Biometric Fingerprint Platform

Developer Guide

Bayometric
1743 Park Avenue,
San Jose, CA 95126,
Phone: 877-917-3287, +1-408-940-3955,
Email: sales@bayometric.com

Table of Content

Bayometric Platform API

Bayometric Platform API is exposed through a SOAP and REST API. This allow developers of different programming languages to use the API on a standard way and avoids the need to recompile code when the Bayometric releases a new version of the platform.

WSDL address and an explanation on how to generate proxy code for .NET and Java is available on Appendix A. Given the complexity of the code generated by gSoap to C++ proxy of Bayometric Platform API, Bayometric supplies a proxy static library that helps developers to integrate C++ application with the API. All references to C++ on this document will be based on the supplied proxy. Platform ships source code of this proxy so developers can change it if needed. Appendix B shows how to call Touch N Go REST API using AJAX.

Depending on the programming language being used and how the service proxy was generated, objects names may vary between this documentation and code generated, but all information here must be available somehow for developer.

Error codes

Table below shows all error codes returned by the API. This is the information that needs to be checked in order to find if the method succeed or not.
Every call to the API returns an object that contains this information.

Name	Value	Description
SUCCESS	0	The operation succeed.
CANCELED_BY_USER	1	User canceled the operation, either by clicking on

		Cancel button or by closing the GUI.
HIT_CONFIRMED	2	Search operation found a match.
NO_HIT	3	Search operation could not find a match.
ALREADY_ENROLLED	4	User being registered is already enrolled on DB with a different ID.
CAPTURE_TIMEOUT	5	Capture operation timed out.
BAD_IMAGE_QUALITY	6	Captured image for search operation was not good enough.
SERVICE_BUSY	7	Service is busy processing a different operation and the requested operation can't be processed.
INCORRECT_CALL	8	Request operation should not be called in this situation.
END_OF_LIST	9	The number of records requested to method is greater than available value and the end of list was reached.
NO_READER_CONNECTED	-1	No fingerprint reader is connected. Please check driver installation and if the platform supports the fingerprint reader being used.
SERVER_CONNECTION_FAILED	-2	Could not connect to server. Check connection configuration.
SERVER_TIMEOUT	-3	The requested operation on server timed out.
SERVER_ERROR	-4	An unexpected error happened on server. Contact the system administrator.
PERSON_ID_NOT_FOUND	-5	The supplied ID could not be found on database. Search operation requesting 1:1 search returns this error when the ID is not available on dabatase.
INVALID_LICENSE	-6	Server has no license installed or the installed license is not valid.
LICENSE_MAX_RECORDS	-7	Database has reached the maximum number of records allowed by license.
LICENSE_MAX_CLIENTS	-8	Platform has reached the maximum number of client machines allowed by license.

LICENSE_EXPIRED	-9	Trial license has expired.
INVALID_SESSION	-10	The supplied session key is not valid.
SESSION_EXPIRED	-11	The supplied session key belongs to an expired session. Another session key must be acquired.
NOT_ALLOWED_OPERATION	-12	Requested operation is not allowed for the authenticated user.
INVALID_CREDENTIALS	-13	Supplied credentials are not valid.
INVALID_PARAMETERS	-14	An invalid parameter was supplied.
REQUIRES_AUTHENTICATION	-15	The requested operation requires a valid session.
USER_NOT_ACTIVE	-16	User is not active or is not an Administrator.

Return objects

Every call to the API returns an object (or derived object) containing at least the following information. For specific cases, method specification will explain the return.

- **ResponseCode:** an integer containing one of the values exposed on Error Codes table.
- **ReturnMessage:** a string containing a brief description of the error.

Creating a session

Privileged operations on the platform (Register and Unregister), requires a valid session key to be supplied. The method responsible for creating a session is CreateSession. It is important to know that for creating sessions, when the method is called, it will request the fingerprint of an Administrator or a Manager, properly registered on the platform.

This method receives a AuthRequestInfo object as parameter, that exposes the following fields:

- **PersonID:** due to security reasons, UserID + Password authentication is not allowed on the current version of the platform. This field is reserved for future use.

