the **Agent correlator** field on the ECC queue record for the response. Provide a unique correlator for each outbound request to associate the correct results in the ECC queue with the request when designing asynchronous automation through a MID Server.

Parameters:

- `correlator` - (String) a unique identifier.

Returns:

```
void
```

setEccParameter(String name, String value)

Override a value from the database by writing to the SOAP message payload. This method only applies to SOAP messages sent through a MID Server. Use this method when a value from the SOAP message in the database is invalid, such as when the endpoint URL is longer than the maximum **SOAP endpoint** field length.

These are valid values for the `name` parameter.

- **source:** the endpoint URL.
- **name:** the SOAP message function to run.

Parameters:

- `name` - (String) the name of the ECC parameter.
- `value` - (String) the value to assign to the specified ECC parameter.

Returns:

```
void
```

setRequestHeader(String headerName, String headerValue)

Set an HTTP header in the SOAP message to the specified value.

Parameters:

- `headerName` - (String) the name of the header.
- `headerValue` - (String) the value to assign to the specified header.

Returns:

```
void
```

setRequestBody(String requestBody)

Set the body content to send to the web service provider. When you set the body content using this method, variables in the body are not substituted for parameters from the SOAP message function record. You must explicitly define all values within the SOAP message body.

Parameters:

- `requestBody` - (String) the body of the SOAP message.

Returns:

```
void
```

setEndpoint(String endpoint)

Set the endpoint for the SOAP message. By default, the SOAP message uses the endpoint specified in the SOAP message record. Use this method to override the default. You must call this method when using the SOAPMessageV2() constructor with no parameters.

Parameters:

- `endpoint` - (String) the URL of the SOAP web service provider you want to interface with.

Returns:

```
void
```

setMIDServer(String midServerName)

Configure the SOAP message to be sent through a MID Server. By default, the SOAP message uses the MID Server specified in the SOAP message function record. Use this method to override the default.

Parameters:

- `midServerName` - (String) the name of the MID Server you want to send the SOAP message through. Your instance must have an active MID Server with the specified name.

Returns:

```
void
```

getRequestBody()

Get the content of the SOAP message body.

Parameters:

- `None`

Returns:

`String` - the SOAP message body.

getEndpoint()

Get the endpoint for the SOAP message.

Parameters:

- `None`

Returns:

`String` - the URL of the SOAP web service provider.

getRequestHeader(String headerName)

Get the value for an HTTP header specified by the SOAP client. By default, this method cannot return the value for a header set automatically by the system. To grant this method access to all headers, set the property `glide.http.log_debug` to **true**.

Parameters:

- `headerName` - (String) the request header you want to get the value for.

Returns:

`String` - the value of the specified header.

getRequestHeaders()

Get HTTP headers that were set by the SOAP client and the associated values. This method does not return headers set automatically by the system. To configure this method to return all headers, set the property `glide.http.log_debug` to **true**.

Parameters:

- `None`

Returns:

`Object` - an Object that maps the name of each header to the associated value.

References

[1] https://developer.servicenow.com/app.do#!/api_doc?v=jakarta&id=c_SOAPMessageV2API
[2] https://developer.servicenow.com/app.do#!/home
[3] http://developer.servicenow.com/app.do#!/api_doc?to=class__soapmessagev2

SOAPResponseV2 API

Overview

The SOAPResponseV2 API allows you to use the data returned by an outbound SOAP message in JavaScript code. A SOAPResponseV2 object is returned by the SOAPMessageV2 functions execute() and executeAsync().

Note: *The outbound SOAP web service API is significantly enhanced with the Fuji release. This page describes current behavior. For previous version information, see scripting outbound SOAP.*

NEW: You can also find this class documented in the easy-to-read API reference on the Developer Program web site: SOAPResponseV2 [1].

Method Summary

Return Type	Method
void	waitForResponse(Number timeoutSecs) Set the amount of time the instance waits for a response.
Number	getStatusCode() Get the numeric HTTP status code returned by the SOAP provider.
String	getHeader(String name) Get the value for a specified HTTP header.
Object	getHeaders() Get all HTTP headers returned in the SOAP response and the associated values.
String	getBody() Get the content of the SOAP response body.
boolean	haveError() Indicate if there was an error during the SOAP transaction.
Number	getErrorCode() Get the numeric error code if there was an error during the SOAP transaction.
String	getErrorMessage() Get the error message if there was an error during the SOAP transaction.

Method Detail

waitForResponse(Number timeoutSecs)

Set the amount of time the instance waits for a response from the web service provider. This method overrides the property `glide.soap.outbound.ecc_response.timeout` for this SOAP response.

