amigopod SOAP API

Unified Visitor Management

Simple Object Access Protocol (SOAP)
Application Programming Interface (API)

Document Revision 1.0

18 March 2010

United States of America
+1 (888) 590-0882

Europe, Middle East & Asia
+34 91 766 57 22

Australia & Pacific
+61 2 8669 1140

http://www.amigopod.com

Table of Contents

1 Introduction

This document explains the SOAP interface available to third-party applications that will integrate with the amigopod Visitor Management Appliance.

1.1 Audience

This document is intended for developers of applications that must interoperate with an amigopod-based visitor management solution.

Basic familiarity with the amigopod Visitor Management Appliance is assumed. For in-depth information about the features and functions of the amigopod appliance, refer to the amigopod Deployment Guide.

Solution developers are assumed to be familiar with HTTP-based web services and the associated concepts and technologies related to these services, including Extensible Markup Language (XML), XML Schemas, Web Service Definition Language (WSDL), and the Simple Object Access Protocol (SOAP).

Many software development tools provide assistance with web services integration. While this document cannot cover all possible integration methods, examples are given using Microsoft Visual C# 2008.

1.2 Document Overview

Section 2 provides a high-level overview of the API, explaining what it is and how to use it. An integration example is provided in section 2.4 to demonstrate the usage of SOAP web services.

Section 3 contains a detailed list of the available API calls, including documentation about each method defined by the web services.

1.3 Disclaimer

The topics of network design, security architectures and visitor access are complex subjects, and no single document can hope to cover all of the possible combinations of network equipment, network design, deployment requirements, and device configurations, nor can all the possible security implications for a particular recommendation be covered.

Therefore, while you read this document, it is best to consider it as a guide to developing your own understanding of the network design topics covered, and as a basis for further investigation.

2 API Overview

2.1 System Requirements

Using the SOAP API on an amigopod Visitor Management Appliance requires the following plugin versions:

- SOAP Web Services 1.0.0 or later
- amigopod Kernel 2.1.13 or later
- Guest Manager 2.1.11 or later

To verify you have the correct plugin versions installed, navigate to **Administrator** > **Plugin Manager** > **Manage Plugins** and check the version number in the list.

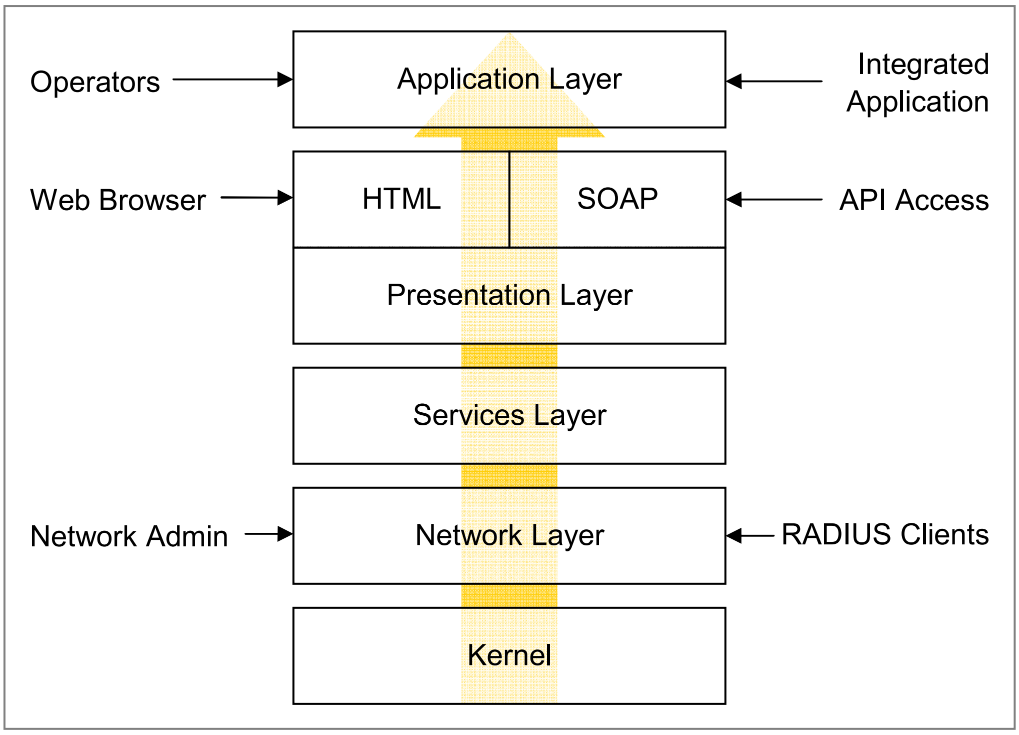
Use the **Update Plugins** link to download and install updated plugins.

2.2 About the API

The amigopod SOAP API provides direct access to the underlying functionality of the amigopod Visitor Management Appliance. Developers wishing to provide integrated applications can make use of this API to programmatically perform actions that would otherwise require manual operation of the GUI.

2.2.1 Architecture overview

The amigopod VMA software is built using multiple layers of software.



At the lowest level, the kernel provides basic functions common to the entire system. This includes the web interface framework, appliance operating system, and runtime support services.

The network layer provides critical networking support, including the RADIUS server and the ability for network administrators to manage and control the networking aspects of the VMA.

The services layer provides one or more implementations of application services that are used by the layers above. Examples of these services include managing a user database used for AAA, handling the authentication of operators, and providing translated text resources.

The presentation layer supplies the tools and framework necessary for the VMA software to interact with the outside world. The basic presentation layer services include authentication checks, session establishment, input checking, validation and conversion, and command execution. Both SOAP and HTML presentation methods are supplied, which adapt the underlying basic presentation to appropriate conventions suitable for a machine-to-machine or human-to-machine interaction.

The application layer provides the page templates, business logic, and the concrete features making up visitor management applications, such as Guest Manager or Hotspot Manager. These applications are built using the services provided by the lower layers.

2.2.2 Authentication and access control

SOAP API requests require that operator authentication information is provided using HTTP Basic authentication.

Page privileges are applied to SOAP authenticated sessions in the same way as the HTML user interface. However, SOAP access also requires the SOAP API privilege to be granted.

Refer to section 2.3.3 for details on creating an operator profile with suitable privileges for SOAP API access.

2.2.3 HTTP headers

When making a SOAP API request, the “SOAPAction” HTTP header is required. The value of this header indicates the type of request being made. Refer to section 0 for the appropriate value for each operation.

