

Ada Web Server User's Guide

AWS - version 2.3.0w

Support for SOAP 1.1 - version 1.4.0

SMTP, POP, LDAP and Jabber protocols.

Document revision level \$Revision: 9419 \$

Date: 5 April 2007

AdaCore

Copyright © 2000, Pascal Obry

Copyright © 2001, Pascal Obry, Dmitriy Anisimkov

Copyright © 2002-2005, AdaCore

This document may be copied, in whole or in part, in any form or by any means, as is or with alterations, provided that (1) alterations are clearly marked as alterations and (2) this copyright notice is included unmodified in any copy.

Table of Contents

1	Introduction	1
2	Building AWS	3
2.1	Requirements	3
2.2	AWS.Net.Std	3
2.3	Building	4
2.4	Demos	4
2.5	Installing	6
3	Using AWS	7
3.1	Setting up environment	7
3.1.1	Using environment variables	7
3.1.2	Using GNAT Project Files	7
3.2	Basic notions	8
3.2.1	Building an AWS server	8
3.2.2	Callback procedure	10
3.2.3	Form parameters	11
3.2.4	Distribution of an AWS server	12
3.3	Building answers	13
3.3.1	Redirection	13
3.3.2	New location for a page	13
3.3.3	Authentication required	13
3.3.4	Sending back an error message	13
3.3.5	Response from a string	13
3.3.6	Response from a Stream_Element_Array	13
3.3.7	Response from a file	14
3.3.8	Response from a stream	14
3.3.9	Response from a on-disk stream	15
3.3.10	Response from a on-disk once stream	15
3.3.11	Response from a memory stream	15
3.3.12	Response from a compressed memory stream	15
3.3.13	Split page	15
3.4	Configuration options	16
3.5	Session handling	19
3.6	Authentication	20
3.7	File upload	21
3.8	Communication	21
3.8.1	Communication - client side	22
3.8.2	Communication - server side	22
3.9	Hotplug module	22
3.9.1	Hotplug module - server activation	23
3.9.2	Hotplug module - creation	23
3.10	Server Push	24
3.11	Working with Server sockets	25
3.12	Server Log	25
3.13	Secure server	27
3.13.1	Initialization	27

3.13.2	Creating a test certificate	27
3.13.3	Protocol	27
3.14	Unexpected exception handler	28
3.15	Socket log	29
3.16	Client side	29
4	High level services	31
4.1	Directory browser	31
4.2	Dispatchers	31
4.2.1	Callback dispatcher	31
4.2.2	Method dispatcher	31
4.2.3	URI dispatcher	31
4.2.4	Virtual host dispatcher	31
4.2.5	Transient pages dispatcher	32
4.2.6	Timer dispatcher	32
4.2.7	Linker dispatcher	32
4.2.8	SOAP dispatcher	32
4.3	Static Page server	33
4.4	Transient Pages	33
4.5	Split pages	34
4.6	Download Manager	34
4.7	Web Elements	36
4.7.1	Installation	36
4.7.2	Ajax	36
4.7.2.1	Steps to do Ajax	37
4.7.2.2	Basic Ajax support	37
4.7.2.3	XML based Ajax	38
4.7.2.4	Advanced Ajax	42
5	Using SOAP	43
5.1	SOAP Client	43
5.2	SOAP Server	44
5.2.1	Step by step instructions	44
5.2.2	SOAP helpers	45
6	Using WSDL	47
6.1	Creating WSDL documents	47
6.1.1	Using ada2wsdl	47
6.1.2	Ada mapping to WSDL	49
6.1.3	ada2wsdl	52
6.1.4	'ada2wsdl' limitations	52
6.2	Working with WSDL documents	52
6.2.1	Client side (stub)	52
6.2.2	Server side (skeleton)	53
6.2.3	wsdl2aws	54
6.2.4	wsdl2aws behind the scene	56
6.2.5	wsdl2aws limitations	56
6.3	Using ada2wsdl and wsdl2aws together	56
7	Working with mails	59
7.1	Sending e-mail	59
7.2	Retrieving e-mail	59

8	LDAP	63
9	Jabber	65
9.1	Jabber presence	65
9.2	Jabber message	65
10	Resources	67
10.1	Building resources	67
10.2	Using resources	67
10.3	Stream resources	67
10.4	awsres tool	67
11	Status page	69
Appendix A	References	73
Appendix B	AWS API Reference	77
B.1	AWS	77
B.2	AWS.Attachments	78
B.3	AWS.Client	80
B.4	AWS.Client.Hotplug	86
B.5	AWS.Communication	87
B.6	AWS.Communication.Client	88
B.7	AWS.Communication.Server	89
B.8	AWS.Config	90
B.9	AWS.Config.Ini	96
B.10	AWS.Config.Set	97
B.11	AWS.Containers.Tables	102
B.12	AWS.Default	104
B.13	AWS.Dispatchers	107
B.14	AWS.Dispatchers.Callback	109
B.15	AWS.Exceptions	110
B.16	AWS.Headers	112
B.17	AWS.Headers.Values	114
B.18	AWS.Jabber	116
B.19	AWS.LDAP.Client	118
B.20	AWS.Log	123
B.21	AWS.Messages	126
B.22	AWS.MIME	131
B.23	AWS.Net	134
B.24	AWS.Net.Buffered	139
B.25	AWS.Net.Log	141
B.26	AWS.Net.Log.Callbacks	143
B.27	AWS.Net.SSL	145
B.28	AWS.Net.SSL.Certificate	148
B.29	AWS.Parameters	149
B.30	AWS.POP	150
B.31	AWS.Resources	154
B.32	AWS.Resources.Embedded	157
B.33	AWS.Resources.Files	159
B.34	AWS.Resources.Streams	160
B.35	AWS.Resources.Streams.Disk	162

B.36	AWS.Resources.Streams.Disk.Once	164
B.37	AWS.Resources.Streams.Memory	165
B.38	AWS.Resources.Streams.Memory.ZLib	167
B.39	AWS.Response	169
B.40	AWS.Server	175
B.41	AWS.Server.Hotplug	179
B.42	AWS.Server.Log	181
B.43	AWS.Server.Push	183
B.44	AWS.Server.Status	187
B.45	AWS.Services.Callbacks	189
B.46	AWS.Services.Directory	190
B.47	AWS.Services.Dispatchers	193
B.48	AWS.Services.Dispatchers.Linker	195
B.49	AWS.Services.Dispatchers.Method	196
B.50	AWS.Services.Dispatchers.URI	198
B.51	AWS.Services.Dispatchers.Virtual_Host	200
B.52	AWS.Services.Download	202
B.53	AWS.Services.Page_Server	204
B.54	AWS.Services.Split_Pages	205
B.55	AWS.Services.Split_Pages.Alpha	207
B.56	AWS.Services.Split_Pages.Alpha.Bounded	209
B.57	AWS.Services.Split_Pages.Uniform	211
B.58	AWS.Services.Split_Pages.Uniform.Alpha	213
B.59	AWS.Services.Split_Pages.Uniform.Overlapping	215
B.60	AWS.Services.Transient_Pages	216
B.61	AWS.Session	217
B.62	AWS.SMTP	221
B.63	AWS.SMTP.Client	224
B.64	AWS.Status	227
B.65	AWS.Templates	232
B.66	AWS.Translator	233
B.67	AWS.URL	235
B.68	SOAP	238
B.69	SOAP.Client	239
B.70	SOAP.Dispatchers	240
B.71	SOAP.Dispatchers.Callback	242
B.72	SOAP.Message	243
B.73	SOAP.Message.XML	245
B.74	SOAP.Parameters	246
B.75	SOAP.Types	249
Index		259

1 Introduction

AWS stand for *Ada Web Server*. It is an Ada implementation of the HTTP/1.1 protocol as defined in the RFC 2616 from June 1999.

The goal is not to build a full Web server but more to make it possible to use a Web browser (like Internet Explorer, or Netscape Navigator) to control an Ada application. As we'll see later it is also possible to have two Ada programs exchange informations via the HTTP protocol. This is possible as **AWS** also implement the client side of the HTTP protocol.

Moreover with this library it is possible to have more than one server in a single application. It is then possible to export different kind of services by using different HTTP ports, or to have different ports for different services priority. Client which must be served with a very high priority can be assigned a specific port for example.

As designed, **AWS** big difference with a standard CGI server is that there is only one executable. A CGI server has one executable for each request or so, this becomes a pain to build and to distribute when the project gets bigger. We will also see that it is easier with **AWS** to deal with session data.

AWS support also HTTPS (secure HTTP) using SSL. This is based on **OpenSSL** a very good and Open Source SSL implementation.

Major supported features are:

- HTTP implementation
- HTTPS (Secure HTTP) implementation based on SSLv3
- Template Web pages (separate the code and the design)
- Web Services - SOAP based
- WSDL support (generate stub/skeleton from WSDL documents)
- Basic and Digest authentication
- Transparent session handling
- File upload
- Server push
- SMTP / POP (client API)
- LDAP (client API)
- Embedded resources (full self dependant Web server)
- Complete client API, including HTTPS
- Web server activity log

2 Building AWS

2.1 Requirements

AWS has been mainly developed with GNAT on Windows. It is built and tested regularly on GNU/Linux and Solaris, it should be fairly portable across platforms. To build AWS you need:

- GNU/Ada (GNAT compiler) ;

To build this version you need at least GNAT GAP 2006 Edition, GNAT GPL 2006 Edition or GNAT Pro 5.04a1 as the Ada 2005 Ada.Containers is used. The code should be fairly portable but has never been tested on another compiler than GNAT.

- a socket binding (**optional**) ;

GNAT.Sockets is the default socket implementation used. It is possible to use AdaSockets (AdaSockets was used in AWS v 1.2) and in this case you must install the AdaSockets package.

On Windows you must use the one distributed at

<http://www.obry.net/contrib> which is a Win32 port (done by Dmitriy Anisimkov) of the ENST version.

On UNIX you must use the socket binding from ENST

<http://www.rfc1149.net/devel/adasockets>.

- OpenSSL (**optional**) ;

OpenSSL is an Open Source toolkit implementing the *Secure Sockets Layer* (SSL v2 and v3) and much more. You'll find libraries for Win32 into this distribution. For other platforms just download the OpenSSL source distribution from <http://www.openssl.org> and build it.

- OpenLDAP (**optional**) ;

OpenLDAP is an Open Source toolkit implementing the *Lightweight Directory Access Protocol*. If you want to use the AWS/LDAP API on UNIX based systems, you need to install properly the OpenLDAP package. On Windows you don't need to install it as the 'libldap.a' library will be built by AWS and will use the standard Windows LDAP DLL 'wldap32.dll'.

You can download OpenLDAP from <http://www.openldap.org>.

2.2 AWS.Net.Std

This package is the standard (non-SSL) socket implementation. It exists different implementation of this package:

GNAT implementation

Version based on GNAT.Sockets, is GNAT dependent and will run on whatever OS GNAT has be ported to. This is the **default implementation** used. To select this implementation just do:

```
$ make gnatsockets
```

AdaSockets implementation

Version using AdaSockets packages. This was the default implementation used up to AWS 1.2. To select this implementation just do:

```
$ make adasockets
```

2.3 Building

Before building be sure to edit `'makefile.conf'`, this file contains many settings important for the build. Note that it is important to run `make setup` each time you edit this file.

When you have built and configured all external libraries you must set the `ADA_PROJECT_PATH` variable to point to the GNAT Project files for the different packages. For XML/Ada support, you also need to set `XMLADA` to `true` in `'makefile.conf'`.

At this point you can build AWS with:

```
$ make setup build
```

Note that by default AWS demos will be built without SSL support except on Windows. If you want to build the demos with SSL on UNIX (in this case you must have `'libssl.a'` and `'libcrypto.a'` available on your platform), open `'makefile.conf'` and set the `SOCKET` variable to `openssl`. Then rebuild with:

```
$ make setup build
```

It is possible to build AWS in debug mode by setting `DEBUG` make's variable in `'makefile.conf'`, or just:

```
$ make DEBUG=1 setup build
```

Note that by default AWS is configured to use the GNAT compiler. So, if you use GNAT you can build AWS just with:

```
$ make setup build
```

If you want to build only the AWS libraries and tools and do not want to build the demos you can set `EXTRA_MODULES` to `""` as in:

```
$ make EXTRA_MODULES="" build
```

2.4 Demos

During the build, the AWS library and demos will be compiled. The demos are a good way to learn how to use AWS. Yet to learn all features in AWS you'll need to read the documentation.

Here are a short description of the demos:

`'agent'`

A program using the AWS client interface. This simple tool can be used to retrieve Web page content. It supports passing through a proxy with authentication and basic authentication on the Web site.

`'auth'`

A simple program to test the Web Basic and Digest authentication feature.

`'com_1'`

`'com_2'`

Two simple programs that use the AWS communication service.

<code>'dispatch'</code>	A simple demo using the dispatcher facility. see Section 4.2.3 [URI dispatcher] , page 31 .
<code>'hello_world'</code>	The famous Hello World program. This is a server that will always return a Web page saying "Hello World!".
<code>'main'</code>	
<code>'hotplug'</code>	A simple test for the hotplug feature.
<code>'res_demo'</code>	A demo using the resource feature. This Web Server embedded a PNG image and an HTML page. The executable is self contained.
<code>'runme'</code>	A complete example that test many AWS features. If this test runs fine on your platform, AWS is certainly ok.
<code>'split'</code>	A demo for the transient pages and page splitter AWS's feature. Here a very big table is split on multiple pages. A set of links can be used to navigate to the next or previous page or to access directly to a given page.
<code>'test_jabber'</code>	A simple Jabber command line client to check the presence of a JID (Jabber ID). This uses the Jabber API, see Section B.18 [AWS.Jabber] , page 116 .
<code>'test_ldap'</code>	A simple LDAP demo which access a public LDAP server and display some information.
<code>'text_input'</code>	A simple demo which handle textarea and display the content.
<code>'vh_demo'</code>	Two servers on the same machine... virtual hosting demo. see Section 4.2.4 [Virtual host dispatcher] , page 31 .
<code>'web_elements'</code>	A driver to browse the Web Elements library and see some examples.
<code>'web_mail'</code>	A simple Web Mail implementation that works on a POP mailbox.
<code>'ws'</code>	A static Web page server and push enabled server.
<code>'wps'</code>	A very simple static Web page server based on <code>AWS.Services.Page_Server</code> . see Section 4.3 [Static Page server] , page 33 .
<code>'zdemo'</code>	A simple demo of the Gzip content encoding feature.

If XML/Ada is installed it is possible to build the SOAP binding and the SOAP demos, for this just type:

```
$ make build_soap
```

There is four demos:

‘soap_client’

‘soap_server’

A simple client/server program to test the SOAP protocol.

‘soap_svs’

A server that implements seven SOAP procedures for testing purpose.

‘soap_cvs’

The client SOAP program that test all seven SOAP procedure of the above server.

In each case you'll certainly have to edit the makefile to correctly set the include path for the libraries OpenSSL, Socket and XML/Ada. For more information, look at the makefiles.

2.5 Installing

When the build is done you must install AWS at a specific location. The target directory is defined with the *INSTALL* ‘makefile.conf’ variable. The default value is your home directory. To install:

```
$ make install
```

To install AWS into another directory you can either edit ‘makefile.conf’ and set *INSTALL* to the directory you like to install AWS or just force the make *INSTALL* variable:

```
$ make INSTALL=/opt install
```

Alternatively, with GNAT 5.03 and above it is possible to install AWS into the GNAT Standard Library location. In this case AWS is ready-to-use as there is no need to set *ADA_PROJECT_PATH*, just set *INSTALL* to point to GNAT root directory:

```
$ make INSTALL=/opt/gnatpro/5.03a install
```

Now you are ready to use AWS !

3 Using AWS

3.1 Setting up environment

3.1.1 Using environment variables

After installing AWS you must set the build environment to point the compiler to the right libraries. First let's say that AWS has been installed in 'awsroot' directory.

Following are the instructions to set the environment yourself. Note that the preferred solution is to use project files. In this case there is no manual configuration.

spec files

The spec files are installed in '<awsroot>/include/aws'. Add this path into `ADA_INCLUDE_PATH` or put it on the command line `-aI<awsroot>/include/aws`.

components

AWS uses some components they are installed in '<awsroot>/include/aws/components'. Add this path into `ADA_INCLUDE_PATH` or put it on the command line `-I<awsroot>/include/aws/components`.

libraries

The GNAT library files ('.ali') and the AWS libraries ('libaws.a') are installed into '<awsroot>/lib/aws'. Add this path into `ADA_OBJECTS_PATH` or put it on the command line `-aO<awsroot>/lib/aws`. Furthermore for `gnatlink` to find the libraries you must add the following library path option on the `gnatmake` command line `-largs -L<awsroot>/lib/aws -laws`.

Note that to build SSL applications you need to add `-lssl -lcrypto` on `gnatmake`'s `-largs` section.

external libraries

You must do the same thing for all external libraries that you will be using. For example you'll have to point GNAT to the `AdaSockets` libraries and pass `-ladasockets` on the `gnatmake`'s `-largs` section. The same apply if you have built AWS with SSL support.

3.1.2 Using GNAT Project Files

The best solution is to use the installed GNAT Project File 'aws.gpr'. This is supported only for GNAT 5.01 or above. You must have installed XML/Ada with project file support too.

If this is the case just set the `ADA_PROJECT_PATH` variable to point to the AWS and XML/Ada install directories. From there you just have to with the AWS project file in your GNAT Project file, nothing else to set.

```
with "aws";

project Simple is

  for Main use ("prog.adb");

  for Source_Dirs use ("src");

  for Object_Dir use "obj";

end Simple;
```

If your application must support SSL, you have to link against the ‘-lcrypto’ and ‘-lssl’ libraries. Add a package `Linker` in your project file, something like:

```
package Linker is
    for Default_Switches ("Ada") use ("-lcrypto", "-lssl");
end Linker;
```

See the *GNAT User's Guide* for more information about GNAT Project Files.

3.2 Basic notions

AWS is not a Web Server like *IIS* or *Apache*, it is a component to embedded HTTP protocol in an application. It means that it is possible to build an application which can also answer to a standard browser like *Internet Explorer* or *Netscape Navigator*. Since AWS provides support client and server HTTP protocol, applications can communicate through the HTTP channel. This give a way to build distributed applications, See [Section B.3 \[AWS.Client\]](#), page 80.

An application using AWS can open many HTTP channels. Each channel will use a specific port. For example, it is possible to embedded many HTTP and/or many HTTPS channels in the same application.

3.2.1 Building an AWS server

To build a server you must:

1. declare the HTTP Web Server

```
WS : AWS.Server.HTTP;
```

2. Start the server

You need to start the server before using it. This is done by calling `AWS.Server.Start` (See [Section B.40 \[AWS.Server\]](#), page 175.)

```
procedure Start
(Web_Server      : in out HTTP;
 Name           : in   String;
 Callback       : in   Response.Callback;
 Max_Connection : in   Positive := Def_Max_Connect;
 Admin_URI      : in   String   := Def_Admin_URI;
 Port          : in   Positive := Def_Port;
 Security       : in   Boolean   := False;
 Session        : in   Boolean   := False;
 Case_Sensitive_Parameters : in Boolean := True;
 Upload_Directory : in String   := Def_Upload_Dir);
-- Start the Web server. It initialize the Max_Connection connections
-- lines. Name is just a string used to identify the server. This is used
-- for example in the administrative page. Admin_URI must be set to enable
-- the administrative status page. Callback is the procedure to call for
-- each resource requested. Port is the Web server port. If Security is
-- set to True the server will use an HTTPS/SSL connection. If Session is
-- set to True the server will be able to get a status for each client
-- connected. A session ID is used for that, on the client side it is a
-- cookie. Case_Sensitive_Parameters if set to False it means that the CGI
-- parameters name will be handled without case sensitivity. Upload
-- directory point to a directory where uploaded files will be stored.
```

`Start` takes many parameters:

Web_Server

this is the Web server to start.

Name

This is a string to identify the server. This name will be used for example in the administrative status page.

Callback

This is the procedure to call for each requested resources. In this procedure you must handle all the possible URI that you want to support. (see below).

Max_Connection

This is the maximum number of simultaneous connections. It means that Max_Connection client's browsers can gets answer at the same time. This parameter must be changed to match your needs. A medium Web server will certainly need something like 20 or 30 simultaneous connections.

Admin_URI

This is a special URI recognized internally by the server. If this URI is requested the server will return the administrative page. This page is built using a specific template page (default is 'aws_status.shtml') see [Chapter 11 \[Status page\]](#), [page 69](#).

The administrative page returns many information about the server. It is possible to configure the server via two configuration files See [Section 3.4 \[Configuration options\]](#), [page 16](#).

Port

This is the port to use for the Web server. You can use any free port on your computer. Note that on some OS specific range could be reserved or needs specials privileges (port 80 on Linux for example).

Security

If Security is set to True the server will use an HTTPS/SSL connection. This part uses the **OpenSSL** library.

Session

If Session is set to true the server will keep a session ID for each client. The client will be able to save and get variables associated with this session ID.

Case_Sensitive_Parameters

If set to True the CGI name parameters will be handled without using the case.

Note that there is other **Start** routines which support other features. For example there is a **Start** routine which use a dispatcher routine instead of the simple callback procedure. see [Section B.40 \[AWS.Server\]](#), [page 175](#). And there is also the version using a **Config.Object** which is the most generic one.

3. provides a callback procedure

The callback procedure has the following prototype:

```
function Service (Request : in AWS.Status.Data) return AWS.Response.Data;
```

This procedure receive the request status. It is possible to retrieve information about the request through the **AWS.Status** API (See [Section B.64 \[AWS.Status\]](#), [page 227](#)).

For example, to know what URI has been asked:

```

URI : constant String := AWS.Status.URI (Request);

if URI = "/whatever" then
    ...
end if;

```

Then this function should return an answer using one of the constructors in `AWS.Response` (See [Section B.39 \[AWS.Response\]](#), page 169.). For example, to return an HTML message:

```

AWS.Response.Build (Content_Type => "text/html",
                    Message_Body => "<p>just a demo");

```

It is also possible to return a file. For example, here is the way to return a PNG image:

```

AWS.Response.File (Content_Type => "image/png",
                  Filename      => "adains.png");

```

Note that the main procedure should exit only when the server is terminated. For this you can use the `AWS.Server.Wait` service.

A better solution is to use a template engine like `Templates.Parser` to build the HTML Web Server answer. `Templates.Parser` module is distributed with this version of AWS.

3.2.2 Callback procedure

The callback procedure is the user's code that will be called by the AWS component to get the right answer for the requested resource. In fact AWS just open the HTTP message, parsing the HTTP header and it builds an object of type `AWS.Status.Data`. At this point it calls the user's callback procedure, passing the object. The callback procedure must returns the right response for the requested resources. Now AWS will just build up the HTTP response message and send it back to user's browser.

But what is the resource ?

Indeed in a standard Web development a resource is either a static object - an HTML page, an XML or XSL document - or a CGI script. With AWS a resource is *just a string* to identify the resource, it does not represent the name of a static object or CGI script.

So this string is just an internal representation for the resource. The callback procedure must be implemented to handle each internal resource and return the right response.

Let's have a small example. For example we want to build a Web server that will answer "Hello World" if we ask for the internal resource `/hello`, and must answer "Hum..." otherwise.


```

with AWS.Response;
with AWS.Server;
with AWS.Status;

procedure Hello_World is

    WS : AWS.Server.HTTP;

    function HW_CB (Request : in AWS.Status.Data)
        return AWS.Response.Data
    is
        URI : constant String := Status.URI (Request);
    begin
        if URI = "/hello" then
            return AWS.Response.Build ("text/html", "<p>Hello world !");
        else
            return AWS.Response.Build ("text/html", "<p>Hum...");
        end if;
    end HW_CB;

begin
    AWS.Server.Start
        (WS, "Hello World", Callback => HW_CB'Unrestricted_Access);
    delay 30.0;
end Hello_World;

```

Now of course the resource internal name can represent a file on disk. It is not mandatory but it is possible. For example it is perfectly possible to build with AWS a simple page server.

As an example, let's build a simple page server. This server will return files in the current directory. Resources internal name represent an HTML page or a GIF or PNG image for example. This server will return a 404 message (Web Page Not Found) if the file does not exist. Here is the callback procedure that implements such simple page server:

```

function Get (Request : in AWS.Status.Data) return AWS.Response.Data is
    URI      : constant String := AWS.Status.URI (Request);
    Filename : constant String := URI (2 .. URI'Last);
begin
    if Utils.Is_Regular_File (Filename) then
        return AWS.Response.File
            (Content_Type => AWS.MIME.Content_Type (Filename),
             Filename     => Filename);

    else
        return AWS.Response.Acknowledge
            (Messages.S404,
             "<p>Page '" & URI & "' Not found.");
    end if;
end Get;

```

3.2.3 Form parameters

Form parameters are stored into a table of key/value pair. The key is the form input tag name and the value is the content of the input field as filled by the user.

```

Enter your name

<FORM METHOD=GET ACTION=/get-form>
<INPUT TYPE=TEXT NAME=name VALUE="<default>" size=15>
<INPUT TYPE=SUBMIT NAME=go VALUE="Ok">
</FORM>

```

Note that as explained above see [Section 3.2.2 \[Callback procedure\]](#), page 10, the resource described in `ACTION` is just an internal string representation for the resource.

In this example there is two form parameters:

name The value is the content of this text field as filled by the client.

go The value is "Ok".