- **Password:** due to security reasons, UserID + Password authentication is not allowed on the current version of the platform. This field is reserved for future use.
- **Timeout:** integer value that defines, in seconds, how long fingerprint reader should wait for a finger to be placed before timing out. If a value greater than 999 or less than or equal 0 is supplied, then default value of 30 seconds will be used.
- **ShowGUI:** boolean type that defines whether or not the platform should show GUI.

Output

This method returns a BFSAuthResponse object, containing the following information. Information explained on section “Return objects” above will also be available:

- **PersonID:** ID of the person who supplied fingerprint to create the session.
- **SessionKey:** the session key that must be supplied to privileged methods.

.NET

```
public BFSAuthResponse CreateSession(AuthRequestInfo authRequestInfo);
```

Java

```
public BFSAuthResponse createSession(AuthRequestInfo authRequestInfo) throws
java.rmi.RemoteException
```

C++

```
BFSAuthResponse CreateSession(AuthRequestInfo authRequestInfo);
```

Ending a session

Every created session should be ended when it is not necessary for the application anymore. This avoids session leaking on the platform. To perform this task, Bayometric Platform exposes a method called EndSesion.

This method receives a string as parameter:

- **sessionKey:** session key returned by CreateSession method.

Output

This method returns a base return object as explained before.

.NET

```
public BFSResponseBase EndSession(string sessionKey);
```

Java

```
public BFSResponseBase endSession(java.lang.String sessionKey) throws
java.rmi.RemoteException
```

C++

```
BFSResponseBase EndSession(string sessionKey);
```

Registering a Person

To register a person, Bayometric Platform exposes a method called RegisterPerson. When this method is called with correct parameters, a GUI will be opened. This GUI is responsible for guiding the user through the enrollment process.

This method receives two strings as parameters:

- **sessionKey**: session key returned by CreateSession method. The session must be valid and with proper privileges in order to be able to run the operation.
- **personID**: unique identifier of a person. If null (empty string for C++) is supplied, the platform will generate an ID. It is important to observe that the platform does not check for string length or characters, so, if anything different of a null string is supplied, it will be used. ID validation must be performed at the application level. *If the supplied ID already exists on the platform, the user will be updated and all previous information will be overwritten.*

Output

This method returns a BFSReponse object with the following fields (and base fields explained before):

- **PersonID:** unique ID of the person enrolled on system.
- **PersonFoundID:** if submited fingerprints already exists on system, this field contains the ID of the person that owns those fingerprints.

.NET

```
public BFSResponse RegisterPerson(string sessionKey, string personID);
```

Java

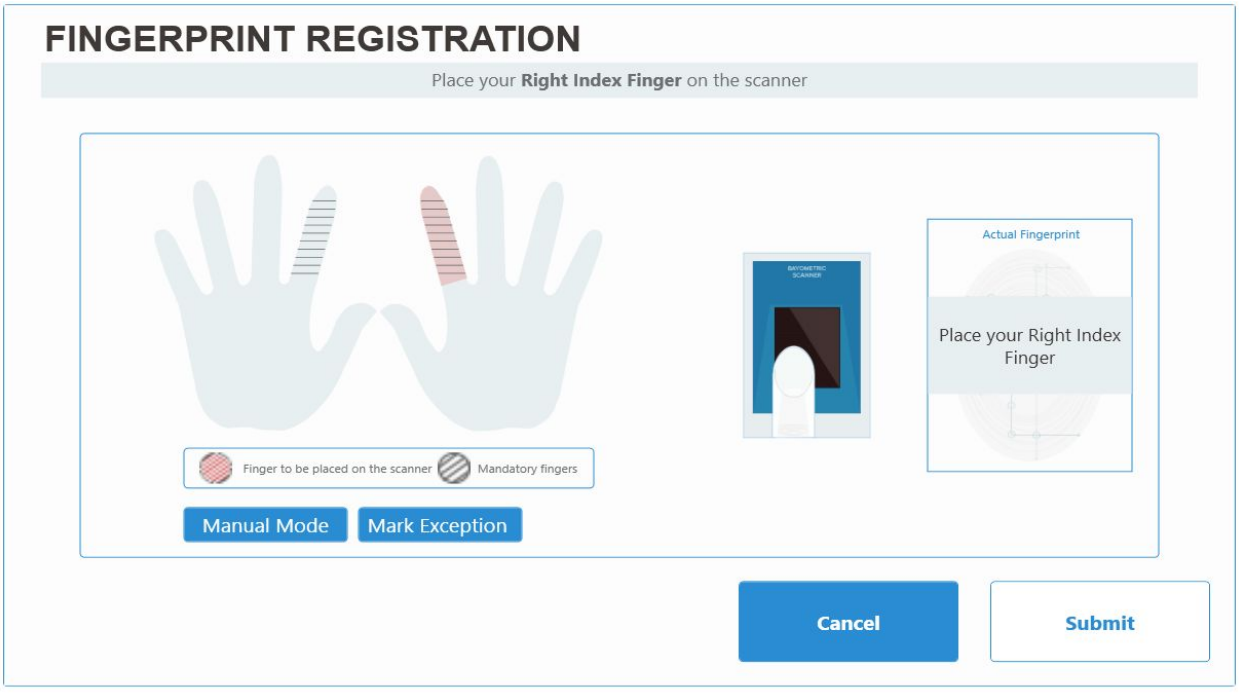
```
public BFSResponse registerPerson(java.lang.String sessionKey, java.lang.String personID)
throws java.rmi.RemoteException
```