Parameters:

- `timeoutSecs` - (Number) the amount of time, in seconds, to wait for this response.

Returns:

`void`

getStatusCode()

Get the numeric HTTP status code returned by the SOAP provider.

Parameters:

- `None`

Returns:

`Number` - the numeric status code returned by the SOAP provider, such as 200 for a successful response.

getHeader(String name)

Get the value for a specified HTTP header.

Parameters:

- `name` - (String) the name of the header that you want the value for, such as Set-Cookie.

Returns:

`String` - the value of the specified header.

getHeaders()

Get all HTTP headers returned in the SOAP response and the associated values.

Parameters:

- `None`

Returns:

`Object` - an Object that maps the name of each header to the associated value.

getBody()

Get the content of the SOAP response body.

Parameters:

- `None`

Returns:

`String` - the SOAP response body.

haveError()

Indicate if there was an error during the SOAP transaction.

Parameters:

- `None`

Returns:

`boolean` - true if there was an error, false if there was no error.

getErrorCode()

Get the numeric error code if there was an error during the SOAP transaction. This error code is specific to the ServiceNow platform, it is not an HTTP error code. Provide this error code if you require assistance from ServiceNow Customer Support.

Parameters:

- `None`

Returns:

`Number` - the numeric error code, such as 1 for a socket timeout.

getErrorMessage()

Get the error message if there was an error during the SOAP transaction.

Parameters:

- `None`

Returns:

`String` - the error message.

References

[1] http://developer.servicenow.com/app.do#!/api_doc?to=class__soapresponsev2

Article Sources and Contributors

SOAP Web Service *Source*: http://wiki.servicenow.com/index.php?oldid=250513 *Contributors*: David Loo, Fuji.publishing.user, G.yedwab, Guy.yedwab, Jeremy.norris, John.andersen, John.ramos, Joseph.messerschmidt, Michelle.Corona, Rachel.sienko, Steven.wood, Vaughn.romero, Wallymarx

SOAP Direct Web Service API *Source*: http://wiki.servicenow.com/index.php?oldid=250900 *Contributors*: Fuji.publishing.user, John.ramos, Joseph.messerschmidt

Scripted Web Services *Source*: http://wiki.servicenow.com/index.php?oldid=250850 *Contributors*: Ajandersen, Anthony.devine, David Loo, Emily.partridge, Fuji.publishing.user, Guy.yedwab, John.ramos, Joseph.messerschmidt, Rachel.sienko, Steven.wood

AttachmentCreator SOAP Web Service *Source*: http://wiki.servicenow.com/index.php?oldid=250040 *Contributors*: David Loo, John.andersen, John.ramos, Joseph.messerschmidt, Mark.stanger, Martin.wood, Publishing.user, Rachel.sienko, Steven.wood, Vaughn.romero

ECC Queue Retry Policy *Source*: http://wiki.servicenow.com/index.php?oldid=199293 *Contributors*: David Loo, David.Bailey, Joseph.messerschmidt, Rachel.sienko, Steven.wood, Wallymarx

SOAP Session Management and Reporting *Source*: http://wiki.servicenow.com/index.php?oldid=100958 *Contributors*: Joseph.messerschmidt, Steven.wood

Long-Running SOAP Request Support *Source*: http://wiki.servicenow.com/index.php?oldid=250694 *Contributors*: Emily.partridge, Fuji.publishing.user, John.ramos, Joseph.messerschmidt, Peter.smith, Steven.wood, Vaughn.romero

Web Services Integrations Best Practices *Source*: http://wiki.servicenow.com/index.php?oldid=248976 *Contributors*: Chuck.tomasi, Fuji.publishing.user, George.rawlins, Joseph.messerschmidt, Norris.merritt, Steven.wood, Vaughn.romero

Outbound SOAP Web Service *Source*: http://wiki.servicenow.com/index.php?oldid=250770 *Contributors*: Fuji.publishing.user, John.ramos, Joseph.messerschmidt, Mary.stromberg, Steven.wood

Scripting Outbound SOAP *Source*: http://wiki.servicenow.com/index.php?oldid=250856 *Contributors*: Fuji.publishing.user, John.ramos, Joseph.messerschmidt

SOAPMessageV2 API *Source*: http://wiki.servicenow.com/index.php?oldid=251147 *Contributors*: Amy.bowman, Fuji.publishing.user, John.ramos, Joseph.messerschmidt

SOAPResponseV2 API *Source*: http://wiki.servicenow.com/index.php?oldid=247222 *Contributors*: Amy.bowman, Fuji.publishing.user, Joseph.messerschmidt

Image Sources, Licenses and Contributors