The Content-Type header must be specified as either “text/xml” or the “application/soap+xml” MIME type.

The Authorization header must contain a valid HTTP Basic authentication string, as specified in RFC 2617.

2.2.4 Character set encoding

The amigopod Visitor Management Appliance supports the Unicode character set, using the UTF-8 encoding.

Although other character sets are supported, it is recommended that all SOAP API requests be constructed using the UTF-8 character set, as this eliminates the need for character set conversions while also allowing all Unicode characters to be expressed directly.

The character set encoding of a request may be specified using the Content-Type header, for example:

```
Content-Type: text/xml; charset=utf-8
```

2.2.5 SOAP Faults

SOAP 1.1 defines a generalized fault response which is used to indicate that the server could not process the body of the request.

The SOAP `<Fault>` element contains a description of the error, which is divided into a `<faultcode>` that briefly summarizes the problem and a `<faultstring>` that contains a description of the error.

Additionally, the API-specific details of the error are provided in a `<details>` element.

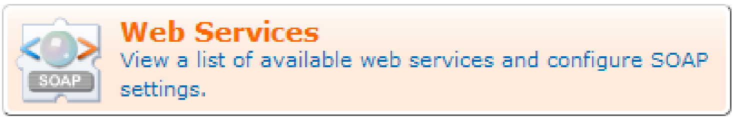
The following table lists the fault codes and corresponding descriptions that may be encountered while using the SOAP API:

Fault Code	**Reason for Fault**
Client.BadRequest	Request exceeds the maximum allowable size. Increase the maximum SOAP request size, or reduce the size of the request.
Client.Authentication	Invalid username or password. Check that the credentials supplied are correct.
Client.MethodNotFound	The SOAP method request was not found.
Client.Error	Another non-specific client error occurred. Check the `<faultstring>` for more details.
Server.MethodNotImplemented	The SOAP method requested is not implemented.
Server.Error	An error occurred while attempting to convert data, or another non-specific server error occurred. Check the `<faultstring>` for more details.

Certain conditions may also cause errors that are not reported as a SOAP fault. These cases are typically caused by errors in constructing the SOAP request. In these cases, a non-XML result may be returned; check the body of the result, or the application log for details about the cause of the error.

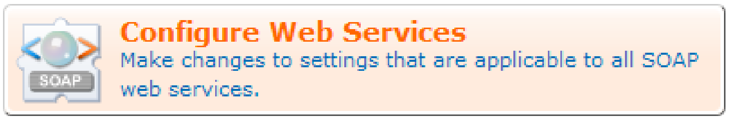
2.3 Using the API

2.3.1 Accessing SOAP Web Services

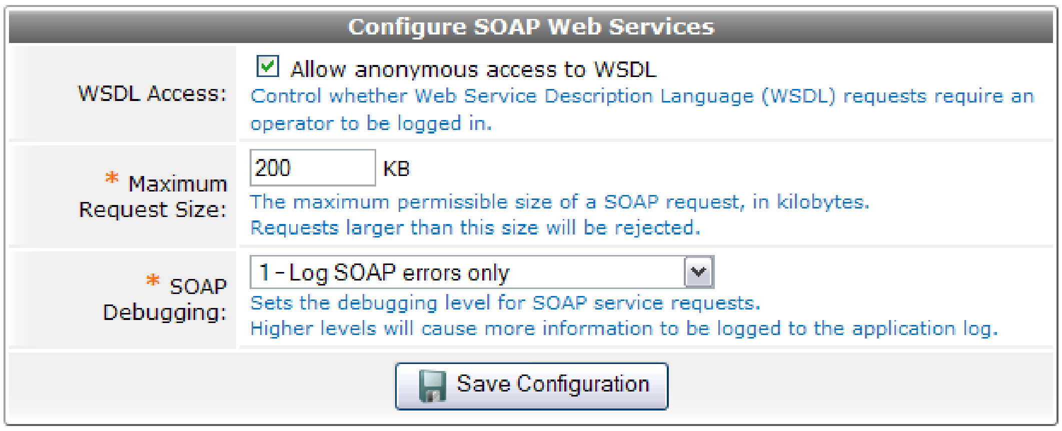

SOAP
Web Services
View a list of available web services and configure SOAP settings.

Use the **Web Services** command link available from the Administrator page, or navigate to **Administrator** > **Web Services**, to access the SOAP Web Services user interface.

2.3.2 Configuring SOAP Web Services


SOAP
Configure Web Services
Make changes to settings that are applicable to all SOAP web services.

Use the **Configure Web Services** command link to make changes to system settings affecting the SOAP API.



Configure SOAP Web Services

WSDL Access:	✓ Allow anonymous access to WSDL Control whether Web Service Description Language (WSDL) requests require an operator to be logged in.
* Maximum Request Size:	200 KB The maximum permissible size of a SOAP request, in kilobytes. Requests larger than this size will be rejected.
* SOAP Debugging:	1 - Log SOAP errors only Sets the debugging level for SOAP service requests. Higher levels will cause more information to be logged to the application log.

2.3.3 SOAP debugging

Select a higher level for the "SOAP Debugging" configuration option to log additional details to the application log.

The application log can be accessed by navigating to **Administrator** > **Plugin Manager** > **Application Log**.

At the highest debugging level of 4, every SOAP request and response will be logged including full HTTP headers and contents, which may be useful when trying to identify the exact cause of a problem.

2.3.4 Creating a SOAP API operator

The SOAP API requires both authentication and authorization components.

- **Authentication** means that suitable credentials must be provided via the HTTP "basic" access authentication method. A valid amigopod operator username and password must be provided.

- **Authorization** for the SOAP API requires that the corresponding user account has sufficient privileges to perform the requested operation. In the amigopod role-based access control system, this requires at a minimum that the SOAP API privilege is granted, as well as any additional privileges required for the operation requested.