C++

```
BFSResponse RegisterPerson(string sessionKey, string personID);
```

The enrollment GUI

Figure below shows Bayometric Platform client GUI that opens when RegisterPerson method is called on the API.



When enrollment GUI opens, it is in “automatic mode”, this means that this is already waiting for the first finger to be placed on fingerprint reader.
Mandatory fingers are marked with a black lines. These fingers must be captured before submitting captured information. Current finger being captured is marked with pink background.

If for any reason user needs to switch to “manual mode”, just click “Manual Mode” button. On manual mode, to start capturing a finger, just click on desired finger.
If person being enrolled does not have a mandatory finger or is not able to capture that finger, select Mark Exception. A Mark Exception Window will open. Please select the reason for exception. After an exception is set, finger is considered to be captured.Once an exception is marked, please select the next finger to be registered by clicking on the finger.

Submit button is enabled after all mandatory fingers are captured.

Unregistering a Person

To unregister a person, Bayometric Platform exposes a method called UnregisterPerson.

This method receives two strings as parameters:

- **sessionKey**: session key returned by CreateSession method. The session must be valid and with proper privileges in order to be able to run the operation.
- **personID**: unique identifier of a person to be deleted.

Output

This method returns a BFSReponse object with the following fields (and base fields explained before). Only base information will be filled. Other fields are reserved for future use.

.NET

```
public BFSResponse UnregisterPerson(string sessionKey, string personID);
```

Java

```
public BFSResponse unregisterPerson(java.lang.String sessionKey, java.lang.String personID)
throws java.rmi.RemoteException
```

C++

```
BFSResponse UnregisterPerson(string sessionKey, string personID);
```

Searching

To search a fingerprint on database, Bayometric Platform exposes two methods called Search and SearchEx.

SearchEx method receives four parameters as input:

- **appToken**: Unique token used by each calling application, eg: "PID of the calling application". If NULL is supplied, a default token will be used. If same token is used by different applications on the same machine, one application will be able to cancel the other one Search request.
- **personID**: unique identifier of a person. If this information is not null (empty string for C++), then platform will perform a Verification, this means that it will compare the captured fingerprint with the fingerprints of this ID stored on server. This will be a 1:1 search. If null is supplied, then a 1:N search will be performed and the supplied fingerprint will be compared to all fingerprints on database.
- **Timeout:** integer value that defines, in seconds, how long fingerprint reader should wait for a finger to be placed before timing out. If a value greater than 999 or less than or equal 0 is supplied, then default value of 30 seconds will be used.
- **ShowGUI:** boolean type that defines whether or not the platform should show GUI.

Search method receives no parameters and use default the following values on SearchEx:

- appToken: NULL (empty string for C++)
- personID: NULL (empty string for C++)
- Timeout: 30
- ShowGUI: True

Output

These methods return a BFSReponse object with the following fields (and base fields explained before):

- **PersonID:** null for search operations.
- **PersonFoundID:** string representing the ID of person found on search request. If a person was not found, this string will be null.