There is many functions (in `AWS.Parameters`) to retrieve the tag name or value and the number of parameters. Here are some examples:

```
function Service (Request : in AWS.Status.Data)
  return AWS.Response.Data
is
  P : constant AWS.Parameters.List := AWS.Status.Parameters (Request);
  ...
```

`AWS.Parameters.Get (P, "name")`
Returns the value for parameter named **name**

`AWS.Parameters.Get_Name (P, 1)`
Returns the string "name".

`AWS.Parameters.Get (P, 1)`
Returns the value for parameter named **name**

`AWS.Parameters.Get (P, "go")`
Returns the string "Ok".

`AWS.Parameters.Get_Name (P, 2)`
Returns the string "go".

`AWS.Parameters.Get (P, 2)`
Returns the string "Ok".

`Request` is the `AWS` current connection status passed to the callback procedure. And `P` is the parameters list retrieved from the connection status data. For a discussion about the callback procedure See [Section 3.2.1 \[Building an AWS server\]](#), page 8.

3.2.4 Distribution of an AWS server

The directory containing the server program must contain the following files if you plan to use a status page See [Chapter 11 \[Status page\]](#), page 69.

`'aws_status.thtml'`
The template HTML file for the AWS status page.

`'aws_logo.png'`
The AWS logo displayed on the status page.

`'aws_up.png'`
The AWS hotplug table up arrow.

`'aws_down.png'`
The AWS hotplug table down arrow.

Note that these filenames are the current `AWS` default. But it is possible to change those defaults using the configuration files see [Section 3.4 \[Configuration options\]](#), page 16.

3.3 Building answers

We have already seen, in simple examples, how to build basic answers using `AWS.Response` API. In this section we present all ways to build answers from basic support to the more advanced support like the compressed memory stream response.

3.3.1 Redirection

A redirection is a way to redirect the client's browser to another URL. Client's won't notice that a redirection has occurs. As soon as the browser has received the response from the server it will retrieve the page as pointed by the redirection.

```
return Response.URL (Location => "/use-this-one");
```

3.3.2 New location for a page

User will receive a Web page saying that this page has moved and eventually pointing to the new location.

```
return Response.Moved
  (Location => "/use-this-one";
   Message  => "This page has moved, please update your reference");
```

3.3.3 Authentication required

For protected pages you need to ask user to enter a password. see [Section 3.6 \[Authentication\]](#), page 20.

3.3.4 Sending back an error message

`Acknowledge` can be used to send back error messages. There is many kind of status code, see `Message.Status_Code` definition. Together with the status code it is possible to pass textual error message in `Message_Body` parameter.

```
return Response.Acknowledge
  (Status_Code  => Messages.S503,
   Message_Body => "Can't connect to the database, please retry later.",
   Content_Type => MIME.Text_Plain);
```

3.3.5 Response from a string

This is the simplest way to build a response object. There is two constructors in `AWS.Response`, one based on a standard string and one for `Unbounded_String`.

```
return Response.Build (MIME.Text_HTML, "My answer");
```

The `Build` routine takes also a status code parameter to handle errors. By default this code is `Messages.S200` which is the standard HTTP status (no error encountered). The other parameter can be used to control caches. see [Section B.39 \[AWS.Response\]](#), page 169.

3.3.6 Response from a `Stream_Element_Array`

This is exactly as above but the `Build` routine takes a `Stream_Element_Array` instead of a string.

3.3.7 Response from a file

To build a `File` response there is a single constructor named `File`. This routine is very similar to the one above except that we specify a filename as the response.

```
return Response.File (MIME.Text_HTML, "index.html");
```

Again there parameters to control the status code and cache. No check on the filename is done at this point, so if `'index.html'` does not exist no exception is raised. The server is responsible to check for the file and to properly send back the 404 message if necessary.

Note that this routine takes an optional parameter named `Once` that is to be used for temporary files created on the server side for the client. With `Once` set to `True` the file will be deleted by the server after sending it (this includes the case where the download is suspended).

3.3.8 Response from a stream

Sometimes it is not possible (or convenient) to build the response in memory as a string object for example. Streams can be used to workaround this. The constructor for such response is again very similar to the ones above except that instead of the data we pass an handle to a `Resources.Streams.Stream_Type` object.

The first step is to build the stream object. This is done by deriving a new type from `Resources.Streams.Stream_Type` and implementing three abstract procedures.

Read

Must return the next chunk of data from the stream. Note that initialization if needed are to be done there during the first call to read.

End_Of_File

Must return `True` when there is no more data on the stream.

Close

Must close the stream and for example release all memory used by the implementation.

The second step is to build the response object:

```
type SQL_Stream is new Resources.Streams.Stream_Type;

Stream_Object : SQL_Stream;

procedure Read (...) is ...
function End_Of_File (...) return Boolean is ...
procedure Close (...) is
...

return Response.Stream (MIME.Text_HTML, Stream_Object);
```

Note that in some cases it is needed to create a file containing the data for the client (for example a tar.gz or a zip archive). But there is no way to properly remove this file from the file system as we really don't know when the upload is terminated when using the `AWS.Response.File` constructor. To solve this problem it is possible to use a stream as the procedure `Close` is called by the server when all data have been read. In this procedure it is trivial to do the necessary clean-up.

3.3.9 Response from a on-disk stream

An ready-to-use implementation of the stream API described above where the stream content is read from an on-disk file.

3.3.10 Response from a on-disk once stream

An ready-to-use implementation of the stream API described above where the stream content is read from an on-disk file. When the transfer is completed the file is removed from the file system.

3.3.11 Response from a memory stream

This is an implementation of the standard stream support described above. In this case the stream is in memory and built by adding data to it.

To create a memory stream just declare an object of type `AWS.Resources.Streams.Memory.Stream_Type`. When created, this memory stream is empty, using the `Streams.Memory.Append` routines it is possible to add chunk of data to it. It is of course possible to call `Append` as many times as needed. When done just return this object to the server.

```
Data : AWS.Resources.Streams.Memory.Stream_Type;

Append (Data, Translator.To_Stream_Element_Array ("First chunk"));
Append (Data, Translator.To_Stream_Element_Array ("Second chunk..."));

...

return Response.Stream (MIME.Text_HTML, Data);
```

Note that you do not have to take care of releasing the allocated memory, the default `Close` routine will do just that.

3.3.12 Response from a compressed memory stream

This is a slight variant of the standard memory stream described above. In this case the stream object must be declared as a `AWS.Resources.Streams.Memory.ZLib.Stream_Type`.

The ZLib stream object must be initialized to enable the compression and select the right parameters. This is done using the `AWS.Resources.Streams.Memory.ZLib.Deflate_Initialize` routine which takes many parameters to select the right options for the compression algorithm, all of them have good default values. When initialized the compressed stream object is used exactly as a standard stream.

```
Data : AWS.Resources.Streams.Memory.ZLib.Stream_Type;

Deflate_Initialize (Data);

Append (Data, Translator.To_Stream_Element_Array ("First chunk"));
Append (Data, Translator.To_Stream_Element_Array ("Second chunk..."));

...

return Response.Stream (MIME.Text_HTML, Data);
```

Note that there is the reverse implementation to decompress a stream. see [Section B.38 \[AWS.Resources.Streams.Memory.ZLib\]](#), page 167. It's usage is identical.

3.3.13 Split page

AWS has a specific high level service to split a large response into a set of pages. For more information see [Section 4.5 \[Split pages\], page 34](#).

3.4 Configuration options

To configure an AWS server it is possible to use a configuration object. This object can be set using the `AWS.Config.Set` API or initialized using a configuration file.

Configuration files are a way to configure the server without recompiling it. Each application can be configured using two configurations files:

`'aws.ini'`

This file is parsed first and corresponds to the configuration for all AWS server runs in the same directory.

`'<program_name>.ini'`

This file is parsed after `'aws.ini'`. It is possible with this initialization file to have specific settings for some servers.

Furthermore, it is possible to read a specific configuration file using the `AWS.Config.Ini.Read` routine. see [Section B.9 \[AWS.Config.Ini\], page 96](#).

Current supported options are:

Accept_Queue_Size (positive)

This is the size of the queue for the incoming requests. Higher this value will be and less "*connection refused*" will be reported to the client. The default value is 64.

Admin_URI (string)

This is the URI to call the administrative page. This can be used when calling `AWS.Server.Start`. The default is `'`.

Certificate (string)

Set the certificate file to be used with the secure servers. The default is `'cert.pem'`. A single certificate or a certificate chain is supported. The certificates must be in PEM format and the chain must be sorted starting with the subject's certificate, followed by intermediate CA certificates if applicable and ending at the highest level (root) CA certificate. If the file contains only a single certificate, it can be followed by a private key. In this case the Key parameter (see below) must empty.

Case_Sensitive_Parameters (boolean)

If set to `True` the HTTP parameters are case sensitive. The default value is `TRUE`.

Check_URL_Validity (boolean)

Server have to check URI for validity. For example it checks that an URL does not reference a resource above the Web root. The default is `TRUE`.

Cleaner_Wait_For_Client_Timeout (duration)

Number of seconds to timeout on waiting for a client request. This is a timeout for regular cleaning task. The default is 80.0 seconds.

Cleaner_Client_Header_Timeout (duration)

Number of seconds to timeout on waiting for client header. This is a timeout for regular cleaning task. The default is 20.0 seconds.

Cleaner_Client_Data_Timeout (duration)

Number of seconds to timeout on waiting for client message body. This is a timeout for regular cleaning task. The default is 28800.0 seconds.

Cleaner_Server_Response_Timeout (duration)

Number of seconds to timeout on waiting for client to accept answer. This is a timeout for regular cleaning task. The default is 28800.0 seconds.

Directory_Browser_Page (string)

Specify the filename for the directory browser template page. The default value is 'aws_directory.thtml'.

Down_Image (string).

The name of the down arrow image to use in the status page. The default is 'aws_down.png'.

Error_Log_Filename_Prefix (string)

This is to set the filename prefix for the log file. By default the log filename prefix is the program name (without extension) followed by "_error".

Error_Log_Split_Mode [None/Each_Run/Daily/Monthly]

It indicates how to split the error logs. Each_Run means that a new log file is used each time the process is started. Daily and Monthly will use a new log file each day or month. The default is NONE.

Exchange_Certificate (boolean)

If set to True it means that the client will be asked to send its certificate to the server. The default value is FALSE.

Force_Wait_For_Client_Timeout (duration)

Number of seconds to timeout on waiting for a client request. This is a timeout for urgent request when resources are missing. The default is 2.0 seconds.

Force_Client_Header_Timeout (duration)

Number of seconds to timeout on waiting for client header. This is a timeout for urgent request when resources are missing. The default is 3.0 seconds.

Force_Client_Data_Timeout (duration)

Number of seconds to timeout on waiting for client message body. This is a timeout for urgent request when resources are missing. The default is 10800.0 seconds.

Force_Server_Response_Timeout (duration)

Number of seconds to timeout on waiting for client to accept answer. This is a timeout for urgent request when resources are missing. The default is 10800.0 seconds.

Free_Slots_Keep_Alive_Limit (positive)

This is the minimum number of remaining free slots to enable keep-alive HTTP connections. For heavy-loaded HTTP servers, the Max_Connection parameter should be big enough, and Free_Slots_Keep_Alive_Limit should be about 1-10% of the Max_Connection parameter depending on the duration of the average server response. Longer is the average time to send back a response bigger Free_Slots_Keep_Alive_Limit should be. The default is 1.

Hotplug_Port (positive)

This is the hotplug communication port needed to register and un-register an hotplug module. The default value is 8888.

Key (string)

Set the RSA key file to be used with the secure servers. The default file is "".

Line_Stack_Size (positive)

The HTTP lines stack size. The stack size must be adjusted for each applications depending on the use of the stack done by the callback procedures. The default is 1376256.

Log_Extended_Fields (string list)

Comma separated list of the extended log field names. If this parameter is empty, the HTTP log would be in the apache compatible format, otherwise log file would be in Extended format. For more information see [Section 3.12 \[Server Log\]](#), page 25.

Log_File_Directory (string)

This is to set the directory where log file must be written. This parameter will be used automatically by `AWS.Log` if logging facility is enabled. By default log files are written in the current directory. The default is `'./'`.

Log_Filename_Prefix (string)

This is to set the filename prefix for the log file. By default the log filename prefix is the program name (without extension).

Log_Split_Mode [None/Each_Run/Daily/Monthly]

It indicates how to split the logs. `Each_Run` means that a new log file is used each time the process is started. `Daily` and `Monthly` will use a new log file each day or month. The default is `NONE`.

Logo (string).

The name of the logo image to use in the status page. The default is `'aws_logo.png'`.

Max_Connection (positive)

This is the maximum number of simultaneous connections for the server. This can be used when calling the `AWS.Server.Start`. The default is 5.

Note that the total number of threads used by a server is:

$$N = \text{<main server thread>} + \text{<max connections>} [+ \text{<session>}]$$

Note: [...] means optional value

Add 1 only if the session feature is activated. This is due to the session cleaner task.

Receive_Timeout (duration)

Number of seconds to timeout when receiving chunk of data. The default is 30.0 seconds.

Security_Mode (string)

Set the security mode to use for the secure connections. The default mode is **"SSLv23"**. see [Section B.27 \[AWS.Net.SSL\]](#), page 145.

Send_Timeout (duration)

Number of seconds to timeout when sending chunk of data. The default is 40.0 seconds.

Server_Host (string)

The name of the host machine. This can be used if a computer has more than one IP address, it is possible to have two servers at the same port on the same machine, both being binded on different IP addresses.

Server_Name (string)

The name of the server. This can be used when calling `AWS.Server.Start`. The default is **"AWS Module"**.

Server_Port (integer)

The port where server will wait for incoming connections requests. This can be used when calling `AWS.Server.Start`. The default is 8080.

Session_Lifetime (duration)

Number of seconds to keep session information. After this period a session is obsoleted and will be removed at next cleanup. The default is 600.0 seconds.

Session_Cleanup_Interval (duration)

Number of seconds between each run of the session cleanup task. This task will remove all session data that have been obsoleted. The default is 300.0 seconds.

Status_Page (string)

The name of the status page to used. The default is `'aws_status.thtml'`.

Transient_Cleanup_Interval (positive)

Specify the number of seconds between each run of the cleaner task to remove transient pages. The default value is 180.0 seconds.

Transient_Lifetime

Specify the number of seconds to keep a transient page. After this period the transient page is obsoleted and will be removed during next cleanup. The default value is 300.0 seconds.

Up_Image (string)

The name of the up arrow image to use in the status page. The default is `'aws_up.png'`.

Upload_Directory (string)

This is to set the directory where upload files must be stored. By default uploaded files are written in the current directory. The default is `'.'`.

WWW_Root (string)

This option sets the Web Server root directory. All Web resources are referenced from this root directory. The default value is `"/"`.

Each option value can be retrieved using the `AWS.Config` unit or set using `AWS.Config.Set`.

For example to build a server where the *port* and the maximum number of *connection* can be changed via a configuration file (either `'aws.ini'` or `'<program_name>.ini'`):

```
WS    : AWS.Server.HTTP;

Conf : constant AWS.Config.Object := AWS.Config.Get_Current;

Server.Start (WS, Service'Access, Conf);
```

3.5 Session handling

AWS provides a way to keep session data while users are browsing. There is no need to mess with the Cookies, AWS will do that for you. The idea is simple. A session ID will be transparently created and then you'll be able to insert and retrieve session data using a standard Ada API (See [Section B.61 \[AWS.Session\]](#), page 217.). Session data are key/value pair each of them being strings. How to deal with that ?

- First you declare and start an HTTP channel with session enabled:

```
WS : AWS.Server.HTTP;

Server.Start (WS,
  Port      => 1234,
  Callback  => Service'Access,
  Session   => True);
```

Here we have built an HTTP channel with a maximum of 3 simultaneous connections using the port 1234. A session ID will be created and sent inside a cookie to the client's browser at the first request. This session ID will be sent back to the server each time the client will ask for a resource to the server.

- Next, in the Service callback procedure that you have provided you must retrieve the Session ID. As we have seen, the callback procedure has the following prototype:

```
function Service (Request : in AWS.Status.Data) return AWS.Response.Data;
```

The Session ID is kept in the Request object and can be retrieved using:

```
Session_ID : constant AWS.Session.ID := AWS.Status.Session (Request);
```

- From there it is quite easy to get or set some session data using the provided API. For example:

```
declare
  C : Integer;
begin
  C := AWS.Session.Get (Session_ID, "counter");
  C := C + 1;
  AWS.Session.Set (Session_ID, "counter", C);
end;
```

This example first get the value (as an Integer) for session data whose key is "counter", increment this counter and then set it back to the new value.

It is also possible to save and restore all session data. It means that the server can be shutdown and launched some time after and all client data are restored as they were at shutdown time. Client will just see nothing. With this feature it is possible to shutdown a server to update its look or because a bug has been fixed for example. It is then possible to restart it keeping the complete Web server context.

3.6 Authentication

AWS supports **Basic** and **Digest** authentication. The authentication request can be sent at any time from the callback procedure. For this the `AWS.Response.Authenticate` message must be returned.

The authentication process is as follow:

1. Send authentication request

From the callback routine return an authentication request when needed.

```
function Service (Request : in Status.Data) return Response.Data is
  URI : constant String := Status.URI (Request);
  User : constant String := Status.Authorization_Name (Request);
begin
  -- URI starting with "/prot/" are protected
  if URI (URI'First .. URI'First + 5) = "/prot/"
    and then User = ""
  then
    return Response.Authenticate ("AWS", Response.Basic);
```

The first parameter is the **Realm**, it is just a string that will be displayed (on the authentication dialog box) by the browser to indicate for which resource the authentication is needed.

2. Check authentication

When an authentication has been done the callback's request data contain the user and password. Checks the values against an ACL for each protected resources.

```
function Protected_Service
  (Request : in AWS.Status.Data)
  return AWS.Response.Data
is
  User : constant String := Status.Authorization_Name (Request);
  Pwd  : constant String := Status.Authorization_Password (Request);
begin
  if User = "xyz" and then Pwd = "azerty" then
    return ...;
```

Note that the **Basic** authentication is not secure at all. The password is sent unencoded by the browser to the server. If security is an issue it is better to use the **Digest** authentication and/or an **SSL** server.

3.7 File upload

File upload is the way to send a file from the client to the server. To enable file upload on the client side the Web page must contain a **FORM** with an **INPUT** tag of type **FILE**. The **FORM** must also contain the **enctype** attribute set to *multipart/form-data*.

```
<FORM enctype="multipart/form-data" ACTION=/whatever METHOD=POST>
File to process: <INPUT NAME=filename TYPE=FILE>
<INPUT TYPE=SUBMIT NAME=go VALUE="Send File">
</FORM>
```

On the server side, AWS will retrieve the file and put it into the upload directory. AWS add a prefix to the file to ensure that the filename will be unique on the server side. The upload directory can be changed using the configuration options. See [Section 3.4 \[Configuration options\]](#), page 16.

AWS will also setup the form parameters as usual. In the above example there is two parameters (See [Section 3.2.3 \[Form parameters\]](#), page 11.)

filename This variable contains two values, one with the client side name and one with the server side name.

First value : Parameters.Get (P, "filename")

The value is the full pathname of the file on the server. (i.e. the upload directory catenated with the prefix and filename).

Second value : Parameters.Get (P, "filename", 2)

The value is the full pathname of the file on the client side.

go The value is "Send File"

3.8 Communication

This API is used to do communication between programs using the HTTP GET protocol. It is a very simple API not to be compared with **GLADE** or **SOAP**. This communication facility is to be used for simple request or when a light communication support is needed. For more complex

communications or to achieve inter-operability with other modules it is certainly a good idea to have a look at the AWS/SOAP support, see [Section B.68 \[SOAP\]](#), page 238.

In a communication there is a Client and a Server. Here is what is to be done on both sides to have programs talking together.

3.8.1 Communication - client side

On the client side it is quite simple. You just have to send a message using `AWS.Communication.Client.Send_Message`.

```
function Send_Message
(Server      : in String;
Port        : in Positive;
Name        : in String;
Parameters  : in Parameter_Set := Null_Parameter_Set)
return Response.Data;
```

The message is sent to the specified server using the given port. A message is composed of a name which is a string and a set of parameters. There is a parameter set constructor in `AWS.Communication`. This function return a response as for any callback procedure.

3.8.2 Communication - server side

On the server side things are a bit more complex but not that difficult. You must instantiate the `AWS.Communication.Server` generic package by providing a callback procedure. This callback procedure will must handle all kind of message that a client will send.

During instantiation you must also pass a context for the communication server. This context will be passed back to the callback procedure.

```
generic

  type T (<>) is limited private;
  type T_Access is access T;

  with function Callback
    (Server      : in String;
     Name        : in String;
     Context     : in T_Access;
     Parameters  : in Parameter_Set := Null_Parameter_Set)
    return Response.Data;

package AWS.Communication.Server is
  ...
```

A complete example can be found in the demos directory. Look for 'com_1.adb' and 'com_2.adb'.

Note that this communication API is used by the Hotplug module facility See [Section 3.9 \[Hotplug module\]](#), page 22.

3.9 Hotplug module

An **Hotplug module** is a module that can be dynamically binded to a running server. It is a Web server and the development process is very similar to what we have seen until now See

[Section 3.2.1 \[Building an AWS server\], page 8](#). The Hotplug module will register itself into a Web server by sending a message using the communication API. The Hotplug module send to the server a regular expression and an URL. The main server will redirect all URL matching the regular expression to the Hotplug module.

Note that the main server will redirect the URL to the first matching regular expression.

3.9.1 Hotplug module - server activation

The first step is to properly create the main server hotplug module registration file. This file must list all hotplug modules that can register into the main server. Each line have the following format:

```
hotplug_module_name:password:server:port
```

hotplug_module_name

The name of the hotplug module. You can choose any name you want. This name will be use during the registration process and to generate the password.

password

The MD5 password, see below.

server

The name of the server where the redirection will be made. This is for security reasons, main server will not permit to redirect requests to any other server.

port

The port to use for the redirection on **server**.

You must create a password for each hotplug modules. The generated password depends on the hotplug module name. A tool named `hotplug_password` is provided with AWS to generate such password. Usage is simple:

```
$ hotplug_password <hotplug_module_name> <password>
```

Then, after starting the main server you must activate the Hotplug feature:

```
AWS.Server.Hotplug.Activate (WS'Unchecked_Access, 2222, "hotplug_conf.ini");
```

'hotplug_conf.ini' is the hotplug module registration file described above.

3.9.2 Hotplug module - creation

Here is how to create an Hotplug module:

1. First you create a standard Web server See [Section 3.2.1 \[Building an AWS server\], page 8](#).

```
WS : AWS.Server.HTTP
    (3, 1235, False, Hotplug_CB.Hotplug'Access, False);
```

Here we have a server listening to the port 1235. This server can be used alone if needed as any Server developed with AWS.

2. Then you register the Hotplug module to the main server See [Section B.4 \[AWS.Client.Hotplug\], page 86](#).

```
Response := AWS.Client.Hotplug.Register
(Name      => "Hotplug_Module_Demo",
 Password => "my_password",
 Server   => "http://dieppe:2222",
 Regexp   => ".*AWS.*",
 URL      => "http://omsk:1235/");
```

The hotplug module `Hotplug_Module_Demo` must have been declared on the main server, the password and redirection must have been properly recorded too for security reasons see [Section 3.9.1 \[Hotplug module - server activation\]](#), page 23. This command register `Hotplug_Module_Demo` into the server running on the machine `dieppe` and ask it to redirect all URL containing `AWS` to the server running on machine `omsk` on port 1235.

3. When the Hotplug module is stopped, you must unregister it

```
Response := AWS.Client.Hotplug.Unregister
(Name      => "Hotplug_Module_Demo",
 Password => "my_password",
 Server   => "http://dieppe:2222",
 Regexp   => ".*AWS.*");
```

Here we ask to unregister `Hotplug_Module_Demo` from server `dieppe`. As for the registration process a proper password must be specified see [Section 3.9.1 \[Hotplug module - server activation\]](#), page 23.

A complete example can be found in the demos directory. Look for 'main.adb' and 'hotplug.adb'.

3.10 Server Push

Server Push is a feature that let the Web Server send continuously data to client's Web Browser or client applications. The client does not have to reload at periodic time (which is what is called client pull) to have the data updated, each time the server send a piece of data it gets displayed on the client.

To build a push server you need to build an instance of the `AWS.Server.Push` package. This package takes a set of formal parameters. Here are the step-by-step instructions to build a Push Server:

1. The data to be sent

First you must create a type that will contains the data to be sent to client's browser except if it is a standard Ada type. See `Client_Output_Type` formal parameter.

2. The data that will be streamed

This is the representation of the data that will be sent to client's browser. This will be either a `String` for Web pages or `Stream_Element_Array` for binary data like pictures. See `Stream_Output_Type` formal parameter.

3. The context

It is often nice to be able to configure each client with different parameters if needed. This can be achieved with the Context data type that will be passed as parameter of the conversion function described below. See `Client_Environment` formal parameter.

4. Provides a function to convert from the data type to be sent to the data that will be streamed.

This is a function that will transform the data described on point 1 above to the form described on point 2 above. See `To_Stream_Output` formal parameter.

5. Build the Push Server

To do so you just need to instantiate `AWS.Server.Push` with the above declarations.