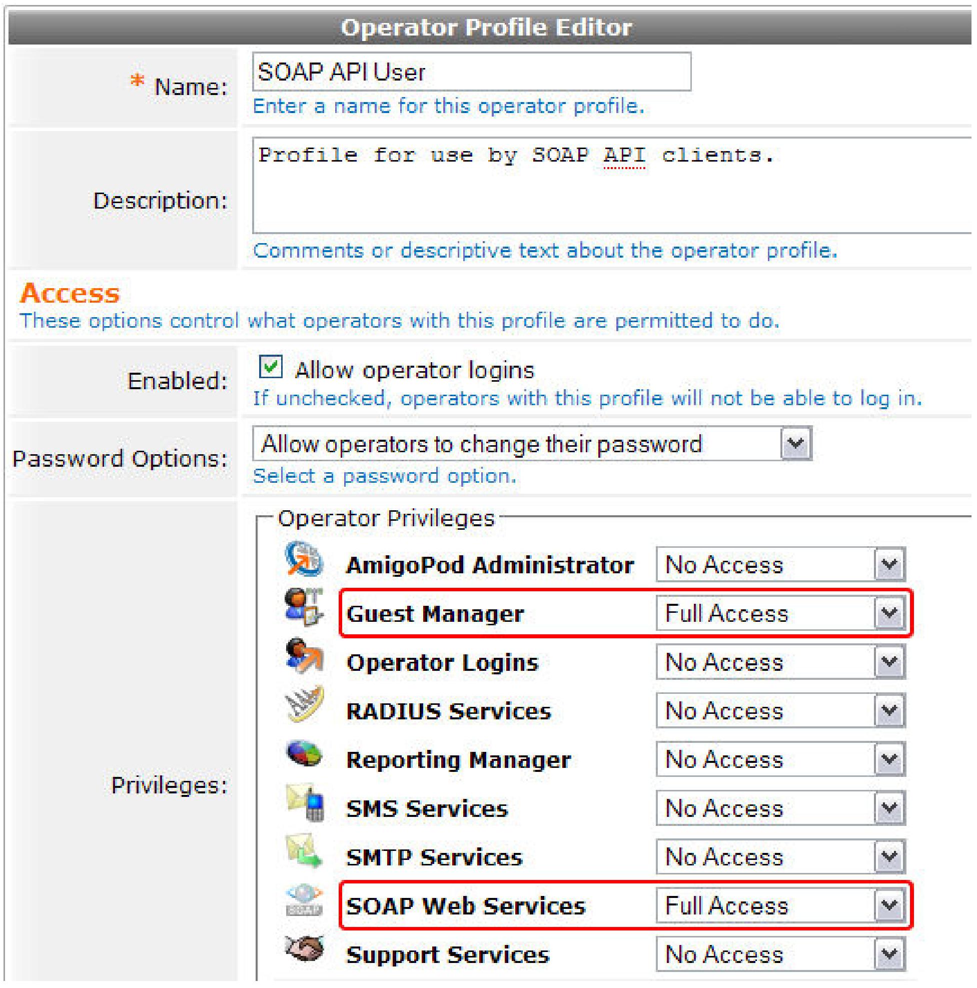
While the default administrative account will automatically gain SOAP API privileges, for security reasons it is strongly recommended that a specific operator profile be created for use by SOAP API clients.

To create a suitable operator profile, navigate to **Administrator** > **Operator Logins** > **Manage Profiles** and then click the **Create a new operator profile** link.

In the Privileges list, select either **Full Access** for the SOAP Web Services privilege, or choose **Custom…** and then select either **Read Only** or **Full** for the SOAP API privilege.

You should also select suitable permissions for the Guest Manager privilege, depending on the type of requests that will be made. For details on the privileges required by each operation, refer to section 0.

An example profile is shown below.



Operator Profile Editor

* Name:	SOAP API User Enter a name for this operator profile.
Description:	Profile for use by SOAP API clients. Comments or descriptive text about the operator profile.

Access
These options control what operators with this profile are permitted to do.

Enabled:	✓ Allow operator logins If unchecked, operators with this profile will not be able to log in.
Password Options:	Allow operators to change their password Select a password option.

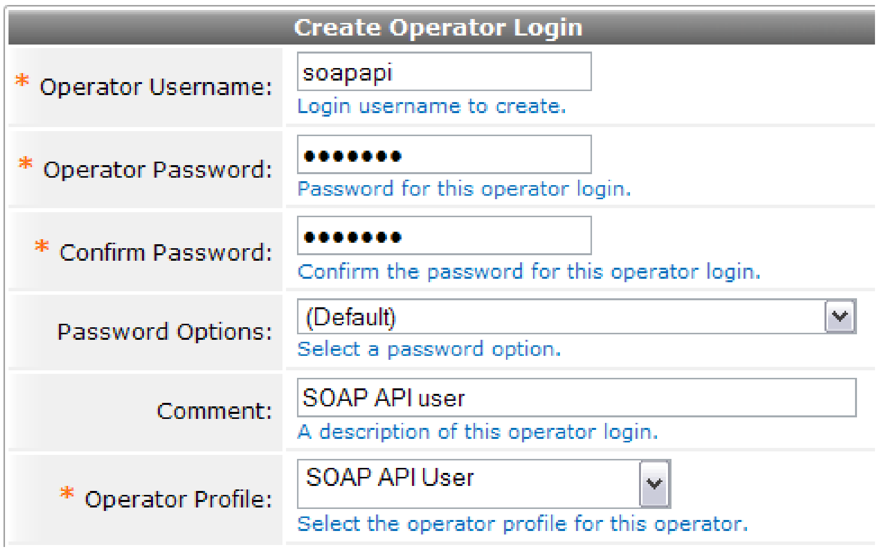
Privileges:

Operator Privileges

AmigoPod Administrator	No Access
Guest Manager	Full Access
Operator Logins	No Access
RADIUS Services	No Access
Reporting Manager	No Access
SMS Services	No Access
SMTP Services	No Access
SOAP Web Services	Full Access
Support Services	No Access

After you have created a suitable operator profile, navigate to **Administrator** > **Operator Logins** > **Manage Operators** and click the **Create operator login** link.

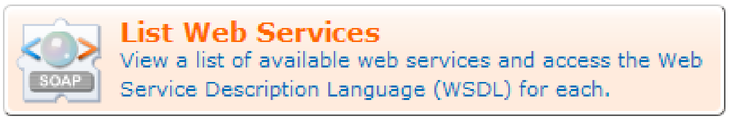
Select the newly created operator profile, and specify the username and password for the operator. An example operator login is shown below:



Create Operator Login

* Operator Username:	soapapi Login username to create.
* Operator Password:	●●●●●●● Password for this operator login.
* Confirm Password:	●●●●●●● Confirm the password for this operator login.
Password Options:	(Default) Select a password option.
Comment:	SOAP API user A description of this operator login.
* Operator Profile:	SOAP API User Select the operator profile for this operator.