.NET

```
public BFSResponse Search();
public BFSResponse SearchEx(string appToken, string personID, bool showGUI, int timeout);
```

Java

```
public BFSResponse search() throws java.rmi.RemoteException
public searchEx(java.lang.String appToken, java.lang.String personID, java.lang.Boolean showGUI, java.lang.Integer timeout) throws java.rmi.RemoteException
```

C++

```
BFSResponse Search();
BFSResponse SearchEx(string appToken, string personID, int showGUI, int timeout);
```

Canceling a search operation

Bayometric Platform exposes a method that allows users to cancel a running search operation. CancelSearch is the method responsible for this and the following string is received as input.

- **appToken:** Unique token used by each calling application, eg: "PID of the calling application". If NULL (empty string for C++) is supplied, a default token will be used. If same token is used by different applications on the same machine, one application will be able to cancel the other one Search request.

Output

This method returns a BFSReponse object. Only base information will be filled. Other fields are reserved for future use.

.NET

```
public BFSResponse CancelSearch(string appToken);
```

Java

```
public BFSResponse cancelSearch(java.lang.String appToken) throws java.rmi.RemoteException
```

C++

```
BFSResponse CancelSearch(string appToken);
```

Getting registered users IDs

API exposes a method that allows developers to retrieve a list of all the IDs that are enrolled on the platform. This is helpful when something goes wrong with application that uses Bayometric API and also for auditing;

GetRegisteredPersons method receives two parameters as input:

- **startIndex**: zero-based starting position to retrieve IDs from server.
- **numberOfPersons**: number of records to be returned starting at startIndex. Maximum value allowed for this item is 3000.

This method returns a BFSInfoReponse object with the following fields (and base fields explained before):

- **RegisteredPersons:** list of strings that contains all registered IDs on the system within the specified range. Depending on the programming language, the data structure that contains the IDs may vary.

.NET

```
public BFSInfoResponse GetRegisteredPersons(int startIndex, int numberOfPersons);
```

Java

```
public BFSInfoResponse getRegisteredPersons(java.lang.Integer startIndex, java.lang.Integer numberOfPersons) throws java.rmi.RemoteException
```

C++

```
BFSInfoResponse GetRegisteredPersons(int startIndex, int numberOfPersons);
```

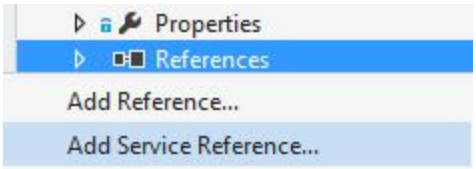
Appendix A - Proxy Generation

Below is presented how to generate proxies for some programming languages and some IDE. Depending on the tool used to generate the proxy, the result may vary, but all information exposed above should still be valid.