6. Registering new clients

In the standard AWS procedure callback it is possible to register a client if requested. This is done by calling `AWS.Server.Push.Register`. It is possible to unregister a client using `AWS.Server.Push.Unregister`. Each client must be identified with a unique client ID. After registering a new client from the callback procedure you must return the `AWS.Response.Socket_Taken` message. This is very important, it tells the server to not close this socket.

7. Sending the data

At this point it is possible to send data to clients. To do so two routines are available.

`AWS.Server.Push.Send_To`

To send a piece of data to a specific client identified by its client ID.

`AWS.Server.Push.Send`

To send a piece of data to all clients registered on this server.

Very large Internet applications should use this feature carefully. A push server keeps a socket reserved for each registered clients and the number of available sockets per process is limited by the OS.

3.11 Working with Server sockets

With AWS it is possible to take out a socket from the server and give it back later. This feature must be used carefully but it gives a lot of flexibility. As the socket is taken away, the connection line (or slot) is released, AWS can then use it to handle other requests.

This can be used to better support heavy loaded servers when some requests need a long time to complete. Long time here means longer than most of the other requests which should be mostly interactivities for a Web server. Of course in such a case a keep-alive connection is kept open.

The usage in such a case is to take out the socket and put it in a waiting line. This releases the connection for the server. When the data to prepare the answer is ready you give back the socket to the server.

- Take a socket from the server

This first step is done from the callback function. A user instead of replying immediately decides to take away the socket from the server. The first step is to record the connection socket by calling `AWS.Status.Socket`. The second step is to tell the server to not release this socket by returning `AWS.Response.Socket_Taken` from the callback function. At this point the server will continue to serve other clients.

Note that this feature is used by the server push implementation see [Section 3.10 \[Server Push\]](#), page 24.

- Give back the socket to the server

Calling `AWS.Server.Give_Back_Socket` will register the socket for reuse. This socket will be placed into a pool, next time the server will check for incoming requests it will be picked up.

3.12 Server Log

It is possible to have the server activity logged into the file '`<progname>-Y-M-D.log`'. To activate the logging you must call the `AWS.Server.Log.Start`, and it is possible to stop logging by calling `AWS.Server.Log.Stop`. Note that `AWS.Server.Log.Start` have a parameter named `Auto_Flush` to control output buffering. This parameter is False by default. If set to True, the

log file will be automatically flushed after each data. If the server logging is not buffered, i.e. `Auto_Flush` is `False`, the log can still be flushed by calling the `AWS.Server.Log.Flush` routine. See [Section B.20 \[AWS.Log\], page 123](#), for more information especially about the way rotating logs can be setup. Using this feature it is possible to have automatic split of the log file each day, each month or at every run. See `AWS.Log` spec. This is very useful to avoid having very big log files.

The log format depend on `Log_Extended_Fields` configuration parameter. If this parameter is empty, the HTTP log would have fixed apache compatible format:

```
<client IP> - <auth name> - [<date and time>] "<request>" <status code> <size>
```

For example:

```
100.99.12.1 - - [22/Nov/2000:11:44:14] "GET /whatever HTTP/1.1" 200 1789
```

If the extended fields list is not empty, the log file format would have user defined fields set:

```
#Version: 1.0
#Date: 2006-01-09 00:00:01
#Fields: date time c-ip cs-method cs-uri cs-version sc-status sc-bytes
2006-01-09 00:34:23 100.99.12.1 GET /foo/bar.html HTTP/1.1 200 30
```

Fields in the comma separated `Log_Extended_Fields` list could be:

<i>date</i>	Date at which transaction completed
<i>time</i>	Time at which transaction completed
<i>time-taken</i>	Time taken for transaction to complete in seconds
<i>c-ip</i>	Client side connected IP address
<i>c-port</i>	Client side connected port
<i>s-ip</i>	Server side connected IP address
<i>s-port</i>	Server side connected port
<i>cs-method</i>	HTTP request method
<i>cs-username</i>	Client authentication username
<i>cs-version</i>	Client supported HTTP version
<i>cs-uri</i>	Request URI
<i>cs-uri-stem</i>	Stem portion alone of URI (omitting query)
<i>cs-uri-query</i>	Query portion alone of URI
<i>sc-status</i>	Response status code
<i>sc-bytes</i>	Length of response message body
<i>cs(<header>)</i>	Any header field name sent from client to server
<i>sc(<header>)</i>	Any header field name sent from server to client

x-<appfield>

Any application defined field name

AWS also support error log files. If activated every internal error detected by AWS will gets logged into this special file. Log file for errors would be in simple apache compatible format. See `AWS.Server.Log.Start_Error` and `AWS.Server.Log.Stop_Error`.

For the full set of routines supporting the log facility see [Section B.42 \[AWS.Server.Log\], page 181](#).

3.13 Secure server

It is not much difficult to use a secure server (HTTPS) than a standard one. Here we describe only what is specific to an HTTPS server.

Before going further you must check that AWS has been configured with SSL support. see [Section 2.3 \[Building\], page 4](#). You must also have installed the OpenSSL library on your system. If this is done, you can continue reading this section.

3.13.1 Initialization

A server is configured as using the HTTPS protocol at the time it is started. The only thing to do is to set the Start's Security parameter to True. This will start a server and activate the SSL layer by default. A secure server must use a valid certificate, the default one is 'cert.pem'. This certificate has been created by the openssl tool and is valid until year 2008. Yet, this certificate has not been signed. To build a secure server user's can rely on, you must have a valid certificate signed by one of the **Certificate Authorities**.

The certificate to be used must be specified before starting the secure server with `AWS.Server.Set_Security`.

```
AWS.Server.Set_Security (WS, Certificate_Filename => "/xyz/aws.pem");
```

3.13.2 Creating a test certificate

The goal here is not to replace the OpenSSL documentation but just to present one way to create a certificate for an HTTPS test server.

The RSA key

```
$ openssl genrsa -rand <filename> -out ca-key.pem
```

Filename must be point to any file, this is used to initialized the random seed.

The Certificate

```
$ openssl req -new -x509 -days 730 -key ca-key.pem -out ca-cert.pem
```

Create a single self contained file

```
$ cat ca-key.pem ca-cert.pem > aws.pem
```

At this point you can use 'aws.pem' with your server.

3.13.3 Protocol

There are different security options, either SSLv2, SSLv3 or TLSv1. SSLv2 and SSLv3 are supported by most if not all Web browsers. These are the default protocol used by AWS.

TLSv1 is not supported at this point.

3.14 Unexpected exception handler

When AWS detects an internal problem, it calls a specific handler. This handler can be used to log the error, send an alert message or build the answer to be sent back to the client's browser. Here is the spec for this handler:

```
type Unexpected_Exception_Handler is access
  procedure (E      : in      Ada.Exceptions.Exception_Occurrence;
             Log     : in out  AWS.Log.Object;
             Error   : in      Data;
             Answer  : in out  Response.Data);
```

The handler can be called in two modes:

Non fatal error (**Error.Fatal is False**)

In this case AWS will continue working without problem. A bug has been detected but it was not fatal to the thread (slot in AWS's terminology) handling. In this case it is possible to send back an application level message to the client's browser. For that you just have to fill the unexpected handler's *Answer* parameter with the right response message. The *Error* parameter receive information about the problem, see [Section B.15 \[AWS.Exceptions\]](#), page 110.

Fatal error (**Error.Fatal is True**)

In this case AWS will continue working but a thread (slot number *Error.Slot* in AWS's terminology) will be killed. It means that AWS will have lost one the simultaneous connection handler. The server will continue working unless it was the last slot handler available. Note that a Fatal error means an AWS internal bug and it should be reported if possible. In this mode there is no way to send back an answer to the client's browser and *Error* value must be ignored.

The default handler for unexpected exceptions send a message to standard error for fatal errors. For non fatal errors it log a message (if the error log is activated for the server) and send back a message back to the client. The message is either a built-in one or, if present in the server's directory, the content of the '500.tmp1t' file. This templates can used the following tags:

AUTH_MODE

The authorization mode (Either NONE, BASIC or DIGEST).

EXCEPTION

Exception information with traceback if activated.

HTTP_VERSION

Either HTTP/1.0 or HTTP/1.1

METHOD The request method (Either GET, HEAD, POST or PUT)

PAYLOAD

The full XML payload for SOAP request.

PEERNAME

The IP address of the client

SOAP_ACTION

Either True or False. Set to True for a SOAP request.

URI

The complete URI

For more information see [Section B.40 \[AWS.Server\]](#), page 175 and see [Section B.15 \[AWS.Exceptions\]](#), page 110.

3.15 Socket log

To ease AWS applications debugging it is possible to log all data sent/received to/from the sockets. For this you need to call the `AWS.Net.Log.Start` routine by passing a write procedure callback. You have to create such procedure or use one read-to-use provided in `AWS.Net.Log.Callbacks` package.

For more information see [Section B.25 \[AWS.Net.Log\]](#), page 141 and see [Section B.26 \[AWS.Net.Log.Callbacks\]](#), page 143.

3.16 Client side

AWS is not only a server it also implement the HTTP and HTTPS protocol from the client side. For example with AWS it is possible to get a Web page content using the `AWS.Client` API, See [Section B.3 \[AWS.Client\]](#), page 80.

It also support client **Keep-Alive** connections. It is then possible to request many URI from the same server using the same connection (i.e. the same sockets).

AWS client API also support proxy, proxy authentication and Web server authentication. Only basic (and not digest) authentication is supported at this time.

Let's say that you want to retrieve the `contrib.html` Web page from Pascal Obry's homepage which is <http://perso.wanadoo.fr/pascal.obry>. The code to do so is:

```
Data := Client.Get
  (URL => "http://perso.wanadoo.fr/pascal.obry/contrib.html");
```

From there you can ask for the result's content type:

```
if Response.Content_Type (Data) = "text/html" then
  ...
end if;
```

Or using the MIME types defined in `AWS.MIME` unit:

```
if Response.Content_Type (Data) = MIME.Text_HTML then
  ...
end if;
```

And display the content if it is some kind of text data:

```
Text_IO.Put_Line (Response.Message_Body (Data));
```

If the content is some kind of binary data (executable, PNG image, Zip archive...), then it is possible to write the result to a file for example. Look at the **agent** program in the **demo**s directory.

If the Web page is protected and you must pass the request through an authenticating proxy, the call will becomes:

```
Data := Client.Get
  (URL      => "http://www.mydomain.net/protected/index.html"
  User      => "me",
  Pwd       => "mypwd",
  Proxy     => "192.168.67.1",
  Proxy_User => "puser",
  Proxy_Pwd => "ppwd");
```

The client upload protocol is implemented. Using `AWS.Client.Upload` it is possible to send a file to a server which support the file upload protocol.

4 High level services

Here you will find a description of high level services. These services are ready to use with AWS and can be used together with user's callbacks.

Refer to the Ada spec for a complete API and usage description.

4.1 Directory browser

This service will help building a Web directory browser. It has a lot of options to sort directory entries and is based on the templates interface see [Section B.65 \[AWS.Templates\], page 232](#). This means that you can use the default directory template or provide your own.

see [Section B.46 \[AWS.Services.Directory\], page 190](#) for complete spec and services descriptions.

4.2 Dispatchers

In many AWS applications it is needed to check the URI to give the right answer. This means that part of the application is a big **if/elsif** procedure. Also, in standard callback it is not possible to have user data. Both of these restrictions are addressed with the Dispatchers facilities.

Working with a dispatcher is quite easy:

1. Create a new dispatcher by inheriting from the service you want to build.
2. Register a set of action based on rules (strings, regular expressions depending on the service)

4.2.1 Callback dispatcher

This is a wrapper around the standard callback procedure. It is needed to mix dispatcher based callback and access to procedure callback. Note that it is not in the `AWS.Services.Dispatchers` hierarchy but in `AWS.Dispatchers.Callback` because this is a basic service needed for the server itself. It is referenced here for documentation purpose but an AWS server can be built with using it.

see [Section B.14 \[AWS.Dispatchers.Callback\], page 109](#) for complete spec description.

4.2.2 Method dispatcher

This is a dispatcher based on the request method. A different callback procedure can be registered for the supported request methods: GET, POST, PUT, HEAD.

see [Section B.49 \[AWS.Services.Dispatchers.Method\], page 196](#) for complete spec description.

4.2.3 URI dispatcher

This is a dispatcher based on the request resource. A different callback procedure can be registered for specific resources. The resource is described either by its full name (string) or a regular expression.

see [Section B.50 \[AWS.Services.Dispatchers.URI\], page 198](#) for complete spec description.

4.2.4 Virtual host dispatcher

This is a dispatcher based on the host name. A different callback procedure can be registered for specific host. This is also known as virtual hosting.

The same computer can be registered into the DNS with different names. So all names point to the same machine. But in fact you want each name to be seen as a different Web server. This is called virtual hosting. This service will just do that, call different **callback** procedures or redirect to some **machine/port** based on the host name in the client's request.

see [Section B.51 \[AWS.Services.Dispatchers.Virtual_Host\], page 200](#) for complete spec description.

4.2.5 Transient pages dispatcher

This is a dispatcher that calls a user's callback and if the resource requested is not found (i.e. the user's callback returns status code 404) it checks if this resource is known as a transient page. see [Section 4.4 \[Transient Pages\], page 33](#).

4.2.6 Timer dispatcher

A timer dispatcher can be used to call different callback routines depending on the current date and time. Such dispatcher is composed of a set of **Period** activated. When the current date and time is inside a **Period** the corresponding callback is called. A **Period** can eventually be repeated. Here are the different kind of **Period** supported by AWS:

Once

A unique period in time. The boundaries are fully described using a year, month, day, hour, minute and second.

Yearly

A period that repeats each year. The boundaries are described using a month, day, hour, minute and second.

Monthly

A period that repeats each month. The boundaries are described using a day, hour, minute and second.

Weekly

A period that repeats each week. The boundaries are described using a day name, hour, minute and second.

Daily

A period that repeats each day. The boundaries are described using an hour, minute and second.

Hourly

A period that repeats each hour. The boundaries are described using a minute and second.

Minutely

A period that repeats each minute. The boundaries are described using a second.

4.2.7 Linker dispatcher

A dispatcher that can be used to chain two dispatchers. The response of the first dispatcher is returned except if it is a 404 (Not Found) error. In this case, the response of the second dispatcher is returned.

4.2.8 SOAP dispatcher

AWS provides also a SOAP specific dispatcher. This is a way to automatically route HTTP requests or SOAP requests to different callback routines.

see [Section 5.2.2 \[SOAP helpers\], page 45](#) for more information.

see [Section B.71 \[SOAP.Dispatchers.Callback\], page 242](#) for complete spec description.

4.3 Static Page server

This service is a ready to use static page server callback. Using it is possible to build a simple static page server, as simple as:

```
with AWS.Server;
with AWS.Services.Page_Server;

procedure WPS is
  WS : AWS.Server.HTTP;
begin
  AWS.Server.Start
    (WS, "Simple Page Server demo",
     Port      => 8080,
     Callback  => AWS.Services.Page_Server.Callback'Access,
     Max_Connection => 5);

  AWS.Server.Wait (AWS.Server.Q_Key_Pressed);

  AWS.Server.Shutdown (WS);
end WPS;
```

Build this program and launch it, it will server HTML pages and images in the current directory.

It is possible to activate the directory browsing facility of this simple page server. This is not activated by default. This feature is based on the directory browsing service see [Section 4.1 \[Directory browser\]](#), page 31.

Note that this service uses two template files:

aws_directory.thtml

The template page used for directory browsing. See [Section B.46 \[AWS.Services.Directory\]](#), page 190 for a full description of this template usage.

404.thtml

The Web page returned if the requested page is not found. This is a template with a single tag variable named PAGE. It will be replaced by the resource which was not found.

Note that on Microsoft IE this page will be displayed only if the total page size is bigger than 512 bytes or it includes at least one image.

see [Section B.53 \[AWS.Services.Page_Server\]](#), page 204 for a complete spec description.

4.4 Transient Pages

A transient page is a resource that has a certain life time on the server. After this time the resource will be released and will not be accessible anymore.

Sometimes you want to reference, in a Web page, a resource that is built in memory by the server. This resource can be requested by the client (by clicking on the corresponding link) or not, in both cases the page must be released after a certain amount of time to free the associated memory.

This is exactly what the transient pages high level service do automatically. Each transient page must be registered into the service, a specific routine named **Get_URI** can be used to create a unique URI on this server. see [Section B.60 \[AWS.Services.Transient_Pages\]](#), page 216.

A transient pages dispatcher can be used to build a transient pages aware server. see [Section 4.2.5 \[Transient pages dispatcher\]](#), page 32.

4.5 Split pages

It is not very convenient to send back a Web page with a large table. In such a case it is better to split the table in chunks (20 lines or so) and to send only the first page. This page references the next pages and can also contain an index of the pages.

The AWS's split page feature can automatically do that for you. Given template `Translate_Table` or `Translate_Set` and the max line per page it returns the first page and creates a set of transient pages for all other pages. A set of template tags are used to reference the previous and next page and also to build the page index.

There are different ways to split a set of pages and ready-to-use splitters are available:

Alpha Split in (at most) 28 pages, one for empty fields, one for all fields that start with a digit, and one for each different initial letter. see [Section B.55 \[AWS.Services.Split_Pages.Alpha\]](#), page 207.

Alpha.Bounded

Same as the alpha splitter, but pages larger than a `Max_Per_Page` value are further splitted. A secondary index is generated that gives the various pages for a given letter. see [Section B.56 \[AWS.Services.Split_Pages.Alpha.Bounded\]](#), page 209.

Uniform Split in pages of length `Max_Per_Page` (except the last one). This corresponds to the default service in `Split_Pages` package. see [Section B.57 \[AWS.Services.Split_Pages.Uniform\]](#), page 211.

Uniform.Alpha

Same as the uniform splitter, but builds in addition an alphabetical secondary index from a key field. see [Section B.58 \[AWS.Services.Split_Pages.Uniform.Alpha\]](#), page 213.

Uniform.Overlapping

Same as the uniform splitter, but pages (except the first one) repeat `Overlap` lines from the previous page in addition to the `Max_Per_Page` lines. see [Section B.59 \[AWS.Services.Split_Pages.Uniform.Overlapping\]](#), page 215.

Using the splitter abstract interface it is possible to build a customized splitter algorithm. see [Section B.54 \[AWS.Services.Split_Pages\]](#), page 205.

4.6 Download Manager

A server that needs to handle a lot of large downloads can run out of connection to answer the standard Web pages. A solution is to increase the number of simultaneous connections, but this is not really efficient as a task is created for each connection and does not ensure that all the connections will be used for the downloads anyway.

The download manager can be used for that, and provides the following feature:

- use a single task for all downloads
- can be configured to limit the number of simultaneous connections
- downloads past this limit are queued
- send messages to the client with the position in the waiting line
- send messages to the client when the download is about to start

The server must be configured to use dispatchers (standard callbacks are not supported, note that it is possible to create a dispatcher for standard callbacks. see [Section B.14 \[AWS.Dispatchers.Callback\]](#), page 109).

To start the download manager you need to pass the main server dispatcher object. The start routine will return a new dispatcher, linked with the download server specific dispatcher,

that must be used to start the standard Web server. See comment in see [Section B.52 \[AWS.Services.Download\]](#), page 202.

To queue a download request in the download manager you just need to create a stream object (can be any kind of stream, see `AWS.Resources.Streams.*`) for the resource to download.

The download manager needs two templates files:

`aws_download_manager_waiting.thtml`

This template is used for sending a message to the client when the request is on the waiting line. The tags defined in this template file are:

NAME the name of the resource to download (the filename), this is the default filename used for the client side save dialog.

RES_URI the URI used to access the resource.

POSITION the position in the waiting line (not counting the current served clients).

`aws_download_manager_start.thtml`

This template is used for sending a message to the client when the download is about to start (the request is out of the waiting line). The tags defined in this template file are:

NAME as above

RES_URI as above

It is important to note that those templates must be reloaded periodically. The best way to do that in the context of an HTML document is to use a meta-tag. For example to refresh the page every two seconds:

```
<meta http-equiv="refresh" content="2">
```

The templates could look like:

`aws_download_manager_waiting.thtml`

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="refresh" content="2">
  <title>Download Manager - waiting</title>
</head>
<body>
  <p>Waiting for downloading @_NAME_@
  <p>Position in the waiting line @_POSITION_@
</body>
</html>
```

aws_download_manager_start.shtml

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="refresh" content="2">
  <title>Download Manager - waiting</title>
</head>
<body>
  <p>Waiting for downloading @_NAME_@
  <p>The download will start in a moment
</body>
</html>
```

4.7 Web Elements

AWS provides some components to help creating nice looking Web interfaces. It is possible to browse those Web Elements using the `web_elements` demo. Just launch this Web application from the demos directory and turn your Web browser to <http://localhost:2400>.

Currently AWS provides:

- Notebooks (based on CSS)
- CSS Menu
- Rounded boxes
- Ajax

All of them are based on templates to be easily reused in other applications. The three first are best described by the Web Elements demos as they are 100% design. The Ajax one is a bit more complex, we will present its use in the following section.

4.7.1 Installation

To ease integration we have used the following design:

- Sub-directories found in the AWS's `web_elements` directory are self contained. The content must be copied into the project. Note that the icons and javascripts directories contain the icons and javascripts code shared by all web elements and must also be copied, see below.
- Each graphic elements (icons) is referenced into the templates with the alias `/we_icons/<icon_name>`. So users must provide the right alias (`/we_icons/`) in the Web server.
- Each JavaScripts code is referenced into the templates with the alias `/we_js/<script>`. So users must provide the right alias (`/we_js/`) in the Web server.

4.7.2 Ajax

First of all, Ajax stand for *Asynchronous JavaScript language and XML*, and is not well defined at the moment. Ajax is on one side able to send HTTP requests to the Web server and on the other side able to manipulate directly the Web browser's DOM tree. On the DOM it can add, remove or replace XML nodes. So, it is possible to change the content of a Web page without reloading it from the server.

Most importantly, Ajax changes the way Web applications are thought from **page** based to **event** based.

As implemented into AWS, Ajax support comes as a set of JavaScript templates. Using those templates there is no need to know JavaScript (except for the JavaScript event names) and

it makes **Ajax** programming lot easier. Two actions are provided, one for replacing another for clearing part of the web page content.

4.7.2.1 Steps to do Ajax

What are the steps to do **Ajax** ?

Remember, do not think about the Web page but about a specific widget (HTML fragments) with the associated event and action.

1. Include the AWS/Ajax support file

This is the **AWS/Ajax** runtime, it contains **JavaScript** code needed for the **AWS/Ajax** support.

2. Create the Web widgets/forms

There is nothing special here, use your favorite Web designer tool.

3. Create Web area

Using some **HTML** `<div>` tags we create areas where we will place **HTML** fragments later. For example when clicking on a button (described above) in our Web interface we want to display a new form in this area.

4. Name the widgets/forms/area using `id="name"` attribute

Give a different name to the widgets using `id="name"`. This name will be later used to identify the widgets on which the event and corresponding action must be placed. We do not want to clutter the Web design with **JavaScript** code like `onclick="dothis()"` or `onchange="dothat()"`.

5. Add the proper event/action to the widgets using the **AWS/Ajax** templates

This is the interesting part. At this point we link events/actions to the widgets and specify in which area the results sent by the server will be placed.

This is not the only way to do **Ajax**, we just presented here a simple approach that works well with the **AWS/Ajax** templates.

4.7.2.2 Basic Ajax support

This section describes the **AWS/Ajax** support where the answer from the server is an **HTML** fragment. This basic support is designed to be used for migration of a Web server to **Ajax**. For new applications, it is worth considering using the **XML** based **Ajax** support, see [Section 4.7.2.3 \[XML based Ajax\]](#), page 38.

Let's have a very simple example:

- The **AWS/Ajax** runtime support

```
@@INCLUDE@@ aws.tjs
```

- The widget: a button

```
<input id="clickme" type="button" value="Clik Me">
```

- The result area: a div

```
<div id="placeholder">... result here ...</div>
```

- The **AWS/Ajax**

```
@@INCLUDE@@ aws_action_replace.tjs onclick clickme placeholder
```

Basically it places an **onclick** attribute (the event) in the HTML **<input>** identified as **clickme** (the action) above. Here is what happen when the button is clicked:

- send the `"/clickme"` HTTP request to the server
- asynchronously wait for the answer, when received place the message body into the `<div>` **placeholder**.

On the server side the code would look like this:

```
function Callback (Request : in Status.Data) return Response.Data is
  URI : constant String := Status.URI (Request);
begin
  if URI = "/clickme" then
    return Response.Build (MIME.Text_HTML, "you click me!");
  ...
end;
```

So when the button is clicked the string **"you click me!"** will replace the **"... result here ..."** string of the place holder div above.

This is a simple and very limited example as there is no parameter passed to the HTTP request. In real Web applications it is necessary to send a context with the request. This can be either the value of other widgets or all values of widgets' form.

References to widgets or forms can be passed to the `'aws_action_replace.tjs'` template starting with the 5th parameter.

```
<input id="field" type="text" value="default value">
...
@@INCLUDE@@ aws_action_replace.tjs (onclick clickme placeholder 5=>field)
```

OR

```
<form id="small_form" name="small_form">
...
</form>
@@INCLUDE@@ aws_action_replace.tjs (onclick clickme placeholder 5=>small_form)
```

Note that the **onclick** event is only one of the possible JavaScript event on a **button**. It is possible to used any supported event, for example on an HTML **<select>** widget it is common to map the action to the **onchange** event.