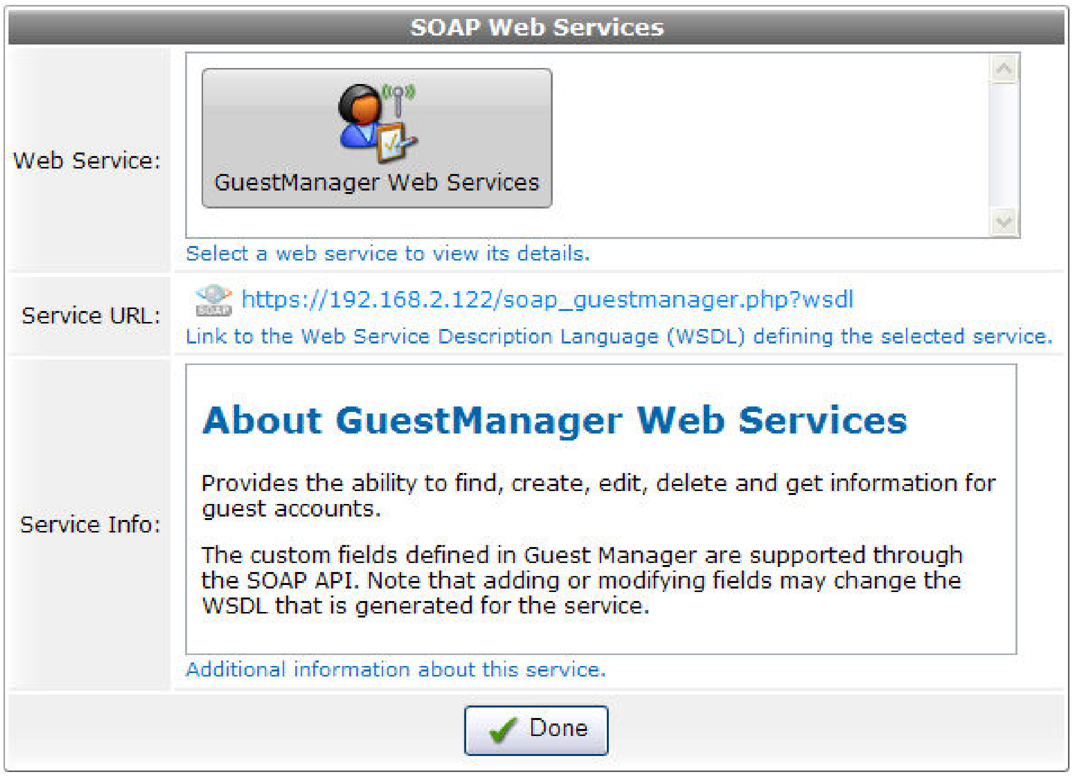
2.3.5 Accessing the WSDL



List Web Services
View a list of available web services and access the Web Service Description Language (WSDL) for each.

Use the **List Web Services** command link to browse the available web services and obtain additional details about each one.

Click the icon for GuestManager Web Services to view the Service URL and additional information about the service:


SOAP Web Services
Web Service:
GuestManager Web Services
Select a web service to view its details.
Service URL:
https://192.168.2.122/soap_guestmanager.php?wsdl
Link to the Web Service Description Language (WSDL) defining the selected service.
Service Info:
Additional information about this service.
Done

About GuestManager Web Services

Provides the ability to find, create, edit, delete and get information for guest accounts.

The custom fields defined in Guest Manager are supported through the SOAP API. Note that adding or modifying fields may change the WSDL that is generated for the service.

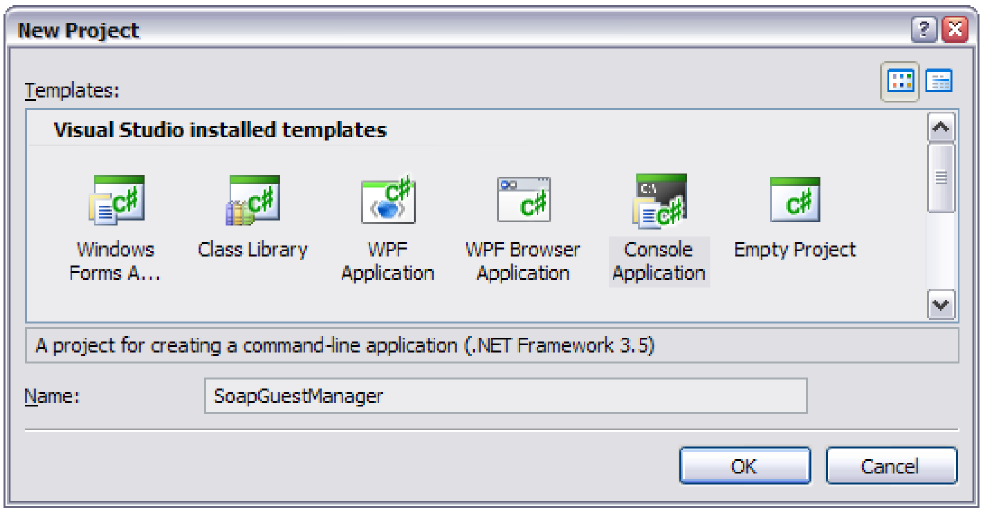
NOTE If the "Allow anonymous access to WSDL" option is specified in the SOAP Web Services configuration, accessing the WSDL through the specified Service URL does not require logging in to the amigopod GUI.

2.4 Integration Example

In this section, a simple console application will be developed using Microsoft Visual C# 2008 Express Edition. The "Add Service Reference" feature of this development tool will be used to automatically create a web service interface which can then be used from the code.

2.4.1 Create a new project

From the File menu, choose New Project.

