SOAP API Setup

The SOAP-based Web Services API in Agiloft is enabled on a per-knowledgebase basis, and uses a token obtained at login to identify the session on subsequent calls.

Enable SOAP Web Services in a Knowledgebase

1. Click the **Setup** gear in the top-right corner and go to **System > Manage Web Services**.
2. By default, the text reads: SOAP Web Services are currently Off
3. Click Enable WS. The screen refreshes, and the text reads: SOAP Web Services are currently On
4. At this point, SOAP Web Services is on, the WSDL is automatically generated for the knowledgebase, and you will be able to access the knowledgebase via the WSDL. A link to your WSDL appears below the Disable WS button:

5. Once enabled, the endpoint for the SOAP Web Service for knowledgebase ABC will become available at: `https://SERVER/ewws/KBNAME/EWWSv2Service?wsdl`
   a. You can access the WSDL document for the purpose of generation of the client code at the following address: `https://SERVER/ewws/KBNAME/EWWSv2Service?wsdl`.
   b. Note that by default for security reasons the WSDL document uses `localhost` as the endpoint. You can either override this immediately after obtaining the WSDL or override the endpoint location in the client code.
6. The Web Services package will persist when the application server restarts, and will be regenerated by the installer when the server is updated.

If any changes are made to the knowledgebase which should also be reflected in the WSDL, re-enable Web Services then refresh the knowledgebase.

SOAP Web Services WSDL

The WSDL defines a set of generic operations common to all knowledgebases, a set of data types specific to the current knowledgebase, and a set of table-specific variants of the generic operations. Each table in the knowledgebase when the WSDL is generated is represented as a complex type definition.

For example

- The EWSelectandRead operation is generic and enables you to retrieve information from the knowledgebase.
- The `EWSelectandRead_WSCase` operation is specific to the Support Cases table and contains all of the data types that are specific to that table.

Table types extend a common `EWWSBaseUserObject` type. The generic methods use this common ancestor for their arguments and result types where relevant. When a generic method is invoked for a specific table, it accepts the table name as a string parameter and returns the corresponding specific descendant type as the result. The code in the client must perform the necessary typecasting, if required. The name of the table is not used, as it can be deduced from the invoked method.

Import the WSDL into your Development Environment

This section provides sample instructions for Apache Axis. For instructions about other development platforms, see your platform's product documentation. *Agiloft* SOAP Web Services follow industry standards and have been tested to work with Java, .NET, PHP, Perl and Python client applications.

Once you have the WSDL file, you need to import it into your development environment to generate the necessary objects for building client Web service applications. For detailed instructions on testing the WSDL, see: Test the SOAP Interface.

Import the WSDL into Apache Axis

Java environments access the API through Java objects that serve as proxies for their server-side counterparts. Before using the API, you must first generate these objects from your organization's WSDL file.
Each SOAP client has its own tool for this process. For Apache Axis, use the WSDL2Java utility.

Before you run WSDL2java, you must have Axis installed on your system and all of its component JAR files must be referenced in your classpath.

The basic syntax for WSDL2Java is `java -classpath pathToJAR/fileName org.apache.axis.wsdl.WSDL2Java`

```
-a pathToWsdlDoc
```

The `-a` switch generates code for all elements, referenced or not, which may be necessary depending on your WSDL.

If you have JAR files in more than one location, list them with a semicolon separating the files. For example, if the Axis JAR files are installed in C:\axis-1_3, and the WSDL is named MyKB.wsdl and is stored in C:\myKB:

```
java -classpath c:\axis-1_3\lib\axis.jar;c:\axis-1_3\lib\axis-ant.jar;
c:\axis-1_3\lib\axis-schema.jar;c:\axis-1_3\lib\commons-discovery-0.2.jar;
c:\axis-1_3\lib\commons-logging-1.0.4.jar;?c:\axis-1_3\lib\jaxrpc.jar;
c:\axis-1_3\lib\log4j-1.2.8.jar;c:\axis-1_3\lib\saaj.jar;
c:\axis-1_3\lib\wsdl4j-1.5.2.jar; org.apache.axis.wsdl.WSDL2Java
-a C:\myKB\MyKB.wsdl
```

This command will generate a set of folders and Java source code files in the same directory in which it was run. After these files are compiled, they can be included in your Java programs for use in creating client applications.

For most Java development environments, you can use wizard-based tools for this process instead of the command line.

For more information about using WSDL2Java, see https://ws.apache.org/axis/java/reference.html.