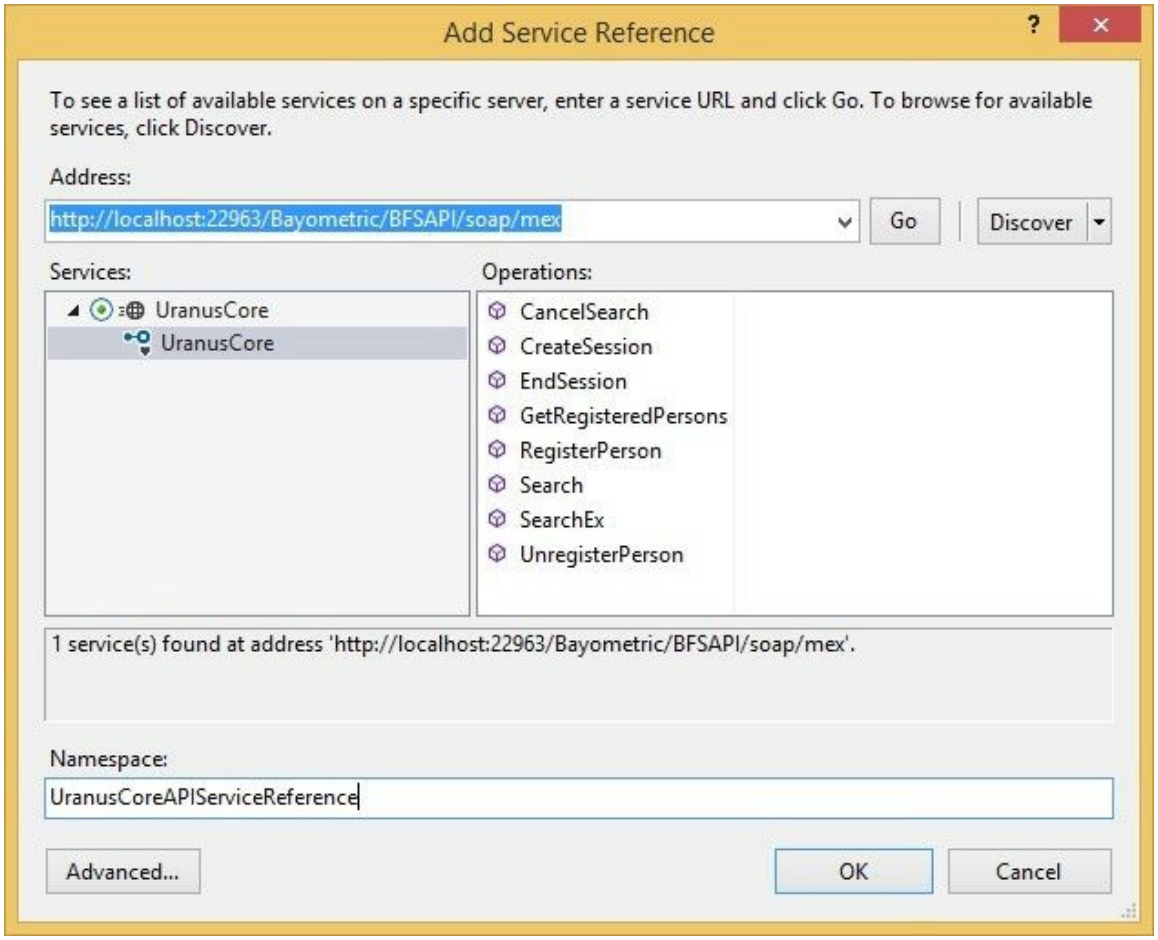
After Uranus (Bayometric client service) is properly installed, wsdl information should be available at: **http://localhost:22963/Bayometric/BFSAPI/soap/mex** .

.NET

Visual Studio supplies two ways of generating service proxy. First and simpler way is to right-click References, on Solution Explorer and click “Add Service Reference…”.



Following screen will open.



Type the address and after clicking Go, under Services, UranusCore should be available. Give a name to the Namespace, UranusCoreAPIServiceReference in this example, and click OK.

If asynchronous methods are desired, make sure to configure it properly, by clicking on “Advanced...”.

Second way to generate the proxy is using WSDL.exe tool, that ships with Visual Studio and is available though Visual Studio Command Prompt. The following command should be typed:

WSDL /verbose C:\PATH\TO\Bayometric.wsdl

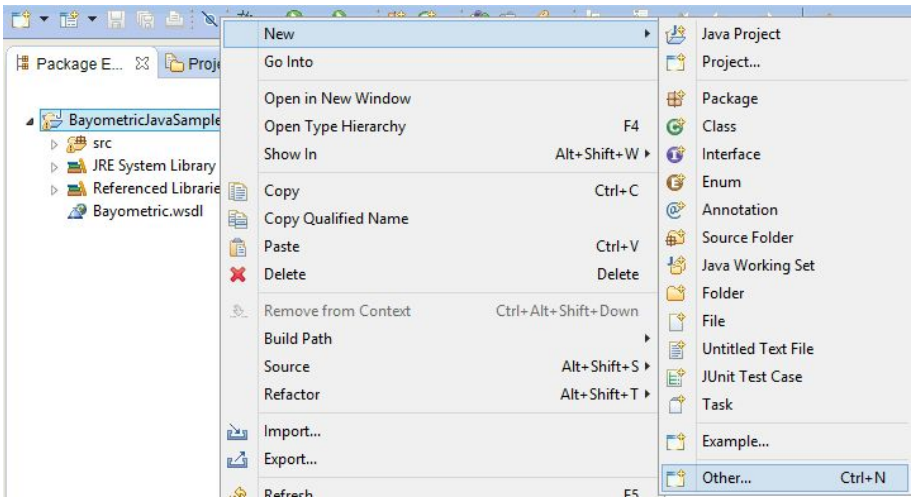
Configuration file and timeout

When Visual Studio generates a proxy to a web service, it adds some fields on the application configuration file. By default, timeout values for connections are 1 minute, and this may not be enough for applications that call RegisterPerson because a user may take longer than 1 minute to capture his fingerprints. To avoid timeout exceptions, we suggest to use a big value or an infinite timeout, just like the example below. Fields in bold can be added. Endpoint name may vary depending on how developer named it during proxy generation.