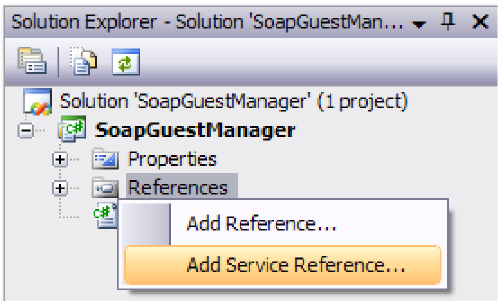


A project for creating a command-line application (.NET Framework 3.5)

Name: SoapGuestManager

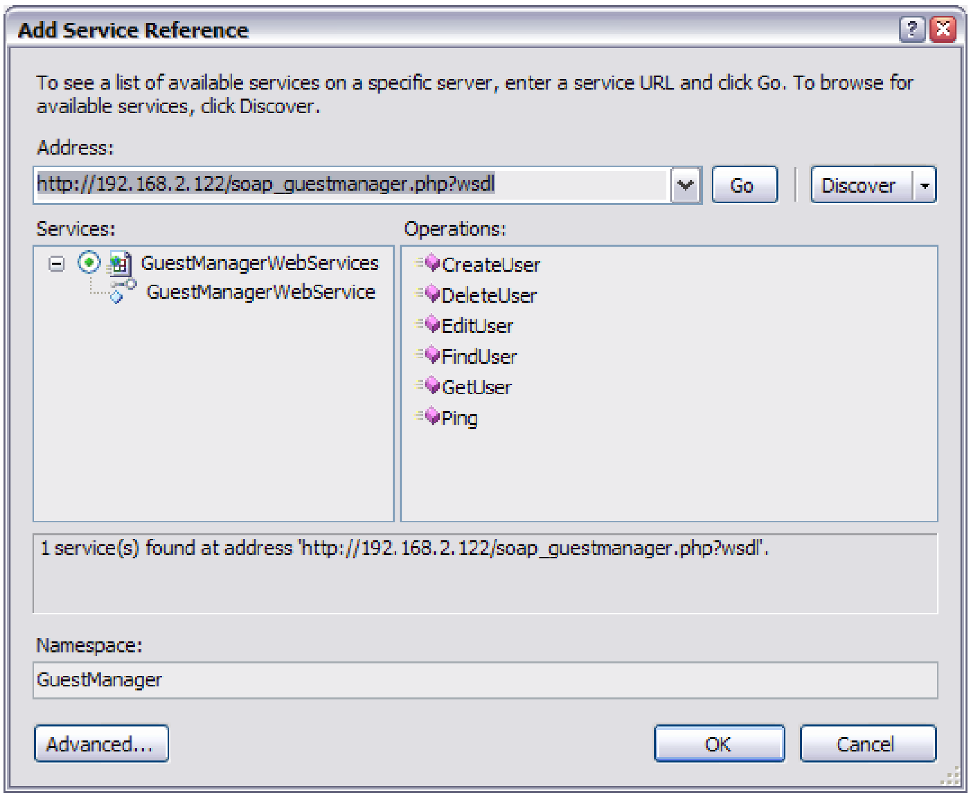
2.4.2 Add service reference

In the Solution Explorer, right-click the References folder, and click Add Service Reference.


Solution Explorer - Solution 'SoapGuestMan...
Solution 'SoapGuestManager' (1 project)
SoapGuestManager
Properties
References
Add Reference...
Add Service Reference...

The Add Service Reference dialog box appears. Enter the Service URL for the GuestManager Web Services into the Address box, and click the Go button.

The WSDL is downloaded, and a list of the web services and operations found is displayed.


Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://192.168.2.122/soap_guestmanager.php?wsdl

Services: GuestManagerWebServices, GuestManagerWebService

Operations: CreateUser, DeleteUser, EditUser, FindUser, GetUser, Ping

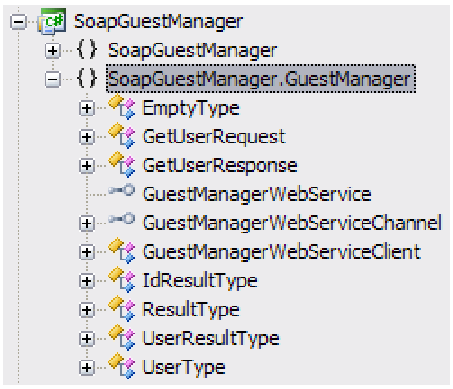
1 service(s) found at address 'http://192.168.2.122/soap_guestmanager.php?wsdl'.

Namespace: GuestManager

In the Namespace text field, type in a name. This name is used to organize the automatically generated code that interfaces with the web service.

Click the OK button to create the web service reference.

To browse the created classes, double-click the GuestManager service reference. The Object Browser will be displayed with the selected namespace highlighted.



2.4.3 Configuring HTTP Basic authentication

Performing a simple API call, such as the “Ping” operation described in section 3.4.6, can be used to verify that the web service is correctly configured and ready for use.

Because the SOAP API requires HTTP Basic authentication, ensure that you have a suitable operator profile and operator login credentials, as explained in section 2.3.4.

Configuring the web service reference to use authentication requires editing the app.config file to make two changes:

- The **mode** attribute of the `<security>` tag must be changed to “TransportCredentialOnly”.
- The **clientCredentialType** attribute of the `<transport>` tag must be changed to “Basic”.

The updated app.config file is shown below, with the appropriate changes highlighted.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="GuestManagerSoapBinding" closeTimeout="00:01:00"
                    openTimeout="00:01:00" receiveTimeout="00:10:00"
                    sendTimeout="00:01:00"
                    allowCookies="false" bypassProxyOnLocal="false"
                    hostNameComparisonMode="StrongWildcard"
                    maxBufferSize="65536" maxBufferPoolSize="524288"
                    maxReceivedMessageSize="65536"
                    messageEncoding="Text" textEncoding="utf-8"
                    transferMode="Buffered"
                    useDefaultWebProxy="true">
                    <readerQuotas maxDepth="32" maxStringContentLength="8192"
                        maxArrayLength="16384"
                        maxBytesPerRead="4096" maxNameTableCharCount="16384" />
                    <security mode="TransportCredentialOnly">
                        <transport clientCredentialType="Basic"
                            proxyCredentialType="None"
                            realm="" />
                        <message clientCredentialType="UserName"
```

```
                            algorithmSuite="Default" />
                    </security>
                </binding>
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="http://192.168.2.122/soap_service.php"
                binding="basicHttpBinding"
                bindingConfiguration="GuestManagerSoapBinding"
                contract="GuestManager.GuestManagerWebService"
                name="GuestManagerWebServicePort" />
        </client>
    </system.serviceModel>
</configuration>
```

2.4.4 Performing an API call

This section outlines the C# code required to use the web service.

First, add a using declaration for the namespace containing the web services:

```
using SoapGuestManager.GuestManager;
```

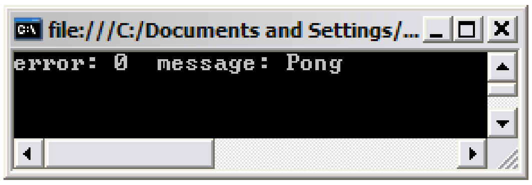
The following code can now be added to invoke the Ping operation and display the result.