AWS also provides support for clearing an area or a widget content (like an input).

```
@@INCLUDE@@ aws_action_clear.tjs (onclick, clear, field)
```

This simple action adds the **onclick** event to the **clear** button to erase the content of the **field** widget.

4.7.2.3 XML based Ajax

In many cases you'll like to update and/or clear multiple areas in your Web interface. With the templates above only a single action is possible. AWS provides support for XML based answers. In this XML documents it is possible to:

- replace an area with a new content

```
<replace id="item_id">new text</replace>
```

- clear an area

```
<clear id="item_id"/>
```

- add an item into a select widget

```
<select action="add" id="item_id"
option_value="value" option_content="content"/>
```

- remove an item from a select widget

```
<select action="delete" id="item_id" option_value="value"/>
```

- select a specific item in a select widget

```
<select action="select" id="item_id" option_value="value"/>
```

- clear a select widget (remove all items)

```
<select action="clear" id="item_id"/>
```

- select a radio button

```
<radio action="select" id="item_id"/>
```

- check a checkbox

```
<check action="select" id="item_id"/>
```

- clear a checkbox

```
<check action="clear" id="item_id"/>
```

- call another URL

```
<get url="http://thishost/action">
  <parameters value="name=Ajax"/>
  <field id="input1"/>
</get>
```

This will send the following request:

```
http://thishost/action?name=Ajax&input1=<val_input1>
```

Where **val_input1** is the current value of the **input1** input widget. The result must be an XML/Ajax document that will be parsed.

- make a list sortable

```
<make_sortable>
  <list id="firstlist"/>
  <list id="secondlist"/>
</make_sortable>
```

Here **firstlist** and **secondlist** are **id** of UL elements. It is possible to specify as many list id as needed. A drag and drop is then possible for all elements in those lists. It is then possible to reference such list by passing the list id as a field to the template. Items on those list will be serialized and passed to the **AWS** callback. Note that for the serialization to work properly, each LI elements must be given the id of the list and then the value we want to pass.

```
<ul id="firstlist">
  <li id="firstlist_red">Red</li>
  <li id="firstlist_green">Green</li>
  <li id="firstlist_blue">Blue</li>
</ul>
```

The serialization will send each value on this list using a multi-valued parameter named **firstlist[]**.

```
http://server?firstlist[]=red&firstlist[]=green&firstlist[]=blue
```

- make a list not sortable

```
<destroy_sortable>
  <list id="firstlist"/>
  <list id="secondlist"/>
</destroy_sortable>
```

Remove the sortable properly from the specified lists.

- redirect to another URL

```
<location url="http://thishost/go_there"/>
```

Redirect the browser to the specified URL.

- Add a CSS style to a given node

```
<apply_style id="node_id">
  <attribute id="display" value="none"/>
</apply_style>
```

Add the CSS style **display:none** to the **node_id** element. It is possible to specify multiple attributes if needed.

Here is an example of such XML document:

```
<response>
  <replace id="xml_status_bar">Fill Widgets...</replace>
  <replace id="text1">Response from XML</replace>
  <replace id="text2">Another response for text2</replace>
  <replace id="input1">tag is input1</replace>
  <replace id="input2">tag is input2</replace>
  <select action="add" id="xmlsel" option_value="one" option_content="1"/>
  <select action="add" id="xmlsel" option_value="two" option_content="2"/>
  <select action="add" id="xmlsel" option_value="three" option_content="3"/>
  <select action="select" id="xmlsel" option_value="two"/>
  <radio action="select" id="radio1"/>
  <check action="select" id="check1"/>
  <check action="select" id="check3"/>
  <check action="clear" id="check2"/>
</response>
```

The template to use this feature is 'aws_action_xml.tjs'. The usage is similar to what is described in the previous section (see [Section 4.7.2.2 \[Basic Ajax support\], page 37](#)) expect that in this case we do not have to pass the placeholder.

Let's revisit the first example above to use the XML Ajax support.

- The AWS/Ajax runtime support

```
@@INCLUDE@@ aws.tjs
```

- The widget: a button

```
<input id="clickme" type="button" value="Clik Me">
```

- The result area: a div

```
<div id="placeholder">... result here ...</div>
```

- The AWS/Ajax

```
@@INCLUDE@@ aws_action_xml.tjs onclick clickme
```

Basically it places an **onclick** attribute (the event) in the HTML **<input>** identified as **clickme** (the action) above. Here is what happen when the button is clicked:

- send the `/clickme` HTTP request to the server
- asynchronously wait for the XML answer, when received parse the answer and perform the actions according to the XML content.

To set the placeholder with **"new text"**, the XML document returned by the server must be:

```
<response>
  <replace id="placeholder">new text</replace>
</response>
```

If we want also to clear the input field named **field** and to select the radio button named **radio1** we must return:

```
<response>
  <replace id="placeholder">new text</replace>
  <clear id="field"/>
  <radio action="select" id="radio1"/>
</response>
```

This is by far the most flexible solution as it is possible to return, from the server, a structured answer.

A final comment, if the text returned by the server to replace a specific area is an HTML fragment, the content must be placed into a CDATA tag:

```
<response>
  <replace id="item_id">
    <![CDATA[ HTML CODE HERE ]]>
  </replace>
</response>
```

4.7.2.4 Advanced Ajax

Finally, if this is not enough because you need to use some specific JavaScript code, AWS provides a template to add an event to a specific widget, the action being the name of a JavaScript routine, see 'aws_action_js.tjs'.

This template together with 'aws_func_replace.tjs', 'aws_func_clear.tjs' and 'aws_func_xml.tjs' can be used to chain multiple actions. Those templates are the function body used by the corresponding templates 'aws_action_replace.tjs', 'aws_action_clear.tjs' and 'aws_action_xml.tjs'.

Let say you want to clear a widget, change the content of another one and calling one of your specific JavaScript routine when clicking on a button. It is not possible to have multiple onclick events on the same widget, the solution is the following:

- Create the JavaScript routine to do the job

For this in the the body of the clear_replace() JavaScript routine we place:

```
function clear_replace()
{
```

```
@@INCLUDE@@ aws_func_replace.tjs (clickme placeholder 4=>field)
@@INCLUDE@@ aws_func_clear.tjs (area)
call_this_routine();
```

```
}
```

Then to add the event on the widget:

```
@@INCLUDE@@ aws_action_js.tjs (onclick clickme clear_replace)
```

Furthermore, it is possible to pass (as the parameter number 20) a routine to call after a specific action to all templates. This is another way to chain multiple actions for a single event.

Note that all AWS/Ajax templates have a set of comments at the start explaining in details the usage of each parameter.

5 Using SOAP

SOAP can be used to implements Web Services. The SOAP implementation uses AWS HTTP as the transport layer. SOAP is platforms and languages independent, to ensure a good inter-operability, AWS/SOAP implementation has been validated through <http://validator.software.org/>, the version number listed on this server corresponds to the AWS version string (AWS.Version) catenated with the SOAP version string (SOAP.Version).

This SOAP implementation is certainly one with the higher level of abstraction. No need to mess with a serializer, to know what is a payload or be an XML expert. All the low level stuffs are completely hidden as the SOAP type system has been binded as much as possible to the Ada type system.

The SOAP type system has been relaxed to be compatible with WSDL based SOAP implementation. In these implementations, types are generally (as in the Microsoft implementation) not part of the payload and should be taken from the WSDL (Web Services Description Language). AWS/SOAP is not WSDL compliant at this stage, all such types are binded into the Ada type system as strings. It is up to the programmer to convert such strings to the desired type.

5.1 SOAP Client

The SOAP client interface is quite simple. Here are the step-by-step instructions to call a SOAP Web Service:

1. Build the SOAP parameters

As for the SOAP servers, the SOAP parameters are built using a SOAP.Parameters.List object.

```
Params : constant Parameters.List
:= +I (10, "v1") & I (32, "v2");
```

2. Build the SOAP Payload

The Payload object is the procedure name and the associated parameters.

```
declare
  Payload : Message.Payload.Object;
begin
  Payload := Message.Payload.Build ("Add", Params);
```

3. Call the SOAP Web Service

Here we send the above Payload to the Web Server which handles the Web Service. Let's say that this server is named myserver, it is listening on port 8082 and the SOAPAction is soapdemo.

```
Resp : constant Message.Response.Object'Class :=
  SOAP.Client.Call ("http://myserver:8082/soapdemo", Payload);
```

4. Retrieve the result

Let's say that the answer is sent back into the parameter named "myres", to get it:

```
My_Res : constant Integer := SOAP.Parameters.Get (Params, "myres");
```

In the above example we have called a Web Service whose spec could be described in Ada as follow:

```
function Add (V1, V2 : in Integer) return Integer;
-- Add V1 and V2 and returns the result. In SOAP the result is named "myres"
```

5.2 SOAP Server

A SOAP server implementation must provides a callback procedure as for standard Web server see [Section 3.2.2 \[Callback procedure\]](#), page 10. This callback must checks for the SOAP Action URI to handle both standard Web requests and SOAP ones. The `SOAPAction` is sent with the HTTP headers and can be retrieved using `AWS.Status.SOAPAction`.

5.2.1 Step by step instructions

Here are the step-by-step instructions to be followed in the SOAP callback procedure:

1. Retrieve the SOAP Payload

The SOAP Payload is the XML message, it contains the procedure name to be called and the associated parameters.

```
function SOAP_CB (Request : in AWS.Status.Data)
  return AWS.Response.Data
is
  use SOAP.Types;
  use SOAP.Parameters;

  Payload : constant SOAP.Message.Payload.Object
    := SOAP.Message.XML.Load_Payload (AWS.Status.Payload (Request));
```

`AWS.Status.Payload` returns the XML Payload as sent by the SOAP Client. This XML Payload is then parsed using `SOAP.Message.XML.Load_Payload` which returns a `SOAP.Message.Payload.Object` object.

2. Retrieve the SOAP Parameters

The SOAP procedure's parameters.

```
Params : constant SOAP.Parameters.List
  := SOAP.Message.Parameters (Payload);
```

`SOAP.Parameters.List` is a structure which holds the SOAP parameters. Each parameter can be retrieved using a `SOAP.Parameters` API, see [Section B.74 \[SOAP.Parameters\]](#), page 246. For example to get the parameter named `myStruc` which is a SOAP struct:

```
My_Struct : constant SOAP_Record
  := SOAP.Parameters.Get (Params, "myStruct");
```

Another example, to get the parameter named `myInt` which is a SOAP integer:

```
My_Int : constant Integer := SOAP.Parameters.Get (Params, "myInt");
```

3. Implements the Web Service

This is the real job, as for any procedure you can do whatever is needed to compute the result.

4. Build the SOAP answer

This is the procedure answer. A SOAP answer is built from the SOAP Payload and by setting the returned parameters.

```

declare
    Resp          : SOAP.Message.Response.Object;
    Resp_Params   : SOAP.Parameters.List;
begin
    Resp := SOAP.Message.Response.From (Payload);

    Resp_Params := +I (My_Int * 2, "answer");

    SOAP.Message.Set_Parameters (Resp, Resp_Params);

```

This build a response which is a single integer value named **answer** with the value **My_Int * 2**.

5. Returns the answer back to the client

This last step will encode the response object in XML and will returns it as the body of an HTTP message.

```

return SOAP.Message.Response.Build (Resp);

```

5.2.2 SOAP helpers

There is two ways to help building the SOAP callbacks. AWS provides a SOAP specific callback, the spec is :

```

function SOAP_Callback
(SOAPAction : in String;
Payload      : in Message.Payload.Object;
Request      : in AWS.Status.Data)
return AWS.Response.Data;

```

With both solutions exposed below, AWS retrieve the SOAPAction and the Payload from the SOAP request. This is transparent to the user.

1. Using Utils.SOAP_Wrapper

It is possible to dispatch to such callback by using the SOAP.Utils.SOAP_Wrapper generic routine.

```

generic
    with function SOAP_CB
        (SOAPAction : in String;
         Payload      : in Message.Payload.Object;
         Request      : in AWS.Status.Data)
        return AWS.Response.Data;
    function SOAP_Wrapper
        (Request : in AWS.Status.Data)
        return AWS.Response.Data;
-- From a standard HTTP callback call the SOAP callback passed as generic
-- formal procedure. Raise Constraint_Error if Request is not a SOAP
-- request.

```

For example, from the standard HTTP callback CB we want to call SOAP_CB for all SOAP requests:

```

function SOAP_CB
(SOAPAction : in String;
 Payload    : in Message.Payload.Object;
 Request    : in AWS.Status.Data)
return AWS.Response.Data
is
begin
    -- Code here
end SOAP_CB;

procedure SOAP_Wrapper is new SOAP.Utils.SOAP_Wrapper (SOAP_CB);

function CB (Request : in AWS.Status.Data)
return AWS.Response.Data
is
    SOAPAction : constant String := Status.SOAPAction (Request);
begin
    if SOAPAction /= "" then
        SOAP_Wrapper (Request);
    else
        ...
    end if;
end CB;

```

2. Using a SOAP Dispatcher

AWS provides also a SOAP specific dispatcher. This dispatcher will automatically calls a standard HTTP or SOAP callback depending on the request. If `SOAPAction` is specified (i.e. it is a SOAP request), the dispatcher will call the SOAP callback otherwise it will call the standard HTTP callback. This is by far the easiest integration procedure. Using dispatcher the above code will be written:

```

function SOAP_CB
(SOAPAction : in String;
 Payload    : in Message.Payload.Object;
 Request    : in AWS.Status.Data)
return AWS.Response.Data
is
begin
    -- Code here
end SOAP_CB;

function CB (Request : in AWS.Status.Data)
return AWS.Response.Data
is
    SOAPAction : constant String := Status.SOAPAction (Request);
begin
    -- Code here
end CB;

-- In the main procedure
begin
    AWS.Server.Start
    (WS,
     Dispatcher =>
        SOAP.Dispatchers.Callback.Create (CB'Access, SOAP_CB'Access),
     Config     =>
        AWS.Config.Default_Config);
end;

```

The dispatcher is created using `SOAP.Dispatchers.Callback.Create`. This routine takes two parameters, one is the standard HTTP callback procedure and the other is the SOAP callback procedure.

6 Using WSDL

WSDL (Web Service Definition Language) is an XML based document which described a set of Web Services either based on SOAP or XML/RPC. By using a WSDL document it is possible to describe, in a formal way, the interface to any Web Services. The WSDL document contains the end-point (URL to the server offering the service), the SOAPAction (needed to call the right routine), the procedure names and a description of the input and output parameters.

AWS provides two tools to work with WSDL documents:

`'ada2wsdl'`

which creates a WSDL document from an Ada package spec.

`'wsdl2aws'`

which create the interfaces to use a Web Service or to implement Web Services. With this tool the SOAP interface is completely abstracted out, users will deal only with Ada API. All the SOAP marshaling will be created automatically.

6.1 Creating WSDL documents

Note that this tool is based on ASIS.

6.1.1 Using ada2wsdl

`ada2wsdl` can be used on any Ada spec file to generated a WSDL document. The Ada spec is parsed using ASIS.

The simplest way to use it is:

```
$ ada2wsdl simple.ads
```

Given the following Ada spec file:

```
package Simple is
  function Plus (Value : in Natural) return Natural;
end Simple;
```

It will generate the following WSDL document:

```

<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Simple"
  targetNamespace="urn:aws:Simple"
  xmlns:tns="urn:aws:Simple"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <message name="Plus_Request">
    <part name="Value" type="xsd:int"/>
  </message>

  <message name="Plus_Response">
    <part name="Result" type="xsd:int"/>
  </message>

  <portType name="Simple_PortType">
    <operation name="Plus">
      <input message="tns:Plus_Request"/>
      <output message="tns:Plus_Response"/>
    </operation>
  </portType>

  <binding name="Simple_Binding" type="tns:Simple_PortType">
    <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>

    <operation name="Plus">
      <soap:operation soapAction="Plus"/>
      <input>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:aws:Simple"
          use="encoded"/>
      </input>
      <output>
        <soap:body
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
          namespace="urn:aws:Simple"
          use="encoded"/>
      </output>
    </operation>
  </binding>

  <service name="Simple_Service">
    <port name="Simple_Port" binding="tns:Simple_Binding">
      <soap:address location="http://.../">
    </port>
  </service>
</definitions>

```

In bold are marked the important parts from a spec point of view. The first item is the name of the WSDL document (the name of the Ada spec package). On the **portType** section we have the description of the Ada **Plus** function. Something important to note is that in Ada a function does not have a named return parameter, ‘ada2wsdl’ use **Result** for the response. Both the input and output parameter are mapped to SOAP **xsd:int** type.

Note that the SOAP address generated by default (<http://.../>) must be edited manually or specified using ‘ada2wsdl’'s -a option.

This is of course a very simple example. `ada2wsdl` does support lot more complex specs and will map Ada records, arrays, enumerations, derived types to a corresponding XML schema definition. See section below for a description of the mapping.

6.1.2 Ada mapping to WSDL

`ada2wsdl` parse Ada records, arrays, derived types, enumerations, procedures and functions and generate the corresponding WSDL document. In this section we describe the mapping between Ada and WSDL.

Integer Mapped to `xsd:int`.

Float Mapped to `xsd:float`. Note that `Float` type should not be used as `xsd:float` is really an Ada `Long_Float`. ‘`ada2wsdl`’ will issue a warning if such type is used. Note also that full interoperability is not supported for type `Float`.

Long_Float
Mapped to `xsd:float`

Long_Long_Float
Mapped to `xsd:double`

Boolean Mapped to `xsd:boolean`

String Mapped to `xsd:string`

Unbounded_String
Mapped to `xsd:string`, note that `Unbounded_String` should be used only inside a record for full interoperability. This is a current limitation.

Character Mapped to a Character schema definition.

```
<simpleType name="Character">
  <restriction base="xsd:string">
    <length value="1"/>
  </restriction>
</simpleType>
```

SOAP.Utls.SOAP_Base64

Mapped to `xsd:base64Binary`. `SOAP.Utls.SOAP_Base64` is a subtype of string which is recognized by `ada2wsdl` to generate the proper SOAP type.

SOAP.Types.Byte

Mapped to `xsd:byte`. `SOAP.Types.Byte` is a type which is recognized by `ada2wsdl` to generate the proper SOAP type.

SOAP.Types.Short

Mapped to `xsd:short`. `SOAP.Types.Short` is a type which is recognized by `ada2wsdl` to generate the proper SOAP type.

SOAP.Types.Long

Mapped to `xsd:long`. `SOAP.Types.Long` is a type which is recognized by `ada2wsdl` to generate the proper SOAP type.

SOAP.Types.Unsigned_Byte

Mapped to `xsd:unsignedByte`. `SOAP.Types.Unsigned_Byte` is a type which is recognized by `ada2wsdl` to generate the proper SOAP type.

SOAP.Types.Unsigned_Short

Mapped to `xsd:unsignedShort`. `SOAP.Types.Unsigned_Short` is a type which is recognized by `ada2wsdl` to generate the proper SOAP type.

SOAP.Types.Unsigned_Int

Mapped to **xsd:unsignedInt**. `SOAP.Types.Unsigned_Int` is a type which is recognized by `ada2wsdl` to generate the proper SOAP type.

SOAP.Types.Unsigned_Long

Mapped to **xsd:unsignedLong**. `SOAP.Types.Unsigned_Long` is a type which is recognized by `ada2wsdl` to generate the proper SOAP type.

Enumerations

Mapped to an enumeration schema definition. For example:

```
type Color is (Red, Green, Blue);
```

is defined as:

```
<simpleType name="Color">
  <restriction base="xsd:string">
    <enumeration value="Red"/>
    <enumeration value="Green"/>
    <enumeration value="Blue"/>
  </restriction>
</simpleType>
```

Records

Mapped to a struct schema definition. For example:

```
type Rec is record
  A : Integer;
  B : Long_Float;
  C : Long_Long_Float;
  D : Character;
  E : Unbounded_String;
  F : Boolean;
end record;
```

is defined as:

```
<complexType name="Rec">
  <all>
    <element name="A" type="xsd:int"/>
    <element name="B" type="xsd:float"/>
    <element name="C" type="xsd:double"/>
    <element name="D" type="tns:Character"/>
    <element name="E" type="xsd:string"/>
    <element name="F" type="xsd:boolean"/>
  </all>
</complexType>
```

Arrays

Mapped to an array schema definition. For example:

```
type Set_Of_Rec is array (Positive range <>) of Rec;
```

is defined as:


```

<complexType name="Set_Of_Rec">
  <complexContent>
    <restriction base="soap-enc:Array">
      <attribute ref="soap-enc:arrayType" wsdl:arrayType="tns:Rec[]" />
    </restriction>
  </complexContent>
</complexType>

```

Array inside a record

This part is a bit delicate. A record field must be constrained but a SOAP arrays is most of the time not constrained at all. To support this AWS use a safe access array component. Such a type is built using a generic runtime support package named `SOAP.Utills.Safe_Pointers`. This package implements a reference counter for the array access and will release automatically the memory when no more reference exists for a given object.

For example, let's say that we have an array of integer that we want to put inside a record:

```
type Set_Of_Int is array (Positive range <>) of Integer;
```

The first step is to create the safe array access support:

```

type Set_Of_Int_Access is access Set_Of_Int;

package Set_Of_Int_Safe_Pointer is
  new SOAP.Utills.Safe_Pointers (Set_Of_Int, Set_Of_Int_Access);

```

Note that the name `Set_Of_Int_Safe_Pointer` (*<type>_Safe_Pointer*) is mandatory (and checked by 'ada2wsdl') to achieve interoperability with 'wsdl2aws'. see [Section 6.2 \[Working with WSDL documents\], page 52](#).

From there the safe array access can be placed into the record:

```

type Complex_Rec is record
  SI : Set_Of_Int_Safe_Pointer.Safe_Pointer;
end record;

```

To create a `Safe_Pointer` given a `Set_Of_Int` one must use `Set_Of_Int_Safe_Pointer.To_Safe_Pointer` routine. Accessing individual items is done with `SI.Item (K)`.

These Ada definitions are fully recognized by 'ada2wsdl' and will generate standard array and record WSDL definitions as seen above.

```

<complexType name="Set_Of_Int">
  <complexContent>
    <restriction base="soap-enc:Array">
      <attribute ref="soap-enc:arrayType" wsdl:arrayType="xsd:int[]" />
    </restriction>
  </complexContent>
</complexType>

<complexType name="Complex_Rec">
  <all>
    <element name="SI" type="tns:Set_Of_Int"/>
  </all>
</complexType>

```

6.1.3 ada2wsdl

```
Usage: ada2wsdl [options] ada_spec
```

ada2wsdl options are:

- a url** Specify the URL for the Web Server address. Web Services will be available at this address. A port can be specified on the URL, `http://server[:port]`. The default value is `http://.../`.
- f** Force creation of the WSDL file. Overwrite exiting file with the same name.
- I path** Add path option for the ASIS compilation step. This option can appear any number of time on the command line.
- noenum** Do not generate WSDL representation for Ada enumerations, map them to standard string. see [Section 6.1.2 \[Ada mapping to WSDL\]](#), page 49.
- o file** Generate the WSDL document into file.
- q** Quiet mode (no output)
- s name** Specify the Web Service name for the WSDL document, by default the spec package's name is used.
- v** Verbose mode, display the parsed spec.

6.1.4 'ada2wsdl' limitations

- Do not handle constraint arrays into a records.
- Unbounded_String are supported with full interoperability only inside a record.
- Only unconstraint arrays are supported
- Arrays with multiple dimentions not supported

6.2 Working with WSDL documents

6.2.1 Client side (stub)

This section describe how to use a Web Service. Let's say that we want to use the Barnes & Noble Price Quote service. The WSDL document for this service can be found at <http://www.xmethods.net/sd/2001/BNQuoteService.wsdl>. In summary this document says that there is a service named `getPrice` taking as input a string representing the ISBN number and returning the price as floating point.

The first step is to generate the client interface (stub):

```
$ wsd2aws -noskel http://www.xmethods.net/sd/2001/BNQuoteService.wsdl
```

This will create many files, the interesting one at this point is `'bnquoteservice-client.ads'`, inside we have:

```
function getPrice
  (isbn : in String)
  return Long_Float;
-- Raises SOAP.SOAP_Error if the procedure fails
```

Let's call this service to find out the price for *The Sword of Shannara Trilogy* book.

```

with Ada.Text_IO;
with BNQuoteService.Client;

procedure Price is
  use Ada;

  ISBN : constant String := "0345453751";
  -- The Sword of Shannara Trilogy ISBN

  package LFIO is new Text_IO.Float_IO (Long_Float);

begin
  Text_IO.Put_Line ("B&N Price for The Sword of Shannara Trilogy");
  LFIO.Put (BNQuoteService.Client.getPrice (ISBN), Aft => 2, Exp => 0);
end Price;

```

That's all is needed to use this Web Service. This program is fully functional, It is possible to build it and to run it to get the answer.