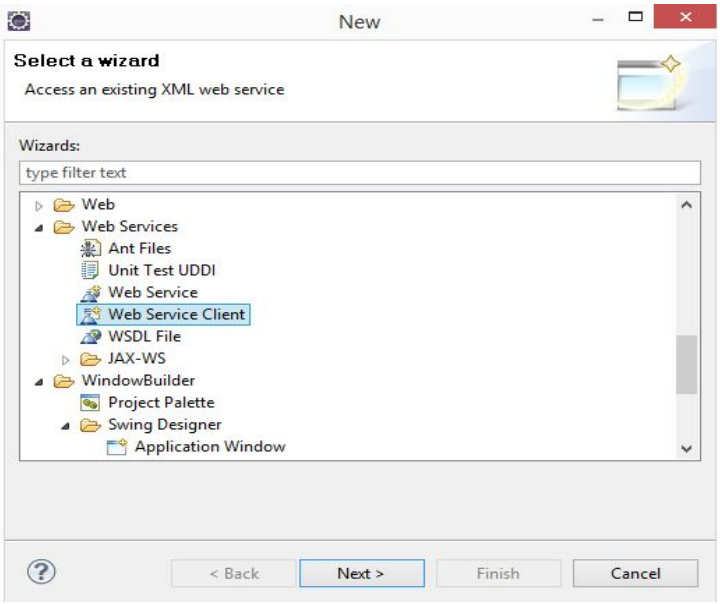
```
<bindings>
        <basicHttpBinding>
           <binding name="UranusCoreEndpoint" closeTimeout="Infinite"
openTimeout="Infinite" receiveTimeout="Infinite" sendTimeout="Infinite"/>
        </basicHttpBinding>
      </bindings>
```

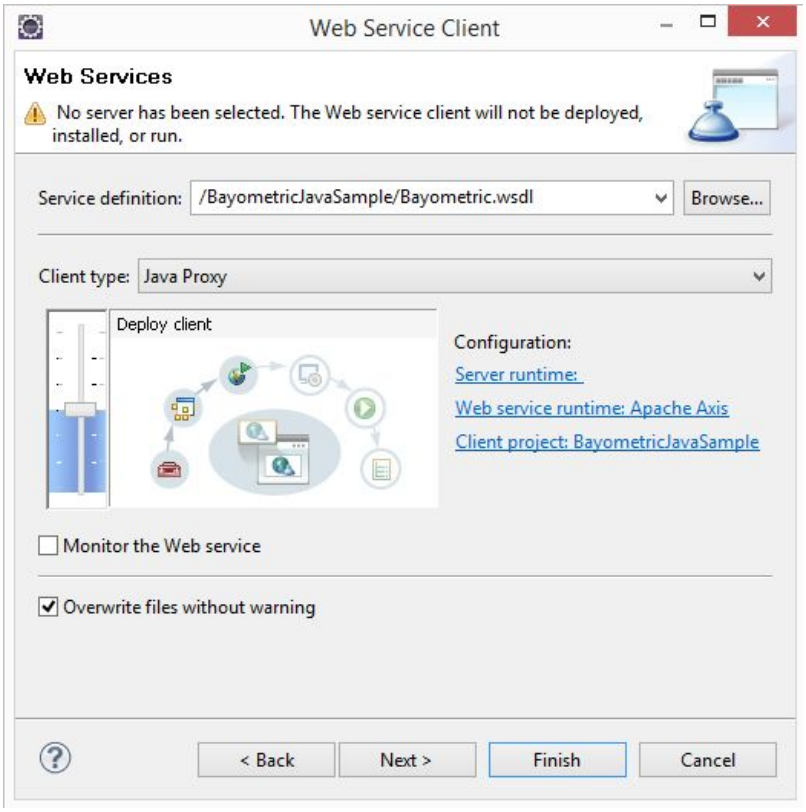
Java

Eclipse IDE supplies an easy way to generate proxy classes based on WSDL file.
In order to do that, add Bayometric.wsdl file to your Java project, so that it appears on Eclipse under the project folder.
After doing this, right click on project, and go to New > Other…, as shown in the figure below.