```
// Create the client
GuestManagerWebServiceClient client = new GuestManagerWebServiceClient();
client.ClientCredentials.UserName.UserName = "soapapi";
client.ClientCredentials.UserName.Password = "bubbles";

// Perform a Ping operation
EmptyType pingRequest = new EmptyType();
ResultType pingResponse = client.Ping(pingRequest);

// Display the response
System.Console.Out.WriteLine("error: {0}  message: {1}",
    pingResponse.error, pingResponse.message);
```

When invoked, this performs the Ping operation and displays the following output:



```
error: 0  message: Pong
```

2.4.5 Securing web services using HTTPS

Because HTTP Basic authentication is insecure, it is strongly recommended that the HTTPS transport be used for all SOAP API calls.

To use HTTPS as the transport for SOAP API requests, the following changes should be made to the application configuration file:

- The **mode** attribute of the `<security>` tag must be changed to “Transport”.

- The **address** attribute of the `<endpoint>` tag must be changed to a URL including the “https:” prefix.

The updated app.config file is shown below, with the relevant changes highlighted.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <system.serviceModel>
        <bindings>
            <basicHttpBinding>
                <binding name="GuestManagerSoapBinding" ...>
                    <readerQuotas ... />
                    <security mode="Transport">
                        <transport clientCredentialType="Basic"
                            proxyCredentialType="None"
                            realm="" />
                        <message clientCredentialType="UserName"
                            algorithmSuite="Default" />
                    </security>
                </binding>
            </basicHttpBinding>
        </bindings>
        <client>
            <endpoint address="https://192.168.2.122/soap_service.php"
                binding="basicHttpBinding"
                bindingConfiguration="GuestManagerSoapBinding"
                contract="GuestManager.GuestManagerWebService"
                name="GuestManagerWebServicePort" />
        </client>
    </system.serviceModel>
</configuration>
```

Additionally, if a self-signed certificate is being used on the remote server, you will need to provide a suitable ServerCertificateValidationCallback implementation to validate the peer’s certificate.

The following code is a minimal implementation that accepts all server certificates without verification:

```
// Trust self-signed certificates
System.Net.ServicePointManager.ServerCertificateValidationCallback =
    ((sender, certificate, chain, sslPolicyErrors) => true);
```

NOTE In a production environment, it is strongly recommended that you deploy an SSL certificate that is signed by a trusted root CA known to all parties, and use the built-in server certificate validation procedures. This will ensure the security of the transaction cannot be compromised by a man-in-the-middle attack.

3 API Documentation

3.1 XML Namespaces

The XML namespace for the GuestManager Web Services is:
http://www.amigopod.com/go/GuestManager.wsdl

Additional XML namespaces that are referenced:

Component	XML Namespace
SOAP Envelope	http://schemas.xmlsoap.org/wsdl/soap/
SOAP Encoding	http://schemas.xmlsoap.org/soap/encoding/
WSDL	http://schemas.xmlsoap.org/wsdl/
XML Schema	http://www.w3.org/2001/XMLSchema

3.2 SOAP Addressing

3.2.1 Web service endpoint

The endpoint of the SOAP service is located at the relative URL: **soap_guestmanager.php**.

This path is relative to the full amigopod URL, which can be constructed using **http:** or **https:** and the fully-qualified domain name of the amigopod Visitor Management Appliance.

- Example: http://192.168.2.122/soap_guestmanager.php
- Example: https://192.168.2.122/soap_guestmanager.php (secure)

3.2.2 Web service definition

The WSDL for the web service may be accessed by appending **?wsdl** to the service URL.

- Example: http://192.168.2.122/soap_guestmanager.php?wsdl

3.3 Types

This section describes the types defined in the WSDL schema.

3.3.1 EmptyType

This type must be empty, i.e. containing zero child elements.

- Example:

```
<ping/>
```

3.3.2 ErrorFlagType

The error flag indicates if the operation completed successfully.

Only the values zero (0) and one (1) are supported.

- A successful operation is indicated with:

```
<error>0</error>
```

- A failed operation is indicated with:

```
<error>1</error>
```

3.3.3 IdResultType

Standard result type (see section 0), with an optional `<id>` element.

- Example:

```
<result>
    <error>0</error>
    <id>551</id>
</result>
```

- Example:

```
<result>
    <error>1</error>
    <message>This username is already in use</message>
</result>
```

3.3.4 IdType

Specifies a user ID. The user ID is a positive integer value, starting at 1.

- Example:

```
<id>551</id>
```

3.3.5 ResultType

Operations return a standard result type. The `<error>` flag indicates if the operation completed successfully. If the operation failed, the `<message>` contains a description of the error.

- Example of a successful operation:

```
<result>
    <error>0</error>
    <message/>
</result>
```

- Example of a successful operation with message:

```
<result>
    <error>0</error>
    <message>Pong</message>
</result>
```

- Example of an unsuccessful operation:

```
<result>
    <error>1</error>
    <message>This username is already in use</message>
```

```
</result>
```

3.3.6 UserResultType

Standard result type, with an optional `<user>` element.

- Example of a successful operation:

```
<result>
  <error>0</error>
  <message></message>
  <user>
    <creator_name>soapapi</creator_name>
    <do_expire>4</do_expire>
    <do_schedule>false</do_schedule>
    <enabled>true</enabled>
    <expire_time>2010-03-18T16:14:25+10:00</expire_time>
    <id>3</id>
    <role_id>2</role_id>
    <role_name>Guest</role_name>
    <schedule_time>1970-01-01T10:00:00+10:00</schedule_time>
    <simultaneous_use>1</simultaneous_use>
    <username>demo@example.com</username>
  </user>
</result>
```

- Example of an unsuccessful operation:

```
<result>
  <error>1</error>
  <message>Cannot find account with ID 3</message>
</result>
```

3.3.7 UserType

The User type defines a visitor account, which consists of a number of fields.

The fields available may be customized in amigopod Guest Manager. Navigate to **Guest Manager** > **Customization** > **Customize Fields** to create new fields or modify existing fields.

NOTE Adding or removing fields will update the UserType schema in the WSDL for GuestManager Web Services. Ensure that you update any clients using this WSDL if the fields are modified.

Each field of the visitor account corresponds to an XML element with the same name as the field.

All fields within the UserType schema are marked as optional; however, certain operations may require that particular fields are provided.