6.2.2 Server side (skeleton)

Building a Web Service can also be done from a WSDL document. Let's say that you are Barnes & Noble and that you want to build Web Service `getPrice` as described in the previous section. You have created the WSDL document to specify the service spec. From there you can create the skeleton:

```
$ wsd12aws -nostub http://www.xmethods.net/sd/2001/BNQuoteService.wsdl
```

This will create many files, the interesting one here is `'bnquoteservice-server.ads'`, inside we have:

```

Port : constant := 80;

generic
  with function getPrice
    (isbn : in String)
    return Long_Float;
function getPrice_CB
  (SOAPAction : in String;
   Payload    : in SOAP.Message.Payload.Object;
   Request    : in AWS.Status.Data)
  return AWS.Response.Data;

```

This is a SOAP AWS's callback routine that can be instantiated with the right routine to retrieve the price of a book given its ISBN number. A possible implementation of such routine could be:

```

function getPrice
  (isbn : in String)
  return Long_Float is
begin
  if isbn = "0987654321" then
    return 45.0;
  elsif ...
end getPrice;

function SOAP_getPrice is new BNQuoteService.Server.getPrice_CB (getPrice);

```

`SOAP_getPrice` is a SOAP AWS's callback routine (i.e. it is not a standard callback). To use it there is different solutions:

Using SOAP.Utils.SOAP_Wrapper

This generic function can be used to translate a standard callback based on `AWS.Status.Data` into a SOAP callback routine.

```
function getPrice_Wrapper is new SOAP.Utils.SOAP_Wrapper (SOAP_getPrice);
```

The routine `getPrice_Wrapper` can be used as any other AWS's callback routines. Note that inside this wrapper the XML payload is parsed to check the routine name and to retrieve the SOAP parameters. To call this routine the payload needs to be parsed (we need to know which routine has been invoked). In this case we have parsed the XML payload twice, this is not efficient.

Building the wrapper yourself

This solution is more efficient if there are many SOAP procedures as the payload is parsed only once.

```
function CB (Request : in Status.Data) return Response.Data is
  SOAPAction : constant String := Status.SOAPAction (Request);
  Payload     : constant SOAP.Message.Payload.Object
    := SOAP.Message.XML.Load_Payload (AWS.Status.Payload (Request));
  Proc        : constant String
    := SOAP.Message.Payload.Procedure_Name (Payload);
begin
  if SOAPAction = "..." then

    if Proc = "getPrice" then
      return SOAP_getPrice (SOAPAction, Payload, Request);
    elsif ...
      ...
    end if;

  else
    ...
  end if;
```

Note that the port to be used by the AWS server is described into the server spec.

6.2.3 wsdl2aws

Usage: `wsdl2aws [options] <file|URL>`

It is possible to pass a WSDL file or direct 'wsdl2aws' to a WSDL document on the Web by passing its URL.

`wsdl2aws` options are:

- q** Quiet mode (no output)
- d** Generate debug code. Will output some information about the payload to help debug a Web Service.
- a** Generate using Ada style names. For example `getPrice` will be converted to `Get_Price`. This formatting is done for packages, routines and formal parameters.
- f** Force creation of the file. Overwrite any existing files with the same name.
- e** Specify the default endpoint to use instead of the one found in the WSDL document.
- s** Skip non supported SOAP routines. If `-s` is not used, `wsdl2aws` will exit with an error when a problem is found while parsing the WSDL document. This option is useful to skip routines using non supported types and still be able to compile the generated files.

- o name** Specify the name of the local WSDL document. This option can be used only when using a Web WSDL document (i.e. passing an URL to `wsdl2aws`).
- doc** Handle document style binding as RPC ones. This is sometimes needed because some WSDL document specify a document style binding even though it is really an RPC one.
- v** Verbose mode, display the parsed spec.
- v -v** Verbose mode, display the parsed spec and lot of information while parsing the WSDL document tree.
- wsdl** Add WSDL document as comment into the generated root unit.
- cvs** Add CVS d tag in every generated file.
- nostub** Do not generated stubs, only skeletons are generated.
- noskel** Do not generated skeletons, only stubs are generated.
- cb** Generate a SOAP dispatcher callback routine for the server. This dispatcher routine contains the code to handle all the operations as described in the WSDL document. You need also to specify the `-spec` and/or `-types` options, see below.
- x operation** Add `operation` to the list of SOAP operations to skip during the code generation. It is possible to specify multiple `-x` options on the command line.
- spec spec** Specify the name of the spec containing the Ada implementation of the SOAP routines. This is used for example by the `-cb` option above to instantiate all the server side SOAP callbacks used by the main SOAP dispatcher routine. If `-types` is not specified, the type definitions are also used from this spec.
- types spec** Specify the name of the spec containing the Ada types (record, array) used by SOAP routines specified with option `-spec`. If `-spec` is not specified, the spec definitions are also used from this spec.
- main filename** Specify the name of the server's procedure main to generate. If file '`<filename>.amt`' (Ada Main Template) is present, it uses this template file to generate the main procedure. The template can reference the following variable tags:

SOAP_SERVICE

The name of the service as described into the WSDL document. This tag can be used to include the right units

```
with @_SOAP_SERVICE_@.Client;
with @_SOAP_SERVICE_@.CB;
```

SOAP_VERSION

The AWS's SOAP version.

AWS_VERSION

The AWS's version.

UNIT_NAME

The name of the generated unit. This is the name of the procedure that will be created.

```

procedure @_UNIT_NAME_@ is
begin
    ...

```

-proxy name|IP

Use this proxy to access the WSDL document and generate code to access to these Web Services via this proxy. The proxy can be specified by its DNS name or IP address.

-pu name User name for the proxy if proxy authentication required.

-pp password

User password for the proxy if proxy authentication required.

6.2.4 wsdl2aws behind the scene

The **wsdl2aws** tool read a WSDL document and creates a root package and a set of child packages as described below:

<root> This is the main package, it contains eventually the full WSDL in comment and the description of the services as read from the WSDL document.

<root>.Types

This package contains the definitions of the types which are not SOAP base types. We find here the definitions of the SOAP structs and arrays with routines to convert them between the Ada and SOAP type model. A subtype definition is also created for every routine's returned type. In fact, all definitions here are only alias or renaming of types and/or routines generated in other packages. The real definitions for structs, arrays, enumerations and derived types are generated into a package whose name depends on the name space used for these entities. This package act as a container for all definitions and it is the only one used in the other generated packages.

<root>.Client

All spec to call Web Services.

<root>.Server

All spec to build Web Services. These specs are all generic and must be instantiated with the right routine to create the web services.

<root>.CB The SOAP dispatcher callback routine.

6.2.5 wsdl2aws limitations

It is hard to know all current limitations as the WSDL and SOAP world is quite complex. We list there all known limitations:

- Some SOAP base types are not supported : *date*, *time*, *xsd:hexBinary*, *decimal*. All these are easy to add (except decimal), it is just not supported with the current version.
- Multi-dimension arrays are not supported.
- abstract types are not supported.
- SOAP MIME attachments are not supported.
- WSDL type inheritance not supported.
- Only the RPC/Encoded SOAP messages' style is supported (the Document/Literal is not)

6.3 Using ada2wsdl and wsdl2aws together

Using both tools together is an effective way to build rapidly a SOAP server. It can be said that doing so is quite trivial in fact. Let's take the following spec:

```
package Graphics is
  type Point is record
    X, Y : Long_Float;
  end record;

  function Distance (P1, P2 : in Point) return Long_Float;
  -- Returns the distance between points P1 and P2

end Graphics;
```

We do not show the body here but we suppose it is implemented. To build a server for this service it is as easy as:

```
$ ada2wsdl -a http://localhost:8787 -o graphics.wsdl graphics.ads
```

The server will be available on localhost at port 8787.

```
$ wsdl2aws -cb -main server -types graphics graphics.wsdl
$ gnatmake server -larges ...
```

Options

- cb is to create the SOAP dispatcher callback routine,
- main server to generate the main server procedure in `'server.adb'`,
- types graphics to use `'graphics.ads'` to get references from user's spec (reference to `Graphics.Point` for example).

7 Working with mails

7.1 Sending e-mail

AWS provides a complete API to send e-mail using SMTP protocol. You need to have access to an SMTP server to use this feature. The API covers sending simple mail with text message and/or with MIME attachments (base64 encoded). Here are the steps to send a simple e-mail:

- Initialize the SMTP server

```
SMTP_Server : SMTP.Receiver
:= SMTP.Client.Initialize ("smtp.hostname");
```

Here AWS uses the default SMTP port to create an SMTP mail server but it is possible to specify a different one. The hostname specified must be a valid SMTP server.

- Send the e-mail

To send an e-mail there is many different API. Let's send a simple text mail.

```
Status : SMTP.Status;

SMTP.Client.Send
(SMTP_Server,
 From    => SMTP.E_Mail ("Pascal Obry", "p.obry@wanadoo.fr"),
 To      => SMTP.E_Mail ("John Doe", "john.doe@here.com"),
 Subject => "About AWS SMTP protocol",
 Message => "AWS can now send mails",
 Status  => Status);
```

Here Status will contain the SMTP returned status.

- Check that everything is ok

Using above status data it is possible to check that the message was sent or not by the server. The status contain a code and an error message, both of them can be retrieved using specific routines, See [Section B.62 \[AWS.SMTP\], page 221](#). It is also possible to check that the call was successful with `SMTP.Is_Ok` routine.

```
if not SMTP.Is_Ok (Status) then
  Put_Line ("Can't send message: " & SMTP.Status_Message (Status));
end if;
```

In the above example, the message content was given as a string but it is possible to specify a disk file. AWS can also send MIME messages either from disk files or with in memory base64 encoded binary data. The API provides also a way to send messages to multiple recipients at the same time. These features are not described here, complete documentation can be found on the spec see [Section B.62 \[AWS.SMTP\], page 221](#) and see [Section B.63 \[AWS.SMTP.Client\], page 224](#).

7.2 Retrieving e-mail

AWS provides an API to retrieve e-mails from a POP mailbox. POP stands for *Post Office Protocol* and is the main protocol used by Internet Service Providers around the world. IMAP is another well known protocol in this area but it is not supported by AWS.

We describes here the POP API. For a complete description see see [Section B.30 \[AWS.POP\], page 150](#).

- Opening the mailbox

The first step is to authenticate using a user name and password. AWS supports two methods one called `Clear_Text` which is the most used and another one `APOP` which is more secure but almost not supported by ISP for the moment (and will probably never be supported as a more secure protocol named `SPA` -Secure Password Authentication- could be used instead).

```
Mailbox : POP.Mailbox
:= POP.Initialize ("pop.hostname", "john.does", "mysuperpwd");
```

The default Authentication method is `Clear_Text`.

- Getting mailbox information

When the connection is opened it is possible to get information about the mailbox like the number of messages or the total number of bytes in the mailbox.

```
N      : constant Natural := POP.Message_Count (Mailbox);

Bytes : constant Natural := POP.Size (Mailbox);
```

- Retrieving individual e-mail

Each message is numbered starting from 1. A function named `Get` will return a message given its mailbox's number.

```
Message : constant POP.Message := POP.Get (Mailbox, 2, Remove => True);
```

`Remove` can be set to `False` for the message to stay on the mailbox. The default value is `False`.

- Iterating through the mailbox content

Another way to retrieve message is by using an iterator.

```
procedure Print_Subject
(Message : in POP.Message
 Index   : in Positive;
 Quit    : in out Boolean) is
begin
  Text_IO.Put_Line (POP.Subject (Message));
end Print_Message;

procedure Print_All_Subjects is new POP.For_Every_Message (Print_Subject);

...

Print_All_Subjects (Mailbox, Remove => True);
```

It exists a set of routines on a `POP.Message` object to get the subject the content, the date or any headers. It is also possible to work with attachments. See point below.

- Working with attachments

A message can have a set of MIME attachments. The number of attachments can be retrieved using `Attachment_Count`.

```
Message : constant POP.Message := ...;

A_Count : constant Natural := POP.Attachment_Count (Message);
```

As for messages it is possible to get a single attachment using its index in the message or by using an iterator.

```
First_Attachment : constant POP.Attachment := POP.Get (Message, 1);

procedure Write_Attachment
  (Attachment : in     POP.Attachment
   Index       : in     Positive;
   Quit        : in out Boolean) is
begin
  POP.Write (Attachment, Directory => ".");
end Print_Message;

procedure Write_All_Attachments is new POP.For_Every_Attachment (Write_Attachment);

...

Write_All_Attachments (Message);
```

It is also possible to retrieve the attachment's filename, the content as a memory stream. see [Section B.30 \[AWS.POP\]](#), page 150.

- Closing the connection

```
POP.Close (POP_Server);
```


8 LDAP

AWS provides a complete API to retrieve information from LDAP servers. Note that there is no support for updating, modifying or deleting information only to read information from the server.

The AWS/LDAP implementation is based on `OpenLDAP`. To build an LDAP application you need to link with the `'libldap.a'` library. This library is built by AWS on Windows based system and will use the `'wldap32.dll'` as provided with Windows NT/2000/XP. On UNIX based systems, you must install properly the `OpenLDAP` package.

The steps required to read information from an LDAP server are:

Initialize the LDAP directory

```
declare
  Directory : LDAP.Client.Directory;
begin
  Directory := LDAP.Client.Init (Host);
```

Host is the hostname where the LDAP directory is running. It is possible to specify the port if the LDAP server does not use the default one.

Bind to the LDAP server

This step is the way to pass a login/password if the LDAP server required an authentication. If not, the login/password must be empty strings.

```
LDAP.Client.Bind (Directory, "", "");
```

Do the search

For the search you must specify the base name, a filter, the scope and a set of attributes to retrieve.

```
Response_Set := LDAP.Client.Search
(Directory, Base_DN, Filter, LDAP.Client.LDAP_Scope_Subtree,
 LDAP.Client.Attributes
 ("cn", "sn", "telephonenumber"));
```

Attributes The set of attributes to retrieve from the directory.

Filter A set of values for some attributes. A filter is `<attribute_name>=<value>` where value can contain `'*'` at the end. For example `"(cn=DUPON*)"` will look for all entries where the common name is starting by the string `"DUPON"`.

Scope Define how far in the hierarchical directory the search will operate. It is either one level, all subtrees or on the base of the tree.

For more information see [Section B.19 \[AWS.LDAP.Client\]](#), page 118.

Iterate through the response set

For this there is two iterators. `First_Entry/Next_Entry` or the generic high level iterator `For_Every_Entry`.

```

declare
  Message : LDAP.Client.LDAP_Message;
begin
  Message := LDAP.Client.First_Entry (Directory, Response_Set);

  while Message /= LDAP.Client.Null_LDAP_Message loop

    Do_Job (Message);

    Message := LDAP.Client.Next_Entry (Directory, Message);
  end loop;
end;

```

Read attributes for each entry

Each entry has an associated set of attributes. To retrieve attributes values there is two iterators. `First_Attribute` / `Next_Attribute` or the generic high level iterator `For_Every_Attribute`.

```

declare
  BER : aliased LDAP.Client.BER_Element;

  Attr : constant String := LDAP.Client.First_Attribute
    (Directory, Message, BER'Unchecked_Access);
begin
  Do_Job (Attr);

  loop
    declare
      Attr : constant String := LDAP.Client.Next_Attribute
        (Directory, Message, BER);
    begin
      exit when Attr = "";
      Do_Job (Attr);
    end;
  end loop;
end;

```

Cleanup

At the end of the processing it is important to release memory associated with LDAP objects.

```

LDAP.Client.Free (Message);
LDAP.Client.Unbind (Directory);

```

see [Section B.19 \[AWS.LDAP.Client\]](#), page 118 for all high level supported API and documentation.

Note that for complete information about AWS/LDAP you should read an LDAP API description. AWS/LDAP is only a binding and follow the LDAP API closely.

9 Jabber

AWS support part of the Jabber protocol. At this stage only two kind of messages are supported:

1. Presence

To check the presence status of a specific JID (Jabber ID)

2. Message

To send messages to a specific JID (Jabber ID)

Note that if you want an application to check the presence or send message to users it is recommended to create a specific Jabber ID on the server for this application and ask users to accept this specific user to check their presence status.

9.1 Jabber presence

To check for the presence of another JID you must first have the right to do so. The jabber server won't let you see presence of another JID unless the JID have permitted you to see its presence.

- First declare the server and status objects

```
Server : AWS.Jabber.Server;  
Status : AWS.Jabber.Presence_Status;
```

- Connect to the server, you must have an account created and must know the login and password

```
AWS.Jabber.Connect  
(Server, "jabber.domain.org", "joe", "mysuperpwd");
```

- Then, to check the presence of user "john"

```
AWS.Jabber.Check_Presence  
(Server, "john@jabber.domain.org", Status);
```

- Then, you just have to close the server

```
AWS.Jabber.Close (Server);
```

9.2 Jabber message

To send a message to a specific JID, you must connect to the server as above and close the server when you don't need to communicate with it anymore. The only different part is to send the message, here is an example:

```
Send_Message  
(Server,  
  JID    => "john@jabber.domain.org",  
  Subject => "Hello there!",  
  Content => "Are you using AWS ?");
```

It is as simple as that !

10 Resources

AWS support embedded resources. It means that it is possible to build a fully self dependent executable. This is useful when distributing a server. The server program contains the code but also the images (PNG, JPEG, GIF), the templates, the HTML pages... more generally any file the Web Server must serve to clients.

10.1 Building resources

To embed the files into the executable you must build a resource tree. This task is greatly simplified using ‘AWSRes’ tool. For example let’s say that you want to build a simple server with a single page containing some text and one PNG image. The text is handled directly in the callback procedure and contain a reference to the image ‘logo.png’. To build the resource tree:

```
$ awsres logo.png
```

This will create a set of packages whose root is the unit **res** by default. The resource tree is created. see [Section 10.4 \[awsres tool\]](#), page 67 for the complete AWS’s usage description.

awsres can also compress the resource files. This can be done by using **awsres**’s **-z** option. Compressed resources are handled transparently. If the Web client supports compression the resource is sent as-is otherwise a decompression stream will be created for the resource to be decompressed on-the-fly while sending it.

10.2 Using resources

This is really the simplest step. The resource tree must be linked with your executable, to do so you just have to “with” the resource tree root into one of your program unit. This will ensure that the resource tree will be compiled and linked into the executable. **AWS** and **Templates_Parser** know about resource files and will pick them up if available.

Note that this is transparent to users. It is possible to build the very same server based on standard files or resources files. The only change in the code is to “with” or not the resource tree.

Note that **AWS** supports only a single resource tree. If more than one resource tree is included into a program only one will be seen.

10.3 Stream resources

Users can build a response directly from a stream. In this case the callback answer is built using **AWS.Response.Stream**. It creates a resource object whose operations have been inherited from **AWS.Resource.Stream.Stream_Type** and redefined by the user. So the **Read** operation can dynamically create the result stream data, the **End_Of_File** operation must returns **True** when the stream data is out and so on. This feature is useful to let users completely create and control dynamically AWS’s response content.

see [Section B.34 \[AWS.Resources.Streams\]](#), page 160.

10.4 awsres tool

AWSRes is a tool to build resource files. It creates a root package named ‘**res**’ by default and a child package for each resource file.

Usage: **awsres** [-hrquz] file1 [-uz] [file2...]

-h Display help message.

- `-q` Quiet mode.
- `-r name` Set the root unit name. Default is **res**.
- `-u` Add following files as uncompressed resources.
- `-z` Add following files as compressed resources.

11 Status page

The status page gives information about the AWS internal status. For example it returns the server socket ID, the number of simultaneous connection, the number of time a connection has been used...

To display the information AWS use a template file. The template file (default is 'aws_status.thtml') is an HTML file with some specific tags recognized by the parser. For more information about how the template parser works, please look for the template parser documentation distributed with AWS.

Here are the tag variables recognized by AWS status page:

ACCEPT_QUEUE_SIZE

see [Section 3.4 \[Configuration options\]](#), page 16.

ABORTABLE_V (vector tag)

A list of boolean. One for each connection. True means that this connection can be aborted if none is available. This is to be inserted in a template table.

ACTIVITY_COUNTER_V (vector tag)

A list of natural. One for each connection. This is the number of request the connection has answered. This counter is reset each time the connection is closed. In other word this is the number of request a keep-alive connection has processed.

ACTIVITY_TIME_STAMP_V (vector tag)

A list of date. One for each connection. This is the time of the latest request answered.

CLEANER_WAIT_FOR_CLIENT_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

CLEANER_CLIENT_HEADER_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

CLEANER_CLIENT_DATA_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

CLEANER_SERVER_RESPONSE_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

FORCE_WAIT_FOR_CLIENT_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

FORCE_CLIENT_HEADER_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

FORCE_CLIENT_DATA_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

FORCE_SERVER_RESPONSE_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

KEYS_M (matrix tag)

A list of set of keys (for each key correspond a value in the tag VALUES_L, see below). Each key in the vector tag start with an HTML "<td>" tag. This is to be able to display the key/value in column.

LOGO

A string to be placed in an img HTML tag. This is the name of the AWS logo image.

LOG (boolean tag)

This is set to true if logging is active.

LOG_FILE

The log file full pathname.

LOG_MODE

The rotating log mode, this is either NONE, DAILY, MONTHLY or EACH_RUN.

MAX_CONNECTION

see [Section 3.4 \[Configuration options\]](#), page 16.

OPENED_V (vector tag)

A list of boolean. One for each connection. True means that the socket is open. This is to be inserted in a template table.

PEERNAME_V (vector tag)

A list of peer name. One for each connection. This is the name of the last peer connected to the slot.

QUIT_V (vector tag)

A list of boolean. One for each connection. True means that the connection will terminate as soon as possible. This is to be inserted in a template table.

RECEIVE_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

SECURITY

A boolean set to True if this is a secure socket (HTTPS/SSL).

SEND_TIMEOUT

see [Section 3.4 \[Configuration options\]](#), page 16.

SERVER_NAME

see [Section 3.4 \[Configuration options\]](#), page 16.

SERVER_PORT

see [Section 3.4 \[Configuration options\]](#), page 16.

SERVER SOCK

Server socket ID.

SESSION_LIFETIME

Number of seconds to keep session information. After this period a session is obsoleted and will be removed at next cleanup.

SESSION_CLEANUP_INTERVAL

Number of seconds between each run of the session cleanup task. This task will remove all session data that have been obsoleted.

SESSIONS_TS_V (vector tag)

A list of time stamp. Each item correspond to a session last access time.

SESSIONS_TERMINATE_V (vector tag)

A list of time. Each item correspond to the time when the session will be obsoleted.

SESSIONS_V (vector tag)

A list of session ID.

SLOT_ACTIVITY_COUNTER_V (vector tag)

A list of natural. One for each connection. This is the total number of requests the slot has answered. This counter is never reseted.

SOCK_V (vector tag)

A list of sockets ID. One for each connection.

STATUS_PAGE

see [Section 3.4 \[Configuration options\]](#), page 16.

UPLOAD_DIRECTORY

see [Section 3.4 \[Configuration options\]](#), page 16.

VALUES_M (matrix tag)

A list of set of values (for each value correspond a key in the vector tag KEYS_L, see above). Each key in the vector tag start with an HTML "<td>" tag. This is to be able to display the key/value in column.

VERSION

AWS version string.

There is also all `Templates_Parser` specific tags. This is not listed here please have a look at the `Templates_Parser` documentation distributed with AWS.

Appendix A References

Here is a list of documents used to implement AWS, the SOAP support and associated services:

RFC 0821

SIMPLE MAIL TRANSFER PROTOCOL

Jonathan B. Postel
August 1982

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

RFC 1867

Network Working Group
Request For Comments: 1867
Category: Experimental

E. Nebel
L. Masinter
Xerox Corporation
November 1995

Form-based File Upload in HTML

RFC 1939

Network Working Group
Request for Comments: 1939
STD: 53
Obsoletes: 1725
Category: Standards Track

J. Myers
Carnegie Mellon
M. Rose
Dover Beach Consulting, Inc.
May 1996

Post Office Protocol - Version 3

RFC 1945

Network Working Group
Request for Comments: 1945
Category: Informational

T. Berners-Lee
MIT/LCS
R. Fielding
UC Irvine
H. Frystyk
MIT/LCS
May 1996

Hypertext Transfer Protocol -- HTTP/1.0

RFC 2049

Network Working Group
Request for Comments: 2049
Obsoletes: 1521, 1522, 1590
Category: Standards Track

N. Freed
Innosoft
N. Borenstein
First Virtual
November 1996

Multipurpose Internet Mail Extensions
(MIME) Part Five:
Conformance Criteria and Examples

RFC 2109

Network Working Group
Request for Comments: 2109
Category: Standards Track

D. Kristol
Bell Laboratories, Lucent Technologies
L. Montulli
Netscape Communications
February 1997

HTTP State Management Mechanism

RFC 2195

Network Working Group
 Request for Comments: 2195
 Category: Standards Track
 Obsoletes: 2095

J. Klensin
 R. Catoe
 P. Krumviede
 MCI
 September 1997

IMAP/POP AUTHorize Extension for Simple Challenge/Response

RFC 2554

Network Working Group
 Request for Comments: 2554
 Category: Standards Track

J. Myers
 Netscape Communications
 March 1999

SMTP Service Extension
for Authentication

RFC 2616

Network Working Group
 Request for Comments: 2616
 Obsoletes: 2068
 Category: Standards Track

R. Fielding
 UC Irvine
 J. Gettys
 Compaq/W3C
 J. Mogul
 Compaq
 H. Frystyk
 W3C/MIT
 L. Masinter
 Xerox
 P. Leach
 Microsoft
 T. Berners-Lee
 W3C/MIT
 June 1999

Hypertext Transfer Protocol -- HTTP/1.1

RFC 2617

Network Working Group
 Request for Comments: 2617
 Obsoletes: 2069
 Category: Standards Track

J. Franks
 Northwestern University
 P. Hallam-Baker
 Verisign, Inc.
 J. Hostetler
 AbiSource, Inc.
 S. Lawrence
 Agranat Systems, Inc.
 P. Leach
 Microsoft Corporation
 A. Luotonen
 Netscape Communications Corporation
 L. Stewart
 Open Market, Inc.
 June 1999

HTTP Authentication: Basic and Digest Access Authentication

Transport Layer Security Working Group
INTERNET-DRAFT
Expire in six months

Alan O. Freier
Netscape Communications
Philip Karlton
Netscape Communications
Paul C. Kocher
Independent Consultant
November 18, 1996

The SSL Protocol
Version 3.0

SOAP (W3C Note 08 May 2000)

Simple Object Access Protocol (SOAP) 1.1

W3C Note 08 May 2000

This version:

<http://www.w3.org/TR/2000/NOTE-SOAP-20000508>

Latest version:

<http://www.w3.org/TR/SOAP>

Authors (alphabetically):

Don Box, DevelopMentor
David Ehnebuske, IBM
Gopal Kakivaya, Microsoft
Andrew Layman, Microsoft
Noah Mendelsohn, Lotus Development Corp.
Henrik Frystyk Nielsen, Microsoft
Satish Thatte, Microsoft
Dave Winer, UserLand Software, Inc.