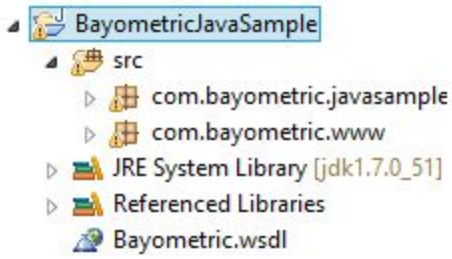


After clicking “Other…” , on the following screen, under “Web Services” folder, select “Web Service Client” and click Next.





Select Bayometric.wsdl file and click Finish. The package “com.bayometric.www” should appear under “src” folder of your Java project.



Appendix B - REST API

Touch N Go exposes a REST API that allows web developers to call it very easily using Javascript. Below is presented how to call each method using AJAX. Please note that code exposed below considers that some HTML elements are present on page. Check sample file available on Sample folder for more details.

Creating a Session

```
var data = '{ "authRequestInfo" : { "PersonID": null,"Password": null,"ShowGUI":
"true","Timeout": "30"}}' ;

var request = $.ajax({
        url: "http://localhost:22963/Bayometric/BFSAPI/json/CreateSession",
        type: "POST",
        contentType: "application/json; charset=utf-8",
        data: data,
        dataType: "json",
        success: function(data) {
                        if(data.ResponseCode == 0){
                                sessionKey = data.SessionKey;
                        }
                        else
                                alert(data.ReturnMessage);
                }
         });
```

Ending a Session

```
var data = '{"sessionKey" :' + "" + sessionKey + "" + '}';
var request = $.ajax({
        url: "http://localhost:22963/Bayometric/BFSAPI/json/EndSession",
        type: "POST",
        contentType: "application/json; charset=utf-8",
        data: data,
        dataType: "json",
        success: function(data) {
                alert(data.ReturnMessage);
                }
        });
```

Registering a Person

```
var data = null;
var personID = $("#personID").val();
if(jQuery.trim(personID).length == 0){
	data = '{"sessionKey" :' + '"' + sessionKey + '"' + ',"personID" : null}';
}
else{
	data = '{"sessionKey" :' + '"' + sessionKey + '"' + ',"personID" : "' + personID + '"}';
}

var request = $.ajax({
	url: "http://localhost:22963/Bayometric/BFSAPI/json/RegisterPerson",
	type: "POST",
	contentType: "application/json; charset=utf-8",
	data: data,
	dataType: "json",
	success: function(data) {
		if(data.ResponseCode == 0){
			alert("User registered with ID: " + data.PersonID);
		}
		else
			alert(data.ReturnMessage);
		}
 });
```

Unregistering a Person

```
var data = null;
var personID = $("#personID").val();
if(jQuery.trim(personID).length == 0){
	data = '{"sessionKey" :' + '"' + sessionKey + '"' + ',"personID" : null}';
}
else{
	data = '{"sessionKey" :' + '"' + sessionKey + '"' + ',"personID" : "' + personID + '"}';
}
```

```
var request = $.ajax({
        url: "http://localhost:22963/Bayometric/BFSAPI/json/UnregisterPerson",
        type: "POST",
        contentType: "application/json; charset=utf-8",
        data: data,
        dataType: "json",
        success: function(data) {
                alert(data.ReturnMessage);
                }
 });
```

Searching

```
var request = $.ajax({
        url: "http://localhost:22963/Bayometric/BFSAPI/json/Search",
        type: "POST",
        success: function(data) {
                if(data.ResponseCode == 2){
                        alert("Match found: " + data.PersonFoundID);
                }
                else
                        alert(data.ReturnMessage);
                        }
        });
```

Canceling a Search

```
var data = '{"appToken" : null}';
var request = $.ajax({
        url: "http://localhost:22963/Bayometric/BFSAPI/json/CancelSearch",
        type: "POST",
        contentType: "application/json; charset=utf-8",
        data: data,
        dataType: "json",
        success: function(data) {
                if(data.ResponseCode != 0){
                        alert(data.ReturnMessage);
                }
        }
});
```