- Example of a user definition:

```
<user>
  <creator_name>soapapi</creator_name>
  <do_expire>4</do_expire>
  <do_schedule>false</do_schedule>
  <enabled>true</enabled>
```

```
    <expire_time>2010-03-18T16:14:25+10:00</expire_time>
    <id>3</id>
    <role_id>2</role_id>
    <role_name>Guest</role_name>
    <schedule_time>1970-01-01T10:00:00+10:00</schedule_time>
    <simultaneous_use>1</simultaneous_use>
    <username>demo@example.com</username>
  </user>
```

3.4 Operations

3.4.1 CreateUser

Creates a new user account.

Headers	
Name	**Value**
SOAPAction	`amigopod.guest.create`
Input Parameters (Request)	
Name	**Type**
`user`	`UserType` (§3.3.7) ▪ See “Notes” below
Output Parameters (Response)	
Name	**Type**
`result`	`IdResultType` (§3.3.3)
Access Control	
Privilege	**Value**
SOAP API	Full Access
Create New Guest Account	Full Access

Notes

- The standard business logic for visitor account creation applies to visitor accounts created with the SOAP API. For details, refer to the section “Business logic for account creation” in the amigopod Deployment Guide, or search for this term in the online help.
- Note that the **creator_accept_terms** field must be set to the Boolean value “true” in order to create an account.
- A value for the **role_id** field must be specified to create a visitor account. The SOAP API user must also be permitted to create visitor accounts with this role.

Examples

Example code implementing visitor account creation:

```
static string TestCreate(GuestManagerWebServiceClient client)
{
    System.Console.Out.WriteLine("Sending CreateUser request...");

    // Perform a CreateUser operation
    UserType createRequest = new UserType();
    createRequest.username = "demo@example.com";
    createRequest.creator_accept_terms = true;
    createRequest.creator_accept_termsSpecified = true;
    createRequest.expire_after = "12";
    createRequest.creator_name = client.ClientCredentials.UserName.UserName;
    createRequest.role_id = "2"; // Guest

    IdResultType createResponse = client.CreateUser(createRequest);

    // Display the response
    System.Console.Out.WriteLine("CreateUser response:  error: {0}  message: {1}",
        createResponse.error, createResponse.message);
    return createResponse.id;
}
```

Example request for CreateUser:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <CreateUser xmlns="http://www.amigopod.com/go/GuestManager.wsdl">
      <user xmlns="">
        <creator_accept_terms>true</creator_accept_terms>
        <creator_name>soapapi</creator_name>
        <expire_after>12</expire_after>
        <role_id>2</role_id>
        <username>demo@example.com</username>
      </user>
    </CreateUser>
  </s:Body>
</s:Envelope>
```

Successful response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:CreateUserResponse>
      <result>
        <error>0</error>
        <message>Created guest account for demo@example.com</message>
        <id>1</id>
      </result>
    </ns1:CreateUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Failure response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:CreateUserResponse>
```

```
      <result>
        <error>1</error>
        <message>This username is already in use</message>
      </result>
    </ns1:CreateUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4.2 DeleteUser

Deletes a user account by ID or matching fields.

Headers	
Name	**Value**
SOAPAction	`amigopod.guest.delete`
Input Parameters (Request)	
Name	**Type**
`user`	`UserType` (§3.3.7) ▪ See “Notes” below
Output Parameters (Response)	
Name	**Type**
`result`	`ResultType` (§0)
Access Control	
Privilege	**Value**
SOAP API	Full Access
Remove Accounts	Full Access
Notes	

- This operation deletes a single visitor account that matches all of the field values specified in the `user` parameter.
- Exactly one account must match; if more than one match is found, or if no match is found, an error will be returned and no visitor accounts will be deleted.

Examples

Example code implementing visitor account deletion:

```
static void TestDelete(GuestManagerWebServiceClient client, UserType user)
{
    System.Console.Out.WriteLine("Sending DeleteUser request...");

    // Perform a DeleteUser operation
    ResultType deleteResponse = client.DeleteUser(user);

    // Display the response
    System.Console.Out.WriteLine("DeleteUser response:  error: {0}  message: {1}",
        deleteResponse.error, deleteResponse.message);
}
```

Example request for DeleteUser:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <DeleteUser xmlns="http://www.amigopod.com/go/GuestManager.wsdl">
      <user xmlns="">
        <id>6</id>
      </user>
    </DeleteUser>
  </s:Body>
</s:Envelope>
```

Successful response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:DeleteUserResponse>
      <result>
        <error>0</error>
        <message>Deleted guest account demo@example.com. User has no active
connections to disconnect.</message>
      </result>
    </ns1:DeleteUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Failure response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:DeleteUserResponse>
      <result>
        <error>1</error>
        <message>Cannot find account with ID 3</message>
      </result>
    </ns1:DeleteUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4.3 EditUser

Modifies properties of a user account by ID.

Headers		
	Name	**Value**
	SOAPAction	`amigopod.guest.edit`
Input Parameters (Request)		
	Name	**Type**
	`user`	`UserType` (§3.3.7) ▪ See “Notes” below

Output Parameters (Response)		
	Name	**Type**
	`result`	`ResultType` (§0)

Access Control		
	Privilege	**Value**
	SOAP API	Full Access
	Remove Accounts	Full Access

Notes

- This operation modifies the properties of a visitor account to match the field values specified in the `user` parameter.
- The **id** field must be specified to indicate the ID of the visitor account to modify. This field is assigned by the system when the visitor account is created and cannot be changed.

Examples

Example code implementing visitor account modification:

```
static void TestEdit(GuestManagerWebServiceClient client, UserType user)
{
    System.Console.Out.WriteLine("Sending EditUser request...");

    // Perform an EditUser operation
    ResultType editResponse = client.EditUser(user);

    // Display the response
    System.Console.Out.WriteLine("EditUser response:  error: {0}  message: {1}",
        editResponse.error, editResponse.message);
}
```

Example request for EditUser:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <EditUser xmlns="http://www.amigopod.com/go/GuestManager.wsdl">
      <user xmlns="">
        <creator_name>soapapi</creator_name>
        <do_expire>4</do_expire>
        <do_schedule>false</do_schedule>
        <enabled>true</enabled>
        <expire_time>2010-03-18T18:29:50+10:00</expire_time>
        <id>5</id>
        <role_id>2</role_id>
        <role_name>Guest</role_name>
        <schedule_time>1970-01-01T10:00:00+10:00</schedule_time>
        <simultaneous_use>1</simultaneous_use>
        <username>demo@example.com</username>
      </user>
    </EditUser>
  </s:Body>
```