Copyright 2000 DevelopMentor, International Business Machines Corporation,
Lotus Development Corporation, Microsoft, UserLand Software

<http://www.w3.org/TR/SOAP/>

A Busy Developer's Guide to SOAP 1.1

By Dave Winer, Jake Savin, UserLand Software, 4/2/01.

<http://www.soapware.org/bdg>

Appendix B AWS API Reference

B.1 AWS

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2006                       --
--                               AdaCore                                         --
--                               -----
--   This library is free software; you can redistribute it and/or modify
--   it under the terms of the GNU General Public License as published by
--   the Free Software Foundation; either version 2 of the License, or (at
--   your option) any later version.
--
--   This library is distributed in the hope that it will be useful, but
--   WITHOUT ANY WARRANTY; without even the implied warranty of
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
--   General Public License for more details.
--
--   You should have received a copy of the GNU General Public License
--   along with this library; if not, write to the Free Software Foundation,
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
--
--   As a special exception, if other files instantiate generics from this
--   unit, or you link this unit with other files to produce an executable,
--   this unit does not by itself cause the resulting executable to be
--   covered by the GNU General Public License. This exception does not
--   however invalidate any other reasons why the executable file might be
--   covered by the GNU Public License.
-----

package AWS is

  pragma Pure;

  Version      : constant String := "2.3.0";

  HTTP_Version : constant String := "HTTP/1.1";

end AWS;

```

B.2 AWS.Attachments

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2004-2005                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but       --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this     --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                        --
-----

with Ada.Strings.Unbounded;
with Ada.Containers.Indefinite_Vectors;

with AWS.Headers;
with AWS.Net;

package AWS.Attachments is

  type Element is private;
  type List is tagged private;

  Empty_List : constant List;

  procedure Add
    (Attachments : in out List;
     Filename     : in String;
     Content_Id   : in String);
  -- Adds an Attachment to the list. The header of the Attachment is
  -- generated.

  procedure Add
    (Attachments : in out List;
     Filename     : in String;
     Headers      : in AWS.Headers.List);
  -- Adds an Attachment to the list

  procedure Reset
    (Attachments : in out List;
     Delete_Files : in Boolean);
  -- Reset the list to be empty. If Delete_Files is set to true the
  -- attached files are removed from the file system.

  function Count (Attachments : in List) return Natural;
  -- Returns the number of Attachments in the data

```

```

function Get
  (Attachments : in List;
   Index       : in Positive)
  return Element;
-- Returns specified Attachment

function Get
  (Attachments : in List;
   Content_Id  : in String)
  return Element;
-- Returns the Attachment with the Content Id

generic
  with procedure Action
    (Attachment : in Element;
     Index       : in Positive;
     Quit        : in out Boolean);
  procedure For_Every_Attachment (Attachments : in List);
  -- Calls action for every Attachment in Message. Stop iterator if Quit is
  -- set to True, Quit is set to False by default.

function Headers (Attachment : in Element) return AWS.Headers.List;
-- Returns the list of header lines for the attachment

function Content_Type (Attachment : in Element) return String;
-- Get value for "Content-Type:" header

function Content_Id (Attachment : in Element) return String;
-- Returns Attachment's content id

function Local_Filename (Attachment : in Element) return String;
-- Returns the local filename of the Attachment.
-- Local filename is the name the receiver used when extracting the
-- Attachment into a file.

function Length
  (Attachments : in List;
   Boundary     : in String) return Natural;
-- Returns the complete size of all attachments including the surrounding
-- boundaries.

procedure Send
  (Socket       : in AWS.Net.Socket_Type'Class;
   Attachments  : in List;
   Boundary     : in String);
-- Send all Attachments, including the surrounding boundarys, in the list
-- to the socket

private
  -- implementation removed
end AWS.Attachments;

```

B.3 AWS.Client

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2005                           --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                  --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

with Ada.Streams;
with Ada.Strings.Unbounded;

with AWS.Attachments;
with AWS.Default;
with AWS.Net.SSL.Certificate;
with AWS.Response;
with AWS.URL;
with AWS.Utils;

with ZLib;

package AWS.Client is

  Connection_Error : exception;
  -- Raised if the connection with the server cannot be established

  Protocol_Error   : exception;
  -- Raised if the client receives wrong HTTP protocol data

  No_Data          : constant String;
  -- Used as the default parameter when no data specified for a specific
  -- parameter.

  Retry_Default : constant := 0;
  -- Number of time a data is requested from the Server if the first
  -- time fails.

  type Timeouts_Values is record
    Send      : Duration;
    Receive   : Duration;
  end record;
  -- Defined the number of seconds for the send and receive timeout

```

```

No_Timeout : constant Timeouts_Values;
-- No timeout, allow infinite time to send or retrieve data

type Content_Bound is new Integer range -1 .. Integer'Last;

Undefined : constant Content_Bound := -1;

type Content_Range is record
  First, Last : Content_Bound := Undefined;
end record;
-- Range for partial download

No_Range : constant Content_Range := (Undefined, Undefined);

type Authentication_Mode is new AWS.Response.Authentication_Mode;

type Authentication_Level is private;

type Authentication_Type is private;

type Auth_Attempts_Count is private;

-----
-- Messages --
-----

function Get
  (URL          : in String;
   User         : in String          := No_Data;
   Pwd          : in String          := No_Data;
   Proxy        : in String          := No_Data;
   Proxy_User   : in String          := No_Data;
   Proxy_Pwd    : in String          := No_Data;
   Timeouts     : in Timeouts_Values := No_Timeout;
   Data_Range   : in Content_Range   := No_Range;
   Follow_Redirection : in Boolean    := False)
  return Response.Data;
-- Retrieve the message data given a specific URL. It open a connection
-- with the server and ask for the resource specified in the URL it then
-- return it in the Response.Data structure.
-- If User/Pwd are given then it uses it to access the URL.
--
-- Eventually it connect through a PROXY using if necessary the Proxy
-- authentication Proxy_User:Proxy_Pwd.
--
-- Only Basic authentication is supported (i.e. Digest is not). Digest
-- authentication is supported with the keep-alive client API, see below.
--
-- If Follow_Redirection is set to True, Get will follow the redirection
-- information for 301 status code response. Note that this is not
-- supported for keep-alive connections as the redirection could point to
-- another server.
--
-- Get will retry one time if it fails.

function Head
  (URL          : in String;
   User         : in String          := No_Data;
   Pwd          : in String          := No_Data;
   Proxy        : in String          := No_Data;
   Proxy_User   : in String          := No_Data;
   Proxy_Pwd    : in String          := No_Data;
   Timeouts     : in Timeouts_Values := No_Timeout)
  return Response.Data;
-- Idem as above but we do not get the message body.

```

```

-- Head will retry one time if it fails.

function Put
  (URL      : in String;
   Data     : in String;
   User     : in String           := No_Data;
   Pwd      : in String           := No_Data;
   Proxy    : in String           := No_Data;
   Proxy_User : in String         := No_Data;
   Proxy_Pwd : in String           := No_Data;
   Timeouts : in Timeouts_Values := No_Timeout)
  return Response.Data;
-- Send to the server URL a PUT request with Data
-- Put will retry one time if it fails.

function Post
  (URL      : in String;
   Data     : in String;
   Content_Type : in String           := No_Data;
   User     : in String           := No_Data;
   Pwd      : in String           := No_Data;
   Proxy    : in String           := No_Data;
   Proxy_User : in String         := No_Data;
   Proxy_Pwd : in String           := No_Data;
   Timeouts : in Timeouts_Values := No_Timeout;
   Attachments : in AWS.Attachments.List := AWS.Attachments.Empty_List)
  return Response.Data;
-- Send to the server URL a POST request with Data
-- Post will retry one time if it fails.

function Post
  (URL      : in String;
   Data     : in Ada.Streams.Stream_Element_Array;
   Content_Type : in String           := No_Data;
   User     : in String           := No_Data;
   Pwd      : in String           := No_Data;
   Proxy    : in String           := No_Data;
   Proxy_User : in String         := No_Data;
   Proxy_Pwd : in String           := No_Data;
   Timeouts : in Timeouts_Values := No_Timeout;
   Attachments : in AWS.Attachments.List := AWS.Attachments.Empty_List)
  return Response.Data;
-- Idem as above but with binary data.

function SOAP_Post
  (URL      : in String;
   Data     : in String;
   SOAPAction : in String;
   User     : in String           := No_Data;
   Pwd      : in String           := No_Data;
   Proxy    : in String           := No_Data;
   Proxy_User : in String         := No_Data;
   Proxy_Pwd : in String           := No_Data;
   Timeouts : in Timeouts_Values := No_Timeout;
   Attachments : in AWS.Attachments.List := AWS.Attachments.Empty_List)
  return Response.Data;
-- Send to the server URL a POST request with Data
-- Post will retry one time if it fails.

function Upload
  (URL      : in String;
   Filename : in String;
   User     : in String           := No_Data;
   Pwd      : in String           := No_Data;
   Proxy    : in String           := No_Data;

```



```

    Proxy_User : in String          := No_Data;
    Proxy_Pwd  : in String          := No_Data;
    Timeouts   : in Timeouts_Values := No_Timeout)
    return Response.Data;
-- This is a file upload request. Filename file's content will be send to
-- the server at address URL.

-----
-- Keep-Alive client implementation --
-----

type HTTP_Connection is limited private;

procedure Create
(Connection : in out HTTP_Connection;
 Host      : in String;
 User      : in String          := No_Data;
 Pwd       : in String          := No_Data;
 Proxy     : in String          := No_Data;
 Proxy_User : in String          := No_Data;
 Proxy_Pwd  : in String          := No_Data;
 Retry     : in Natural          := Retry_Default;
 Persistent : in Boolean         := True;
 Timeouts   : in Timeouts_Values := No_Timeout;
 Server_Push : in Boolean        := False;
 Certificate : in String         := Default.Client_Certificate;
 User_Agent : in String         := Default.User_Agent);
-- Create a new connection. This is to be used with Keep-Alive client API
-- below. The connection will be tried Retry times if it fails. If
-- persistent is True the connection will remain open otherwise it will be
-- closed after each request. User/Pwd are the server authentication info,
-- Proxy is the name of the proxy server to use, Proxy_User/Proxy_Pwd are
-- the proxy authentication data. Only Basic authentication is supported
-- from this routine, for Digest authentication see below. Timeouts are
-- the send/receive timeouts for each request. If Server_Push is True the
-- connection will be used to push information to the client.
-- Certificate can be set to specify the certificate filename to use for
-- the secure connection. User_Agent can be overridden to whatever you want
-- the client interface to present itself to the server.

function Get_Certificate
(Connection : in HTTP_Connection)
return AWS.Net.SSL.Certificate.Object;
-- Return the certificate used for the secure connection. If this is not a
-- secure connection, returns Net.SSL.Certificate.Undefined.

procedure Set_WWW_Authentication
(Connection : in out HTTP_Connection;
 User      : in String;
 Pwd       : in String;
 Mode      : in Authentication_Mode);
-- Sets the username password and authentication mode for the Web
-- authentication.
--
-- "Any" mean that user want to use Digest server authentication mode but
-- could use Basic if the server does not support Digest authentication.
--
-- "Basic" mean that client will send basic authentication. "Basic"
-- authentication is send with the first request and is a fast
-- authentication protocol.
--
-- "Digest" mean that the client ask for Digest authentication, it
-- requires that a first unauthorized request be sent to the server. The
-- server will answer "nonce" for the authentication protocol to continue.

```

```

procedure Set_Proxy_Authentication
  (Connection : in out HTTP_Connection;
   User       : in      String;
   Pwd        : in      String;
   Mode       : in      Authentication_Mode);
-- Sets the username, password and authentication mode for the proxy
-- authentication.

procedure Copy_Cookie
  (Source      : in      HTTP_Connection;
   Destination : in out HTTP_Connection);
-- Copy a session Id from connection Source to connection Destination.
-- Allow both connections to share the same user environment. Note that
-- user's environment are thread-safe.

function Read_Until
  (Connection : in HTTP_Connection;
   Delimiter   : in String)
return String;
-- Read data on the Connection until the delimiter (including the
-- delimiter). It can be used to retrieve the next piece of data from a
-- push server. If returned data is empty or does not terminate with the
-- delimiter the server push connection is closed.

procedure Read_Until
  (Connection : in out HTTP_Connection;
   Delimiter   : in      String;
   Result      : in out Ada.Strings.Unbounded.Unbounded_String);
-- Idem as above but returns the result as an Unbounded_String

procedure Read_Some
  (Connection : in out HTTP_Connection;
   Data       : out Ada.Streams.Stream_Element_Array;
   Last       : out Ada.Streams.Stream_Element_Offset);
-- Reads any available data from the client's connection.
-- If no data available, it will wait for some data to become available or
-- until it timeouts. Returns Last < Data'First when there is no data
-- available in the HTTP response. Connection have to be created with
-- parameter Server_Push => True.

procedure Read
  (Connection : in out HTTP_Connection;
   Data       : out Ada.Streams.Stream_Element_Array;
   Last       : out Ada.Streams.Stream_Element_Offset);
-- Reads data from the client's connection until Data buffer is filled
-- or it reached the end of the response. Returns Last < Data'Last if
-- there is no more data available in HTTP response. Connection have
-- to be created with parameter Server_Push => True.

procedure Get
  (Connection : in out HTTP_Connection;
   Result      : out Response.Data;
   URI         : in      String           := No_Data;
   Data_Range  : in      Content_Range    := No_Range);
-- Same as Get above but using a Connection

procedure Head
  (Connection : in out HTTP_Connection;
   Result      : out Response.Data;
   URI         : in      String           := No_Data);
-- Same as Head above but using a Connection

procedure Put
  (Connection : in out HTTP_Connection;
   Result      : out Response.Data;

```

```

    Data      : in      String;
    URI       : in      String      := No_Data);
-- Same as Put above but using a Connection

procedure Post
(Connection   : in out HTTP_Connection;
 Result      : out Response.Data;
 Data        : in      String;
 Content_Type : in      String      := No_Data;
 URI         : in      String      := No_Data;
 Attachments : in      AWS.Attachments.List
              := AWS.Attachments.Empty_List);
-- Same as Post above but using a Connection

procedure Post
(Connection   : in out HTTP_Connection;
 Result      : out Response.Data;
 Data        : in      Ada.Streams.Stream_Element_Array;
 Content_Type : in      String      := No_Data;
 URI         : in      String      := No_Data;
 Attachments : in      AWS.Attachments.List
              := AWS.Attachments.Empty_List);
-- Same as Post above but using a Connection

procedure Upload
(Connection : in out HTTP_Connection;
 Result    : out Response.Data;
 Filename  : in      String;
 URI       : in      String      := No_Data);
-- Same as Upload above but using a Connection

procedure SOAP_Post
(Connection : in      HTTP_Connection;
 Result    : out Response.Data;
 SOAPAction : in      String;
 Data      : in      String;
 Streaming  : in      Boolean      := False;
 Attachments : in      AWS.Attachments.List := AWS.Attachments.Empty_List);
-- Same as SOAP_Post above but using a Connection
-- Streaming is to be able to parse response XML on the fly,
-- without intermediate buffer.

procedure Close (Connection : in out HTTP_Connection);
-- Close connection, it releases all associated resources

procedure Set_Streaming_Output
(Connection : in out HTTP_Connection;
 Value      : in      Boolean);
pragma Inline (Set_Streaming_Output);
-- Call this routine with Value => True to be able to read data as a
-- stream by using Read and/or Read_Some routines above. Note that
-- Connection is already in Streaming mode if it has been created
-- with Server_Push => True.

procedure Set_Debug (On : in Boolean);
-- Set debug mode on/off. If debug is activated the request header and the
-- server response header will be displayed.

private
  -- implementation removed
end AWS.Client;
```

B.4 AWS.Client.Hotplug

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2004                           --
--                               ACT-Europe                                    --
--                               --                                           --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                               --                                           --
--  This library is distributed in the hope that it will be useful, but      --
--  WITHOUT ANY WARRANTY; without even the implied warranty of               --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--  General Public License for more details.                                  --
--                               --                                           --
--  You should have received a copy of the GNU General Public License        --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                               --                                           --
--  As a special exception, if other files instantiate generics from this    --
--  unit, or you link this unit with other files to produce an executable,   --
--  this unit does not by itself cause the resulting executable to be        --
--  covered by the GNU General Public License. This exception does not       --
--  however invalidate any other reasons why the executable file might be    --
--  covered by the GNU Public License.                                        --
-----

```

with AWS.Response;

package AWS.Client.Hotplug is

```

--  Below are two routines to register/unregister hotplug modules into
--  server. Note that such server must be configured to accept hotplug
--  modules. Password parameter is the clear text password, it will be sent
--  encoded. An authorization entry for module Name with Password (and the
--  given URL host for registration) must be found in the server's
--  authorization file. See AWS.Server.Hotplug.Activate.

```

function Register

```

  (Name      : in String;
   Password  : in String;
   Server    : in String;
   Regexp    : in String;
   URL       : in String)
  return Response.Data;

```

```

--  Register hotplug module Name into Server with address URL to respond to
--  requests matching Regexp. Server must be a valid URL, http://host:port.
--  If port is not specified the default HTTP port is used.

```

function Unregister

```

  (Name      : in String;
   Password  : in String;
   Server    : in String;
   Regexp    : in String)
  return Response.Data;

```

```

--  Unregister hotplug module Name responding to Regexp requests from
--  Server. See comment above about Password.

```

end AWS.Client.Hotplug;

B.5 AWS.Communication

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2000-2006                             --
--                                     AdaCore                                             --
--                                     -----
--  This library is free software; you can redistribute it and/or modify
--  it under the terms of the GNU General Public License as published by
--  the Free Software Foundation; either version 2 of the License, or (at
--  your option) any later version.
--
--  This library is distributed in the hope that it will be useful, but
--  WITHOUT ANY WARRANTY; without even the implied warranty of
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
--  General Public License for more details.
--
--  You should have received a copy of the GNU General Public License
--  along with this library; if not, write to the Free Software Foundation,
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
--
--  As a special exception, if other files instantiate generics from this
--  unit, or you link this unit with other files to produce an executable,
--  this unit does not by itself cause the resulting executable to be
--  covered by the GNU General Public License. This exception does not
--  however invalidate any other reasons why the executable file might be
--  covered by the GNU Public License.
-----

--  The communication protocol uses a light encoding scheme based on the HTTP
--  GET method. For standard, XML based, communication you can use the SOAP
--  protocol. This API can be convenient if you do not plan to build AWS with
--  SOAP support.

with Ada.Strings.Unbounded;

package AWS.Communication is

  use Ada.Strings.Unbounded;

  type Parameter_Set is array (Positive range <>) of Unbounded_String;

  Null_Parameter_Set : constant Parameter_Set;

  function Parameters (P1, P2, P3, P4, P5 : in String := "")
    return Parameter_Set;
  -- Constructor function to help create a Parameter_Set. This function will
  -- return a Parameter_Set array containing any parameter with a non empty
  -- string value.

private
  -- implementation removed
end AWS.Communication;

```

B.6 AWS.Communication.Client

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2003                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                                                                              --
--  This library is distributed in the hope that it will be useful, but      --
--  WITHOUT ANY WARRANTY; without even the implied warranty of               --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--  General Public License for more details.                                  --
--                                                                              --
--  You should have received a copy of the GNU General Public License         --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.           --
--                                                                              --
--  As a special exception, if other files instantiate generics from this    --
--  unit, or you link this unit with other files to produce an executable,   --
--  this unit does not by itself cause the resulting executable to be        --
--  covered by the GNU General Public License. This exception does not       --
--  however invalidate any other reasons why the executable file might be    --
--  covered by the GNU Public License.                                        --
-----

with AWS.Response;

package AWS.Communication.Client is

  function Send_Message
    (Server      : in String;
     Port       : in Positive;
     Name       : in String;
     Parameters  : in Parameter_Set := Null_Parameter_Set)
    return Response.Data;
  -- Send a message to server with a set of parameters. The destination is
  -- server is http://Server:Port, the message name is Name and the set of
  -- parameters is to be found into Parameters.
  --
  -- The complete message format is:
  --
  -- http://<Server>:<Port>/AWS_Com?HOST=<host>&NAME=<name>
  --      &P1=<param1>&P2=<param2>

end AWS.Communication.Client;

```

B.7 AWS.Communication.Server

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2004                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                                                                              --
--  This library is distributed in the hope that it will be useful, but      --
--  WITHOUT ANY WARRANTY; without even the implied warranty of               --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--  General Public License for more details.                                  --
--                                                                              --
--  You should have received a copy of the GNU General Public License         --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--  As a special exception, if other files instantiate generics from this    --
--  unit, or you link this unit with other files to produce an executable,   --
--  this unit does not by itself cause the resulting executable to be        --
--  covered by the GNU General Public License. This exception does not       --
--  however invalidate any other reasons why the executable file might be    --
--  covered by the GNU Public License.                                        --
-----

with AWS.Response;

generic

  type T (<>) is limited private; -- Data type received by this server
  type T_Access is access T;

  with function Callback
    (Server      : in String; -- Host name
     Name        : in String; -- Message name
     Context     : in T_Access;
     Parameters  : in Parameter_Set := Null_Parameter_Set)
    return Response.Data;

package AWS.Communication.Server is

  -- Each instantiation of this package will create an HTTP server waiting
  -- for incoming requests at the Port specified in the Start formal
  -- parameter. This communication server must be started with the Start
  -- procedure and can be stopped with the procedure Shutdown below.

  procedure Start (Port : in Positive; Context : in T_Access);
  -- Start communication HTTP server listening at the given port

  procedure Shutdown;
  -- Shutdown the communication HTTP server

end AWS.Communication.Server;

```

B.8 AWS.Config

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but       --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License         --
--   along with this library; if not, write to the Free Software Foundation,   --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this     --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not       --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                         --
-----

--   This package provide an easy way to handle server configuration options.
--
--   If initialization of this package is not done all functions below will
--   return the default value as declared in AWS.Default.

with Ada.Strings.Unbounded;

with AWS.Containers.String_Vectors;
with AWS.Default;

package AWS.Config is

  use Ada.Strings.Unbounded;

  type Object is private;

  Default_Config : constant Object;

  function Get_Current return Object;
  -- Returns a configuration record. This is the properties as read in files
  -- 'aws.ini' and 'progrname.ini'. This configuration object holds only the
  -- per-server options.

  -----
  -- Per Server options --
  -----

  -----
  -- Server --
  -----

  function Server_Name (O : in Object) return String;
  pragma Inline (Server_Name);
  -- This is the name of the server as set by AWS.Server.Start

```



```

function Server_Host (O : in Object) return String;
pragma Inline (Server_Host);
-- This is the server host. Can be used if the computer has a more than
-- one IP address. It is possible to have two servers at the same port
-- on the same machine, both being binded on different IP addresses.

function Server_Port (O : in Object) return Positive;
pragma Inline (Server_Port);
-- This is the server port as set by the HTTP object declaration

function Hotplug_Port (O : in Object) return Positive;
pragma Inline (Hotplug_Port);
-- This is the hotplug communication port needed to register and
-- un-register an hotplug module.

function Session (O : in Object) return Boolean;
pragma Inline (Session);
-- Returns True if the server session is activated

function Case_Sensitive_Parameters (O : in Object) return Boolean;
pragma Inline (Case_Sensitive_Parameters);
-- HTTP parameters are case sensitive

-----
-- Connection --
-----

function Max_Connection (O : in Object) return Positive;
pragma Inline (Max_Connection);
-- This is the max simultaneous connections as set by the HTTP object
-- declaration.

function Free_Slots_Keep_Alive_Limit (O : in Object) return Natural;
pragma Inline (Free_Slots_Keep_Alive_Limit);
-- The minimum number of free slots where keep-alive connections are still
-- enabled. After this limit no more keep-alive connection will be
-- accepted by the server. This parameter must be used for heavy-loaded
-- servers to make sure the server will never run out of slots. This limit
-- must be less than Max_Connection.

function Keep_Alive_Force_Limit (O : in Object) return Positive;
pragma Inline (Keep_Alive_Force_Limit);
-- Server could have more than Max_Connection keep-alive sockets. Keep
-- alive sockets are waiting for client input in the internal server socket
-- set. This parameter defines the maximum number of keep alive sockets
-- processed by the server with standard timeouts. If number of keep-alive
-- sockets becomes more than Keep_Alive_Force_Limit the server starts to
-- use shorter timeouts. If this parameter is not defined in the
-- configuration, the server uses Max_Connection * 2 as value.

function Accept_Queue_Size (O : in Object) return Positive;
pragma Inline (Accept_Queue_Size);
-- This is the size of the queue for the incoming requests. Higher this
-- value will be and less "connection refused" will be reported to the
-- client.

function Line_Stack_Size (O : in Object) return Positive;
pragma Inline (Line_Stack_Size);
-- HTTP lines stack size

-----
-- Data --
-----

```

```

function WWW_Root (O : in Object) return String;
pragma Inline (WWW_Root);
-- This is the root directory name for the server. This variable is not
-- used internally by AWS. It is supposed to be used by the callback
-- procedures who want to retrieve physical objects (images, Web
-- pages...). The default value is the current working directory.

function Upload_Directory (O : in Object) return String;
pragma Inline (Upload_Directory);
-- This point to the directory where uploaded files will be stored. The
-- directory returned will end with a directory separator.

function Directory_Browser_Page (O : in Object) return String;
pragma Inline (Directory_Browser_Page);
-- Filename for the directory browser template page

-----
-- Log --
-----

function Log_File_Directory (O : in Object) return String;
pragma Inline (Log_File_Directory);
-- This point to the directory where log files will be written. The
-- directory returned will end with a directory separator.

function Log_Filename_Prefix (O : in Object) return String;
pragma Inline (Log_Filename_Prefix);
-- This is the prefix to use for the log filename

function Log_Split_Mode (O : in Object) return String;
pragma Inline (Log_Split_Mode);
-- This is split mode for the log file. Possible values are : Each_Run,
-- Daily, Monthly and None. Any other values will raise an exception.

generic
  with procedure Field_Id (Id : in String);
procedure Log_Extended_Fields_Generic_Iterate (O : in Object);
-- Calls procedure Field_Id for each extended http log field identifier.

function Log_Extended_Fields_Length (O : in Object) return Natural;
pragma Inline (Log_Extended_Fields_Length);
-- Returns the number of extended http log fields identifiers.
-- If returned value is zero then http log is not extended.

function Error_Log_Filename_Prefix (O : in Object) return String;
pragma Inline (Error_Log_Filename_Prefix);
-- This is the prefix to use for the log filename.

function Error_Log_Split_Mode (O : in Object) return String;
pragma Inline (Error_Log_Split_Mode);
-- This is split mode for the log file. Possible values are : Each_Run,
-- Daily, Monthly and None. Any other values will raise an exception.

-----
-- Status --
-----

function Admin_URI (O : in Object) return String;
pragma Inline (Admin_URI);
-- This is the name of the admin server page as set by AWS.Server.Start.
-- It is also known as the status page.

function Status_Page (O : in Object) return String;
pragma Inline (Status_Page);
-- Filename for the status template page

```

```

function Up_Image (O : in Object) return String;
pragma Inline (Up_Image);
-- Filename for the up arrow image used in the status page

function Down_Image (O : in Object) return String;
pragma Inline (Down_Image);
-- Filename for the down arrow image used in the status page

function Logo_Image (O : in Object) return String;
pragma Inline (Logo_Image);
-- Filename for the AWS logo image used in the status page

-----
-- Timeouts --
-----

function Cleaner_Wait_For_Client_Timeout (O : in Object) return Duration;
pragma Inline (Cleaner_Wait_For_Client_Timeout);
-- Number of seconds to timeout on waiting for a client request.
-- This is a timeout for regular cleaning task.

function Cleaner_Client_Header_Timeout (O : in Object) return Duration;
pragma Inline (Cleaner_Client_Header_Timeout);
-- Number of seconds to timeout on waiting for client header.
-- This is a timeout for regular cleaning task.

function Cleaner_Client_Data_Timeout (O : in Object) return Duration;
pragma Inline (Cleaner_Client_Data_Timeout);
-- Number of seconds to timeout on waiting for client message body.
-- This is a timeout for regular cleaning task.

function Cleaner_Server_Response_Timeout (O : in Object) return Duration;
pragma Inline (Cleaner_Server_Response_Timeout);
-- Number of seconds to timeout on waiting for client to accept answer.
-- This is a timeout for regular cleaning task.

function Force_Wait_For_Client_Timeout (O : in Object) return Duration;
pragma Inline (Force_Wait_For_Client_Timeout);
-- Number of seconds to timeout on waiting for a client request.
-- This is a timeout for urgent request when resources are missing.

function Force_Client_Header_Timeout (O : in Object) return Duration;
pragma Inline (Force_Client_Header_Timeout);
-- Number of seconds to timeout on waiting for client header.
-- This is a timeout for urgent request when resources are missing.

function Force_Client_Data_Timeout (O : in Object) return Duration;
pragma Inline (Force_Client_Data_Timeout);
-- Number of seconds to timeout on waiting for client message body.
-- This is a timeout for urgent request when resources are missing.

function Force_Server_Response_Timeout (O : in Object) return Duration;
pragma Inline (Force_Server_Response_Timeout);
-- Number of seconds to timeout on waiting for client to accept answer.
-- This is a timeout for urgent request when resources are missing.

function Send_Timeout (O : in Object) return Duration;
pragma Inline (Send_Timeout);
-- Number of seconds to timeout when sending chunk of data

function Receive_Timeout (O : in Object) return Duration;
pragma Inline (Receive_Timeout);
-- Number of seconds to timeout when receiving chunk of data.

```

```

-----
-- Security --
-----

function Check_URL_Validity (O : in Object) return Boolean;
pragma Inline (Check_URL_Validity);
-- Server have to check URI for validity. For example it checks that an
-- URL does not reference a resource above the Web root.

function Security (O : in Object) return Boolean;
pragma Inline (Security);
-- Is the server working through th SSL

function Certificate (O : in Object) return String;
pragma Inline (Certificate);
-- Returns the certificate to be used with the secure server. Returns the
-- empty string if the server is not a secure one.

function Key (O : in Object) return String;
pragma Inline (Key);
-- Returns the key to be used with the secure server. Returns the
-- empty string if the server is not a secure one.

function Security_Mode (O : in Object) return String;
pragma Inline (Security_Mode);
-- Returns the security mode to be used with the secure server. Returns the
-- empty string if the server is not a secure one.

function Exchange_Certificate (O : in Object) return Boolean;
pragma Inline (Exchange_Certificate);
-- Returns True if the client is requested to send its certificate to the
-- server. Note that this option must not be used if the client is a Web
-- Browser.

-----
-- Per Process options --
-----

function Session_Cleanup_Interval return Duration;
pragma Inline (Session_Cleanup_Interval);
-- Number of seconds between each run of the cleaner task to remove
-- obsolete session data.

function Session_Lifetime return Duration;
pragma Inline (Session_Lifetime);
-- Number of seconds to keep a session if not used. After this period the
-- session data is obsoleted and will be removed during next cleanup.

function Transient_Cleanup_Interval return Duration;
pragma Inline (Transient_Cleanup_Interval);
-- Number of seconds between each run of the cleaner task to remove
-- transient pages.

function Transient_Lifetime return Duration;
pragma Inline (Transient_Lifetime);
-- Number of seconds to keep a transient page. After this period the
-- transient page is obsoleted and will be removed during next cleanup.

function Max_Concurrent_Download return Positive;
pragma Inline (Max_Concurrent_Download);
-- Number of maximum concurrent download supported by the download manager
-- service.

private
-- implementation removed

```

```
end AWS.Config;
```

B.9 AWS.Config.Ini

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                                                                              --
--  This library is distributed in the hope that it will be useful, but      --
--  WITHOUT ANY WARRANTY; without even the implied warranty of               --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--  General Public License for more details.                                  --
--                                                                              --
--  You should have received a copy of the GNU General Public License         --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--  As a special exception, if other files instantiate generics from this     --
--  unit, or you link this unit with other files to produce an executable,   --
--  this unit does not by itself cause the resulting executable to be        --
--  covered by the GNU General Public License. This exception does not       --
--  however invalidate any other reasons why the executable file might be    --
--  covered by the GNU Public License.                                       --
-----

--  Handle .ini style configuration files. In those files each option is on one
--  line. The first word is the option name and the second one is the option
--  value.

package AWS.Config.Ini is

  function Program_Ini_File return String;
  -- Returns initialization filename for current server (using the
  -- executable name and adding .ini).

  procedure Read
    (Config : in out Object;
     Filename : in String);
  -- Read Filename and update the configuration object with the
  -- options read from it. Raises Ada.Text_IO.Name_Error if Filename does
  -- not exist. Raises Constraint_Error in case of wrong any parameter name
  -- or value.

end AWS.Config.Ini;
```

B.10 AWS.Config.Set

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                     --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                     --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                     --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.
-----

--   This package can be used to Set any AWS parameters.

package AWS.Config.Set is

  -----
  -- Per Server Options --
  -----

  -----
  -- Server --
  -----

  procedure Server_Name (O : in out Object; Value : in String);
  --   This is the name of the server as set by AWS.Server.Start

  procedure Server_Host (O : in out Object; Value : in String);
  --   This is the server host as set by the HTTP object declaration

  procedure Server_Port (O : in out Object; Value : in Positive);
  --   This is the server port as set by the HTTP object declaration

  procedure Hotplug_Port (O : in out Object; Value : in Positive);
  --   This is the hotplug communication port needed to register and
  --   un-register an hotplug module.

  procedure Session (O : in out Object; Value : in Boolean);
  --   Enable session handling is Value is True

  procedure Case_Sensitive_Parameters (O : in out Object; Value : in Boolean);
  --   Parameters are handled with the case if Value is True

  procedure Line_Stack_Size (O : in out Object; Value : in Positive);
  --   HTTP lines stack size

```

```

-----
-- Connection --
-----

procedure Max_Connection (O : in out Object; Value : in Positive);
-- This is the max simultaneous connections as set by the HTTP object
-- declaration.

procedure Free_Slots_Keep_Alive_Limit
  (O : in out Object; Value : in Natural);
-- The minimum number of free slots where keep-alive connections are still
-- enabled. After this limit no more keep-alive connection will be
-- accepted by the server. This parameter must be used for heavy-loaded
-- servers to make sure the server will never run out of slots. This limit
-- must be less than Max_Connection.

procedure Keep_Alive_Force_Limit (O : in out Object; Value : in Natural);
-- Define maximum number of keep alive sockets where server process it with
-- normal timeouts. If number of keep-alive sockets become more than
-- Keep_Alive_Force_Limit, server start to use shorter force timeouts.
-- If this parameter not defined in configuration or defined as 0 value
-- server use calculated value Max_Connection * 2.

procedure Accept_Queue_Size (O : in out Object; Value : in Positive);
-- This is the size of the queue for the incoming requests. Higher this
-- value will be and less "connection refused" will be reported to the
-- client.

-----
-- Data --
-----

procedure WWW_Root (O : in out Object; Value : in String);
-- This is the root directory name for the server. This variable is not
-- used internally by AWS. It is supposed to be used by the callback
-- procedures who want to retrieve physical objects (images, Web
-- pages...). The default value is the current working directory.

procedure Upload_Directory (O : in out Object; Value : in String);
-- This point to the directory where uploaded files will be stored. The
-- directory returned will end with a directory separator.

procedure Directory_Browser_Page (O : in out Object; Value : in String);
-- Filename for the directory browser template page.

-----
-- Log --
-----

procedure Log_File_Directory (O : in out Object; Value : in String);
-- This point to the directory where log files will be written. The
-- directory returned will end with a directory separator.

procedure Log_Filename_Prefix (O : in out Object; Value : in String);
-- This is the prefix to use for the log filename.

procedure Log_Split_Mode (O : in out Object; Value : in String);
-- This is split mode for the log file. Possible values are : Each_Run,
-- Daily, Monthly and None. Any other values will raise an exception.

procedure Log_Extended_Fields (O : in out Object; Value : in String);
-- Comma separated list of the extended log field names. If this parameter
-- is empty, the HTTP log would have fixed apache compatible format:
--
-- 127.0.0.1 - - [25/Apr/1998:15:37:29 +0200] "GET / HTTP/1.0" 200 1363

```



```

--
-- If the extended fields list is not empty, the log file format would have
-- user defined fields set:
--
-- #Version: 1.0
-- #Date: 2006-01-09 00:00:01
-- #Fields: date time cs-method cs-uri cs-version sc-status sc-bytes
-- 2006-01-09 00:34:23 GET /foo/bar.html HTTP/1.1 200 30
--
-- Fields in the list could be:
--
-- date          Date at which transaction completed
-- time          Time at which transaction completed
-- c-ip          Client side connected IP address
-- c-port        Client side connected port
-- s-ip          Server side connected IP address
-- s-port        Server side connected port
-- cs-method     HTTP request method
-- cs-username   Client authentication username
-- cs-version    Client supported HTTP version
-- cs-uri        Request URI
-- cs-uri-stem   Stem portion alone of URI (omitting query)
-- cs-uri-query  Query portion alone of URI
-- sc-status     Responce status code
-- sc-bytes      Length of response message body
-- cs(<header>) Any header field name sent from client to server
-- sc(<header>) Any header field name sent from server to client
-- x-<apffield> Any application defined field name

procedure Error_Log_Filename_Prefix (O : in out Object; Value : in String);
-- This is the prefix to use for the log filename.

procedure Error_Log_Split_Mode (O : in out Object; Value : in String);
-- This is split mode for the log file. Possible values are : Each_Run,
-- Daily, Monthly and None. Any other values will raise an exception.

-----
-- Status --
-----

procedure Admin_URI (O : in out Object; Value : in String);
-- This is the name of the admin server page as set by AWS.Server.Start

procedure Status_Page (O : in out Object; Value : in String);
-- Filename for the status template page

procedure Up_Image (O : in out Object; Value : in String);
-- Filename for the up arrow image used in the status page

procedure Down_Image (O : in out Object; Value : in String);
-- Filename for the down arrow image used in the status page

procedure Logo_Image (O : in out Object; Value : in String);
-- Filename for the AWS logo image used in the status page

-----
-- Timeouts --
-----

procedure Cleaner_Wait_For_Client_Timeout
(O      : in out Object;
 Value : in      Duration);
-- Number of seconds to timeout on waiting for a client request.
-- This is a timeout for regular cleaning task.

```

```

procedure Cleaner_Client_Header_Timeout
  (O      : in out Object;
   Value : in      Duration);
-- Number of seconds to timeout on waiting for client header.
-- This is a timeout for regular cleaning task.

procedure Cleaner_Client_Data_Timeout
  (O      : in out Object;
   Value : in      Duration);
-- Number of seconds to timeout on waiting for client message body.
-- This is a timeout for regular cleaning task.

procedure Cleaner_Server_Response_Timeout
  (O      : in out Object;
   Value : in      Duration);
-- Number of seconds to timeout on waiting for client to accept answer.
-- This is a timeout for regular cleaning task.

procedure Force_Wait_For_Client_Timeout
  (O      : in out Object;
   Value : in      Duration);
-- Number of seconds to timeout on waiting for a client request.
-- This is a timeout for urgent request when resources are missing.

procedure Force_Client_Header_Timeout
  (O      : in out Object;
   Value : in      Duration);
-- Number of seconds to timeout on waiting for client header.
-- This is a timeout for urgent request when resources are missing.

procedure Force_Client_Data_Timeout
  (O      : in out Object;
   Value : in      Duration);
-- Number of seconds to timeout on waiting for client message body.
-- This is a timeout for urgent request when resources are missing.

procedure Force_Server_Response_Timeout
  (O      : in out Object;
   Value : in      Duration);
-- Number of seconds to timeout on waiting for client to accept answer.
-- This is a timeout for urgent request when resources are missing.

procedure Send_Timeout (O : in out Object; Value : in Duration);
-- Number of seconds to timeout when sending chunk of data.

procedure Receive_Timeout (O : in out Object; Value : in Duration);
-- Number of seconds to timeout when receiving chunk of data.

-----
-- Security --
-----

procedure Check_URL_Validity (O : in out Object; Value : in Boolean);
-- Set the check URL validity flag. If True an URL that reference a
-- resource above the Web root will be rejected.

procedure Security (O : in out Object; Value : in Boolean);
-- Enable security (HTTPS/SSL) if Value is True

procedure Certificate (O : in out Object; Filename : in String);
-- Set the certificate to be used with the secure server

procedure Key (O : in out Object; Filename : in String);
-- Set the key to be used with the secure server

```

```

procedure Security_Mode (O : in out Object; Mode : in String);
-- Set the security mode to be used with the secure server. Only values
-- from AWS.Net.SSL.Method can be used.

procedure Exchange_Certificate (O : in out Object; Value : in Boolean);
-- Set to True to request the client to send its certificate to the server

-----
-- Per Process Options --
-----

procedure Session_Cleanup_Interval (Value : in Duration);
-- Number of seconds between each run of the cleaner task to remove
-- obsolete session data.

procedure Session_Lifetime (Value : in Duration);
-- Number of seconds to keep a session if not used. After this period the
-- session data is obsoleted and will be removed during next cleanup.

procedure Transient_Cleanup_Interval (Value : in Duration);
-- Number of seconds between each run of the cleaner task to remove
-- transient pages.

procedure Transient_Lifetime (Value : in Duration);
-- Number of seconds to keep a transient page. After this period the
-- transient page is obsoleted and will be removed during next cleanup.

procedure Parameter
(Config      : in out Object;
 Name       : in   String;
 Value      : in   String;
 Error_Context : in   String := "");
-- Set one of the AWS HTTP per server parameters. Raises Constraint_Error
-- in case of wrong parameter name or wrong parameter value.
-- Error_Context may contain additional information about the parameter.
-- This message will be added to the Constraint_Error exception.
-- One way to use Error_Context is to set it with information about
-- where this parameter come from.

procedure Parameter
(Name       : in String;
 Value      : in String;
 Error_Context : in String := "");
-- Set one of the AWS HTTP per process parameters. See description above

end AWS.Config.Set;

```

B.11 AWS.Containers.Tables

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2006                           --
--                                     AdaCore                                           --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify               --
--  it under the terms of the GNU General Public License as published by               --
--  the Free Software Foundation; either version 2 of the License, or (at             --
--  your option) any later version.                                                    --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                        --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                 --
--  General Public License for more details.                                           --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                 --
--  along with this library; if not, write to the Free Software Foundation,           --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                   --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this             --
--  unit, or you link this unit with other files to produce an executable,            --
--  this unit does not by itself cause the resulting executable to be                 --
--  covered by the GNU General Public License. This exception does not                --
--  however invalidate any other reasons why the executable file might be             --
--  covered by the GNU Public License.                                                 --
-----

with Ada.Containers.Indefinite_Hashed_Maps;
with Ada.Containers.Indefinite_Vectors;
with Ada.Containers.Vectors;
with Ada.Strings.Hash;
with Ada.Strings.Unbounded;

package AWS.Containers.Tables is

  use Ada.Strings.Unbounded;

  type Table_Type is tagged private;

  type Element (Name_Length, Value_Length : Natural) is record
    Name   : String (1 .. Name_Length);
    Value  : String (1 .. Value_Length);
  end record;
  -- Data type to store name/value pair retrieved from a Table_Type

  Null_Element : constant Element;

  type VString_Array is array (Positive range <>) of Unbounded_String;

  function Count (Table : in Table_Type) return Natural;
  -- Returns the number of item in Table

  function Name_Count (Table : in Table_Type) return Natural;
  -- Returns the number of unique key name in Table

  function Count (Table : in Table_Type; Name : in String) return Natural;
  -- Returns the number of value for Key Name in Table. It returns
  -- 0 if Key does not exist.

  function Exist (Table : in Table_Type; Name : in String) return Boolean;
  -- Returns True if Key exist in Table

```

```

function Get
  (Table : in Table_Type;
   Name   : in String;
   N      : in Positive := 1) return String;
-- Returns the Nth value associated with Key into Table. Returns
-- the empty string if key does not exist.

function Get_Name
  (Table : in Table_Type;
   N      : in Positive := 1) return String;
-- Returns the Nth Name in Table or the empty string if there is
-- no parameter with this number.

function Get_Value
  (Table : in Table_Type;
   N      : in Positive := 1) return String;
-- Returns the Nth Value in Table or the empty string if there is
-- no parameter with this number.

function Get
  (Table : in Table_Type;
   N      : in Positive) return Element;
-- Returns N'th name/value pair. Returns Null_Element if there is no
-- such item in the table.

function Get_Names
  (Table : in Table_Type;
   Sort  : in Boolean := False) return VString_Array;
-- Returns array of unique key names. If Sort is True, the returned names
-- array is sorted in alphabetical order. This is of course slightly
-- slower than returning unsorted results.

function Get_Values
  (Table : in Table_Type;
   Name  : in String) return VString_Array;
-- Returns all values for the specified parameter key name

generic
  with procedure Process (Name, Value : in String);
procedure Generic_Iterate_Names
  (Table : in Table_Type; Coupler : in String);
-- Iterates over all names in the table.
-- All Values of the same name would be given Coupler separated.

private
  -- implementation removed
end AWS.Containers.Tables;

```

B.12 AWS.Default

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2005                             --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at             --
--  your option) any later version.                                                     --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                 --
--  General Public License for more details.                                           --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                  --
--  along with this library; if not, write to the Free Software Foundation,           --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                     --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this              --
--  unit, or you link this unit with other files to produce an executable,            --
--  this unit does not by itself cause the resulting executable to be                 --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be             --
--  covered by the GNU Public License.                                                 --
-----

--  This package contains the default AWS configuration values. These values
--  are used to initialize the configuration objects. Users should not modify
--  the values here, see AWS.Config.* API.

package AWS.Default is

  pragma Pure;

  -- All times are in seconds

  One_Hour      : constant := 3_600.0;
  One_Minute    : constant :=   60.0;

  Eight_Hours   : constant := 8.0 * One_Hour;
  Three_Hours   : constant := 3.0 * One_Hour;

  Three_Minutes : constant := 3.0 * One_Minute;
  Five_Minutes  : constant := 5.0 * One_Minute;
  Ten_Minutes   : constant := 10.0 * One_Minute;

  -- Server configuration

  Server_Name      : constant String := "AWS Module";
  WWW_Root         : constant String := "./";
  Admin_URI        : constant String := "";
  Server_Port      : constant       := 8080;
  Hotplug_Port     : constant       := 8888;
  Max_Connection   : constant       := 5;
  Free_Slots_Keep_Alive_Limit : constant := 1;
  Keep_Alive_Force_Limit : constant := 0;
  Accept_Queue_Size : constant       := 64;
  Upload_Directory : constant String := "";
  Line_Stack_Size   : constant       := 16#150_000#;
  Case_Sensitive_Parameters : constant Boolean := True;

```

```

Max_Concurrent_Download      : constant      := 25;

-- Client configuration

User_Agent                   : constant String
    := "AWS (Ada Web Server) v" & Version;

-- Log values. The character '@' in the error log filename prefix is
-- replaced by the running program name.

Log_File_Directory           : constant String := "./";

Log_Split_Mode                : constant String := "NONE";
Log_Filename_Prefix           : constant String := "@";

Error_Log_Split_Mode          : constant String := "NONE";
Error_Log_Filename_Prefix     : constant String := "@_error";

-- Session

Session                       : constant Boolean := False;
Session_Cleanup_Interval      : constant Duration := Five_Minutes;
Session_Lifetime              : constant Duration := Ten_Minutes;

-- Transient pages

Transient_Cleanup_Interval     : constant Duration := Three_Minutes;
Transient_Lifetime             : constant Duration := Five_Minutes;

-- Server's timeouts

Cleaner_Wait_For_Client_Timeout : constant Duration := 80.0;
Cleaner_Client_Header_Timeout   : constant Duration := 7.0;
Cleaner_Client_Data_Timeout     : constant Duration := Eight_Hours;
Cleaner_Server_Response_Timeout : constant Duration := Eight_Hours;

Force_Wait_For_Client_Timeout   : constant Duration := 2.0;
Force_Client_Header_Timeout     : constant Duration := 2.0;
Force_Client_Data_Timeout       : constant Duration := Three_Hours;
Force_Server_Response_Timeout   : constant Duration := Three_Hours;

Send_Timeout                   : constant Duration := 40.0;
Receive_Timeout                 : constant Duration := 30.0;

-- Directory template

Directory_Browser_Page         : constant String := "aws_directory.shtml";

-- Status page

Status_Page                    : constant String := "aws_status.shtml";
Up_Image                       : constant String := "aws_up.png";
Down_Image                     : constant String := "aws_down.png";
Logo_Image                     : constant String := "aws_logo.png";

-- Security

Security                       : constant Boolean := False;
Security_Mode                   : constant String := "SSLv23";
Certificate                     : constant String := "cert.pem";
Key                             : constant String := "";
Client_Certificate              : constant String := "client.pem";
Exchange_Certificate            : constant Boolean := False;
Check_URL_Validity              : constant Boolean := True;

```

```
end AWS.Default;
```


B.13 AWS.Dispatchers

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but       --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License         --
--   along with this library; if not, write to the Free Software Foundation,   --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this     --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not       --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                         --
-----

--   This package provides a service to build Callbacks which can support
--   user's data. It is possible to build a new dispatcher by inheriting the
--   handler type and to provides the Dispatch routine.

with Ada.Finalization;

with AWS.Response;
with AWS.Status;
with AWS.Utils;

package AWS.Dispatchers is

  type Handler is abstract new Ada.Finalization.Controlled with private;

  overriding procedure Initialize (Dispatcher : in out Handler);
  overriding procedure Adjust     (Dispatcher : in out Handler);
  overriding procedure Finalize   (Dispatcher : in out Handler);
  -- Initialize/Adjust/Finalize is doing the reference counting, children
  -- should just call these routines if possible. It is possible to know if
  -- no more object are referenced by calling Ref_Counter below.

  function Dispatch
    (Dispatcher : in Handler;
     Request     : in Status.Data)
    return Response.Data is abstract;
  -- Call the appropriate inherited dispatcher

  function Ref_Counter (Dispatcher : in Handler) return Natural;
  -- Returns the reference counter for Handler. If 0 is returned then this
  -- object is not referenced anymore, it is safe to deallocate resources.

  type Handler_Class_Access is access all Handler'Class;

  procedure Free (Dispatcher : in out Handler_Class_Access);

```

```
pragma Inline (Free);  
-- Release memory associated with the dispatcher  
  
private  
-- implementation removed  
end AWS.Dispatchers;
```

B.14 AWS.Dispatchers.Callback

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2000-2006                             --
--                                     AdaCore                                             --
--                                     -----                                             --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at             --
--  your option) any later version.                                                     --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU                  --
--  General Public License for more details.                                           --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,           --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                     --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this              --
--  unit, or you link this unit with other files to produce an executable,            --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                 --
-----

-- Dispatch on a Callback procedure

with AWS.Dispatchers;
with AWS.Response;
with AWS.Status;

package AWS.Dispatchers.Callback is

  type Handler is new Dispatchers.Handler with private;
  -- This is a simple wrapper around standard callback procedure (access to
  -- function). It will be used to build dispatchers services and for the
  -- main server callback.

  function Create
    (Callback : in Response.Callback)
    return Handler;
  pragma Inline (Create);
  -- Build a dispatcher for the specified callback

  function Dispatch
    (Dispatcher : in Handler;
     Request    : in Status.Data)
    return Response.Data;

private
  -- implementation removed
end AWS.Dispatchers.Callback;

```

B.15 AWS.Exceptions

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2003-2004                             --
--                                     ACT-Europe                                           --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                 --
--  it under the terms of the GNU General Public License as published by                 --
--  the Free Software Foundation; either version 2 of the License, or (at               --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                  --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

with Ada.Exceptions;

with AWS.Log;
with AWS.Response;
with AWS.Status;

package AWS.Exceptions is

  type Data is record
    Fatal    : Boolean;
    -- If True it means that we go a fatal error. The slot will be
    -- terminated so AWS will loose one of it's simultaneous connection.
    -- This is clearly an AWS internal error that should be fixed in AWS.