```
</s:Envelope>
```

Successful response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:EditUserResponse>
      <result>
        <error>0</error>
        <message>
          Updated user account demo@example.com in the database&lt;br /&gt;
        </message>
      </result>
    </ns1:EditUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Failure response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:EditUserResponse>
      <result>
        <error>1</error>
        <message>Invalid UserType: id must be specified</message>
      </result>
    </ns1:EditUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4.4 FindUser

Returns properties of a user account by matching fields.

Headers		
	Name	**Value**
	SOAPAction	`amigopod.guest.find`
Input Parameters (Request)		
	Name	**Type**
	`user`	`UserType` (§3.3.7)
Output Parameters (Response)		
	Name	**Type**
	`result`	`UserResultType` (§0)
Access Control		
	Privilege	**Value**
	SOAP API	Read Only Access
	List Guest Accounts	Read Only Access

Notes

- This operation locates a single visitor account that matches all of the field values specified in the `user` parameter.
- Exactly one account must match; if more than one match is found, or if no match is found, an error will be returned.
- If a visitor account was found, its properties will be returned in the `<user>` element of the result.

Examples

Example code implementing search for a visitor account based on username:

```
static UserType TestFind(GuestManagerWebServiceClient client, string username)
{
    System.Console.Out.WriteLine("Sending FindUser request for {0}...",
        username);

    // Perform a FindUser operation
    UserType findRequest = new UserType();
    findRequest.username = username;
    UserResultType findResponse = client.FindUser(findRequest);

    // Display the response
    System.Console.Out.WriteLine("FindUser response:  error: {0}  message: {1}",
        findResponse.error, findResponse.message);
    if (findResponse.user != null)
    {
        System.Console.Out.WriteLine("  id: {0}", findResponse.user.id);
    }
    return findResponse.user;
}
```

Example request for FindUser:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <FindUser xmlns="http://www.amigopod.com/go/GuestManager.wsdl">
      <user xmlns="">
        <username>demo@example.com</username>
      </user>
    </FindUser>
  </s:Body>
</s:Envelope>
```

Successful response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:FindUserResponse>
      <result>
        <error>0</error>
        <message></message>
        <user>
          <creator_name>soapapi</creator_name>
          <do_expire>4</do_expire>
          <do_schedule>false</do_schedule>
          <enabled>true</enabled>
          <expire_time>2010-03-18T16:30:59+10:00</expire_time>
          <id>4</id>
          <role_id>2</role_id>
          <role_name>Guest</role_name>
          <schedule_time>1970-01-01T10:00:00+10:00</schedule_time>
          <simultaneous_use>1</simultaneous_use>
          <username>demo@example.com</username>
        </user>
      </result>
    </ns1:FindUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Failure response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:FindUserResponse>
      <result>
        <error>1</error>
        <message>No user account found</message>
        <user/>
      </result>
    </ns1:FindUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4.5 GetUser

Returns properties of a user account by ID.

Headers		
	Name	**Value**
	SOAPAction	`amigopod.guest.get`
Input Parameters (Request)		
	Name	**Type**
	`id`	`IdType` (§3.3.4)
Output Parameters (Response)		
	Name	**Type**
	`result`	`UserResultType` (§0)
Access Control		
	Privilege	**Value**
	SOAP API	Read Only Access
	List Guest Accounts	Read Only Access

Notes

- Returns a `<user>` element corresponding to the visitor account with the specified ID.
- If the specified ID is invalid, no `<user>` element is returned and the `<error>` flag is set to 1.

Examples

Example code implementing a guest lookup operation:

```
static UserType TestGet(GuestManagerWebServiceClient client, string id)
{
    System.Console.Out.WriteLine("Sending GetUser request for ID {0}...", id);

    // Perform a GetUser operation
    UserResultType getResponse = client.GetUser(id);

    // Display the response
    System.Console.Out.WriteLine("GetUser response:  error: {0}  message: {1}",
        getResponse.error, getResponse.message);
    if (getResponse.user != null)
    {
        System.Console.Out.WriteLine("  username: {0}", getResponse.user.username);
    }
    return getResponse.user;
}
```

Example request for GetUser:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <GetUser xmlns="http://www.amigopod.com/go/GuestManager.wsdl">
      <id xmlns="">3</id>
    </GetUser>
  </s:Body>
</s:Envelope>
```

Successful response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:GetUserResponse>
      <result>
        <error>0</error>
        <message></message>
        <user>
          <creator_name>soapapi</creator_name>
          <do_expire>4</do_expire>
          <do_schedule>false</do_schedule>
          <enabled>true</enabled>
          <expire_time>2010-03-18T16:14:25+10:00</expire_time>
          <id>3</id>
          <role_id>2</role_id>
          <role_name>Guest</role_name>
          <schedule_time>1970-01-01T10:00:00+10:00</schedule_time>
          <simultaneous_use>1</simultaneous_use>
          <username>demo@example.com</username>
        </user>
      </result>
    </ns1:GetUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Failure response – e.g., user ID not found:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <SOAP-ENV:Body>
    <ns1:GetUserResponse>
      <result xsi:type="ns1:ResultType">
        <error>1</error>
        <message>Cannot find account with ID 3</message>
      </result>
    </ns1:GetUserResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4.6 Ping

Checks that the SOAP server is alive.

Headers		
	Name	**Value**
	SOAPAction	`amigopod.guest.ping`
Input Parameters (Request)		
	Name	**Type**
	`ping`	`EmptyType` (§3.3.1)
Output Parameters (Response)		
	Name	**Type**
	`result`	`ResultType` (§0)
Access Control		
	Privilege	**Value**
	SOAP API	Read Only Access
Notes		

- Returns a standard result type with the **message** set to “Pong”.

Examples

Example code implementing a ping test operation:

```
static void TestPing(GuestManagerWebServiceClient client)
```

```
{
    System.Console.Out.WriteLine("Sending Ping request...");

    // Perform a Ping operation
    EmptyType pingRequest = new EmptyType();
    ResultType pingResponse = client.Ping(pingRequest);

    // Display the response
    System.Console.Out.WriteLine("Ping response:  error: {0}  message: {1}",
        pingResponse.error, pingResponse.message);
}
```

Example request for Ping:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <Ping xmlns="http://www.amigopod.com/go/GuestManager.wsdl">
      <ping xmlns=""/>
    </Ping>
  </s:Body>
</s:Envelope>
```

Successful response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://www.amigopod.com/go/GuestManager.wsdl">
  <SOAP-ENV:Body>
    <ns1:PingResponse>
      <result>
        <error>0</error>
        <message>Pong</message>
      </result>
    </ns1:PingResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```