    Slot     : Positive;
    -- The failing slot number

    Request  : Status.Data;
    -- The complete request information that was served when the slot has
    -- failed. This variable is set only when Fatal is False.
  end record;

  type Unexpected_Exception_Handler is access
    procedure (E      : in Ada.Exceptions.Exception_Occurrence;
              Log     : in out AWS.Log.Object;
              Error    : in Data;
              Answer   : in out Response.Data);
    -- Unexpected exception handler can be set to monitor server errors.
    -- Answer can be set with the answer to send back to the client's
    -- browser. Note that this is possible only for non fatal error
    -- (i.e. Error.Fatal is False).
    -- Log is the error log object for the failing server, it can be used
    -- to log user's information (if error log is activated for this
    -- server). Note that the server will have already logged information

```

```
-- about the problem.  
end AWS.Exceptions;
```

B.16 AWS.Headers

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2004                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--   This library is free software; you can redistribute it and/or modify    --
--   it under the terms of the GNU General Public License as published by    --
--   the Free Software Foundation; either version 2 of the License, or (at   --
--   your option) any later version.                                          --
--                                                                           --
--   This library is distributed in the hope that it will be useful, but     --
--   WITHOUT ANY WARRANTY; without even the implied warranty of              --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU       --
--   General Public License for more details.                                --
--                                                                           --
--   You should have received a copy of the GNU General Public License       --
--   along with this library; if not, write to the Free Software Foundation, --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.         --
--                                                                           --
--   As a special exception, if other files instantiate generics from this   --
--   unit, or you link this unit with other files to produce an executable,  --
--   this unit does not by itself cause the resulting executable to be      --
--   covered by the GNU General Public License. This exception does not     --
--   however invalidate any other reasons why the executable file might be   --
--   covered by the GNU Public License.                                       --
-----

with AWS.Containers.Tables;
with AWS.Net;

package AWS.Headers is

  type List is new AWS.Containers.Tables.Table_Type with private;
  -- Header container. This set handles a set of HTTP header line, each new
  -- header line is inserted at the end of the list (see AWS.Headers.Set API)
  -- and can be retrieved by the following services. Header lines are
  -- numbered from 1 to N.

  subtype VString_Array is AWS.Containers.Tables.VString_Array;

  subtype Element is AWS.Containers.Tables.Element;

  Format_Error : exception;
  -- Raised when header line format is wrong

  procedure Send_Header
    (Socket : in Net.Socket_Type'Class;
     Headers : in List);
  -- Send all header lines in Headers list to the socket

  function Get_Line
    (Headers : in List;
     N       : in Positive)
    return String;
  -- Returns the Nth header line in Headers container. The returned value is
  -- formatted as a correct header line:
  --
  --     message-header = field-name ":" [ field-value ]
  --
  -- That is the header-name followed with character ':' and the header
  -- values. If there is less than Nth header line it returns the empty

```

```
-- string. Note that this routine does returns all header line values, for
-- example it would return:
--
--     Content_Type: multipart/mixed; boundary="0123_The_Boundary_Value_"
--
-- For a file upload content type header style.

function Get_Values
  (Headers : in List;
   Name    : in String)
  return String;
-- Returns all values for the specified header field Name in a
-- comma-separated string. This format is conformant to [RFC 2616 - 4.2]
-- (see last paragraph).

function Length (Headers : in AWS.Headers.List) return Natural;
-- Returns the length (in bytes) of the header, including the ending
-- empty line.

-- See AWS.Containers.Tables for inherited routines.

private
  -- implementation removed
end AWS.Headers;
```

B.17 AWS.Headers.Values

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2002-2003                       --
--                               ACT-Europe                                    --
--                               -----                                    --
--   This library is free software; you can redistribute it and/or modify    --
--   it under the terms of the GNU General Public License as published by    --
--   the Free Software Foundation; either version 2 of the License, or (at  --
--   your option) any later version.                                         --
--                                                                           --
--   This library is distributed in the hope that it will be useful, but     --
--   WITHOUT ANY WARRANTY; without even the implied warranty of             --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU      --
--   General Public License for more details.                                --
--                                                                           --
--   You should have received a copy of the GNU General Public License       --
--   along with this library; if not, write to the Free Software Foundation, --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.         --
--                                                                           --
--   As a special exception, if other files instantiate generics from this   --
--   unit, or you link this unit with other files to produce an executable,  --
--   this unit does not by itself cause the resulting executable to be      --
--   covered by the GNU General Public License. This exception does not     --
--   however invalidate any other reasons why the executable file might be   --
--   covered by the GNU Public License.                                     --
-----

with Ada.Strings.Unbounded;

package AWS.Headers.Values is

  use Ada.Strings.Unbounded;

  -- Data represent a token from an header line. There is two kinds of
  -- token, either named or un-named.
  --
  --   Content-Type: xyz boundary="uvt"
  --
  -- Here xyz is an un-named value and uvt a named value the name is
  -- boundary.

  type Data (Named_Value : Boolean := True) is record
    Value : Unbounded_String;
    case Named_Value is
      when True =>
        Name : Unbounded_String;
      when False =>
        null;
    end case;
  end record;

  type Set is array (Positive range <>) of Data;

  -----
  -- Parse --
  -----

  generic

    with procedure Value
      (Item : in String;

```



```

        Quit : in out Boolean);
-- Called for every un-named value read from the header value

with procedure Named_Value
(Name : in String;
 Value : in String;
 Quit : in out Boolean);
-- Called for every named value read from the header value

procedure Parse (Header_Value : in String);
-- Look for un-named values and named ones (Name="Value" pairs) in the
-- header line, and call appropriate routines when found. Quit is set to
-- False before calling Value or Named_Value, the parsing can be stopped
-- by setting Quit to True.

-----
-- Split / Index --
-----

function Split (Header_Value : in String) return Set;
-- Returns a Set with each named and un-named values splited from Data.

function Index
(Set : in Values.Set;
 Name : in String;
 Case_Sensitive : in Boolean := True)
return Natural;
-- Returns index for Name in the set or 0 if Name not found.
-- If Case_Sensitive is false the find is case_insensitive.

-----
-- Other search routines --
-----

function Search
(Header_Value : in String;
 Name : in String;
 Case_Sensitive : in Boolean := True)
return String;
-- Returns Value for Name in Header_Value or the empty string if Name not
-- found. If Case_Sensitive is False the search is case insensitive.

function Get_Unnamed_Value
(Header_Value : in String;
 N : in Positive := 1)
return String;
-- Returns N-th un-named value from Header_Value.

function Unnamed_Value_Exists
(Header_Value : in String;
 Value : in String;
 Case_Sensitive : in Boolean := True)
return Boolean;
-- Returns True if the unnamed value specified has been found in
-- Header_Value.

end AWS.Headers.Values;
```

B.18 AWS.Jabber

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2002-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this     --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                       --
-----

with Ada.Strings.Unbounded;

with AWS.Containers.Key_Value;
with AWS.Net;
with AWS.Utils;

package AWS.Jabber is

  type Server is limited private;
  -- This is the Jabber server connection. This object is initialized by
  -- Connect below and is used with all services.

  Default_Port : constant := 5222;
  -- Standard Jabber Server default port is 5222. The SSL based connection
  -- port is 5223 but this is not supported by this API.

  Server_Error : exception;
  -- Raised by any routine below when an server or protocol error occurs. A
  -- message is attached to the exception, this correspond to the <error>
  -- XML protocol tag if present.

  procedure Connect
    (Server : in out Jabber.Server;
     Host    : in String;
     User    : in String;
     Password : in String;
     Port    : in Positive := Default_Port);
  -- Connect to a Jabber server Host:Port using User/Password account. It
  -- returns the Server object which can be used with services below.

  procedure Close (Server : in out Jabber.Server);
  -- Close the connection with the Jabber server.

  procedure Send_Message

```

```
(Server : in Jabber.Server;
 JID    : in String;
 Subject : in String;
 Content : in String);
-- Send a message to user JID (Jabber ID) via the specified Server. The
-- message is composed of Subject and a body (Content).

type Presence_Status
  is (Offline, Available, Chat, Away, Extended_Away, Do_Not_Disturb);

procedure Check_Presence
  (Server : in Jabber.Server;
   JID    : in String;
   Status : out Presence_Status);
-- Returns the presence status for JID.

private
  -- implementation removed
end AWS.Jabber;
```

B.19 AWS.LDAP.Client

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2003-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                                                                              --
--  This library is distributed in the hope that it will be useful, but      --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--  General Public License for more details.                                  --
--                                                                              --
--  You should have received a copy of the GNU General Public License         --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--  As a special exception, if other files instantiate generics from this     --
--  unit, or you link this unit with other files to produce an executable,   --
--  this unit does not by itself cause the resulting executable to be        --
--  covered by the GNU General Public License. This exception does not       --
--  however invalidate any other reasons why the executable file might be    --
--  covered by the GNU Public License.                                         --
-----

-- Provides an API to read information from an LDAP server. This API does not
-- cover modifying, adding or deleting information into the server. It is a
-- thick binding, see AWS.LDAP.Thin for a thin binding.
--
-- This API has been tested on Windows and Linux (OpenLDAP).

with Ada.Exceptions;
with Ada.Strings.Unbounded;

with AWS.LDAP.Thin;

package AWS.LDAP.Client is

  use Ada.Strings.Unbounded;

  Default_Port : constant Positive := Positive (Thin.LDAP_PORT);

  subtype Directory    is Thin.LDAP_Type;
  -- An LDAP directory. This object must be initialized with Init and Bind
  -- and terminated with Unbind.

  subtype LDAP_Message is Thin.LDAP_Message;
  -- An LDAP message or set of messages. There is a set of iterators to
  -- access all messages returned by the search procedure.

  subtype BER_Element is Thin.BerElement;
  -- An iterator structure. Initialized and used to iterate through all the
  -- attributes for a specific message.

  Null_Directory    : constant Directory    := Thin.Null_LDAP_Type;
  Null_LDAP_Message : constant LDAP_Message := Thin.Null_LDAP_Message;

  type Scope_Type is

```

```

    (LDAP_Scope_Default, LDAP_Scope_Base,
     LDAP_Scope_One_Level, LDAP_Scope_Subtree);
-- LDAP scope for the search

type String_Set is array (Positive range <>) of Unbounded_String;
-- A set of strings, this is used to map C array of strings (a char **)
-- from the thin binding.

Null_Set : constant String_Set;

function Get_Error
  (E : in Ada.Exceptions.Exception_Occurrence)
  return Thin.Return_Code;
-- Returns the error code in the LDAP_Error exception occurrence E. Returns
-- Think.LDAP_SUCCESS if no error code has been found.

-----
-- Attributes --
-----

subtype Attribute_Set is String_Set;
-- Used to represent the set of attributes to retrieve from the LDAP server

function Attributes
  (S1, S2, S3, S4, S5, S6, S7, S8, S9, S10 : in String := "")
  return Attribute_Set;
-- Returns a String_Set object containing only none empty values. Values
-- for S1 through S10 must be set in the order of the parameters. This is
-- an helper routine to help building an array of unbounded string from a
-- set of string.

function uid (Val : in String := "") return String;
-- Returns the uid attribute, if Val is specified "<Val>" is
-- added after the attribute name.

function givenName (Val : in String := "") return String;
-- Returns the given name (firstname) attribute. if Val is specified
-- "<Val>" is added after the attribute name.

function cn (Val : in String := "") return String;
function commonName (Val : in String := "") return String renames cn;
-- Returns the common Name attribute, if Val is specified "<Val>" is
-- added after the attribute name.

function sn (Val : in String := "") return String;
function surname (Val : in String := "") return String renames sn;
-- Returns the surname attribute, if Val is specified "<Val>" is
-- added after the attribute name.

function telephoneNumber (Val : in String := "") return String;
-- Returns the phone number. if Val is specified "<Val>" is
-- added after the attribute name. Val must use the international notation
-- according to CCITT E.123.

function mail (Val : in String := "") return String;
-- Returns the mail attribute. if Val is specified "<Val>" is added after
-- the attribute name.

function l (Val : in String := "") return String;
function localityName (Val : in String := "") return String renames l;
-- Returns the locality attribute, if Val is specified "<Val>" is
-- added after the attribute name.

function o (Val : in String := "") return String;
function organizationName (Val : in String := "") return String renames o;

```

```

-- Returns the organization attribute, if Val is specified "<Val>" is
-- added after the attribute name.

function ou (Val : in String := "") return String;
function organizationalUnitName (Val : in String := "") return String
renames ou;
-- Returns the organizational unit attribute, if Val is specified "<Val>"
-- is added after the attribute name.

function st (Val : in String := "") return String;
function stateOrProvinceName (Val : in String := "") return String
renames st;
-- Returns the state name attribute, if Val is specified "<Val>" is
-- added after the attribute name.

function c (Val : in String := "") return String;
function countryName (Val : in String) return String renames c;
-- Returns country code attribute, if Val is specified "<Val>" is
-- added after the attribute name. Val must be a two-letter ISO 3166
-- country code.

function dc (Val : in String := "") return String;
function domainComponent (Val : in String := "") return String renames dc;
-- Returns a domain component attribute, if Val is specified "<Val>" is
-- added after the attribute name.

function Cat
(S1, S2, S3, S4, S5, S6, S7, S8, S9, S10 : in String := "")
return String;
-- Returns a string object containing only none empty values. Values for
-- S1 through S10 must be set in the order of the parameters. All values
-- are catenated and separated with a coma. This is an helper routine to
-- help building a filter objects or base distinguished name.

-----
-- Initialize --
-----

function Init
(Host : in String;
Port : in Positive := Default_Port)
return Directory;
-- Must be called first, to initialize the LDAP communication with the
-- server. Returns Null_Directory in case of error.

procedure Bind
(Dir      : in out Directory;
Login     : in      String;
Password  : in      String);
-- Bind to the server by providing a login and password

procedure Unbind (Dir : in out Directory);
-- Must be called to release resources associated with the Directory. Does
-- nothing if Dir is Null_Directory.

function Is_Open (Dir : in Directory) return Boolean;
-- Returns True if the directory has correctly been initialized and binded
-- with the server.

-----
-- Search --
-----

function Search
(Dir      : in Directory;

```

```

    Base      : in String;
    Filter     : in String;
    Scope      : in Scope_Type      := LDAP_Scope_Default;
    Attrs      : in Attribute_Set := Null_Set;
    Attrs_Only : in Boolean         := False)
    return LDAP_Message;
-- Do a search on the LDAP server. Base is the name of the database.
-- Filter can be used to retrieve a specific set of entries. Attrs specify
-- the set of attributes to retrieve. If Attrs_Only is set to True only
-- the types are returned. Raises LDAP_Error in case of problem.

-----
-- Iterators --
-----

function First_Entry
  (Dir   : in Directory;
   Chain : in LDAP_Message)
  return LDAP_Message;
-- Returns the first entry (or Node) for the search result (Chain).

function Next_Entry
  (Dir       : in Directory;
   Entries   : in LDAP_Message)
  return LDAP_Message;
-- Returns next entry (or Node) for Entries.

function Count_Entries
  (Dir   : in Directory;
   Chain : in LDAP_Message)
  return Natural;
-- Returns the number of entries in the search result (Chain).

procedure Free (Chain : in LDAP_Message);
-- Release memory associated with the search result Chain.

generic
  with procedure Action
    (Node : in LDAP_Message;
     Quit : in out Boolean);
procedure For_Every_Entry (Dir : in Directory; Chain : in LDAP_Message);
-- This iterator call Action for each entry (Node) found in the LDAP result
-- set as returned by the search procedure. Quit can be set to True to
-- stop iteration; its initial value is False.

function First_Attribute
  (Dir   : in Directory;
   Node  : in LDAP_Message;
   BER   : not null access BER_Element)
  return String;
-- Returns the first attribute for the entry. It initialize an iterator
-- (the BER structure). The BER structure must be released after used by
-- using the Free routine below.

function Next_Attribute
  (Dir   : in Directory;
   Node  : in LDAP_Message;
   BER   : in BER_Element)
  return String;
-- Returns next attribute for iterator BER. First_Attribute must have been
-- called to initialize this iterator.

procedure Free (BER : in BER_Element);
-- Releases memory associated with the BER structure which has been
-- allocated by the First_Attribute routine.

```

```

generic
  with procedure Action
    (Attribute : in String;
     Quit      : in out Boolean);
  procedure For_Every_Attribute
    (Dir : in Directory;
     Node : in LDAP_Message);
  -- This iterator call action for each attribute found in the LDAP Entries
  -- Node as returned by First_Entry or Next_Entry. Quit can be set to True
  -- to stop iteration; its initial value is False.

  -----
  -- Accessors --
  -----

  function Get_DN
    (Dir : in Directory;
     Node : in LDAP_Message)
    return String;
  -- Returns the distinguished name for the given entry Node.

  function DN2UFN (DN : in String) return String;
  -- Returns a distinguished name converted to a user-friendly format

  function Get_Values
    (Dir : in Directory;
     Node : in LDAP_Message;
     Target : in String)
    return String_Set;
  -- Returns the list of values of a given attribute (Target) found in entry
  -- Node.

  function Explode_DN
    (DN : in String;
     No_Types : in Boolean := True)
    return String_Set;
  -- Breaks up an entry name into its component parts. If No_Types is set to
  -- True the types information ("cn=") won't be included.

private
  -- implementation removed
end AWS.LDAP.Client;

```


B.20 AWS.Log

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but       --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this     --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not       --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                         --
-----

--   This package handle the logging facility for AWS. The log file is named
--   '<programe>-Y-M-D.log' and is written by default in the directory where
--   the server is launched, see configuration file.
--
--   Note that this package is used internally by AWS to log server requests
--   but it can also be used by users to handle application's log.
--
--   This package is thread safe.

with Ada.Finalization;
with Ada.Text_IO;
with Ada.Strings.Unbounded;

with AWS.Containers.String_Vectors;
with AWS.Headers;
with AWS.Status;
with AWS.Response;
with AWS.Messages;
with AWS.Utils;

with Strings_Maps;

package AWS.Log is

  type Object is limited private;
  -- A log object. It must be activated by calling Start below

  type Split_Mode is (None, Each_Run, Daily, Monthly);
  -- It specifies when to create a new log file.
  -- None      : all log info gets accumulated into the same file.
  -- Each_Run  : a new log file is created each time the server is started.
  -- Daily     : a new log file is created each day.
  -- Monthly   : a new log file is created each month.

```

```

type Fields_Table is private;
-- Type to keep record for Extended Log File Format

Empty_Fields_Table : constant Fields_Table;

Not_Specified : constant String;

procedure Start
(Log          : in out Object;
 Split        : in      Split_Mode := None;
 File_Directory : in      String    := Not_Specified;
 Filename_Prefix : in      String    := Not_Specified;
 Auto_Flush    : in      Boolean    := False);
-- Activate server's activity logging. Split indicate the way the log file
-- should be created. Log_File_Prefix is the log filename prefix. If it is
-- not specified the default prefix is the program name.
-- Set Auto_Flush to True if you want every write to the log to be flushed
-- (not buffered). Auto_Flush should be set to True only for logs with few
-- entries per second as the flush has a performance penalty.

procedure Register_Field (Log : in out Object; Id : in String);
-- Register field to be written into extended log format

procedure Set_Field
(Log : in Object; Data : in out Fields_Table; Id, Value : in String);
-- Set field value into the extended log record. Data could be used only
-- in one task and with one log file. Different tasks could write own Data
-- using the Write routine with Fields_Table parameter type.

procedure Set_Header_Fields
(Log      : in      Object;
 Data     : in out Fields_Table;
 Prefix   : in      String;
 Header   : in      AWS.Headers.List);
-- Set header fields into extended log record.
-- Name of the header fields would be <Prefix><Header_Name>.
-- Prefix should be "cs" - Client to Server or "sc" - Server to Client.

procedure Write (Log : in out Object; Data : in out Fields_Table);
-- Write extended format record to log file and prepare record for the next
-- data. It is not allowed to use same Fields_Table with different extended
-- logs.

procedure Write
(Log          : in out Object;
 Connect_Stat : in      Status.Data;
 Answer       : in      Response.Data);
-- Write log info if activated (i.e. Start routine above has been called)

procedure Write
(Log          : in out Object;
 Connect_Stat : in      Status.Data;
 Status_Code  : in      Messages.Status_Code;
 Content_Length : in      Response.Content_Length_Type);
-- Write log info if activated (i.e. Start routine above has been called).
-- This version separated the Content_Length from Status.Data, this is
-- required for example in the case of a user defined stream content. See
-- AWS.Resources.Stream.

procedure Write
(Log          : in out Object;
 Connect_Stat : in      Status.Data;
 Data         : in      String);
-- Write user's log info if activated. (i.e. Start routine above has been
-- called).

```

```
procedure Write (Log : in out Object; Data : in String);  
-- Write Data into the log file. This Data is unstructured, only a time  
-- tag prefix is prepended to Data. This routine is designed to be used  
-- for user's info in error log file.  
  
procedure Flush (Log : in out Object);  
-- Flush the data to the Log file, for be able to see last logged  
-- messages.  
  
procedure Stop (Log : in out Object);  
-- Stop logging activity  
  
function Is_Active (Log : in Object) return Boolean;  
-- Returns True if Log is activated  
  
function Filename (Log : in Object) return String;  
-- Returns current log filename or the empty string if the log is not  
-- activated.  
  
function Mode (Log : in Object) return Split_Mode;  
-- Returns the split mode. None will be returned if log is not activated.  
  
private  
-- implementation removed  
end AWS.Log;
```

B.21 AWS.Messages

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2004                             --
--                                     ACT-Europe                                           --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                 --
--  it under the terms of the GNU General Public License as published by                 --
--  the Free Software Foundation; either version 2 of the License, or (at               --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                           --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                   --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,             --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,              --
--  this unit does not by itself cause the resulting executable to be                   --
--  covered by the GNU General Public License. This exception does not                  --
--  however invalidate any other reasons why the executable file might be               --
--  covered by the GNU Public License.                                                    --
-----

with Ada.Calendar;

package AWS.Messages is

  use Ada;

  -----
  -- HTTP tokens --
  -----

  HTTP-Token : constant String := "HTTP/";
  Get-Token  : constant String := "GET ";
  Put-Token   : constant String := "PUT ";
  Head-Token  : constant String := "HEAD ";
  Post-Token  : constant String := "POST ";

  -----
  -- HTTP header tokens --
  -----

  -- General header tokens RFC 2616
  Cache_Control-Token : constant String := "Cache-Control";
  Connection-Token    : constant String := "Connection";
  Date-Token          : constant String := "Date";
  Pragma-Token        : constant String := "Pragma";
  Trailer-Token       : constant String := "Trailer";
  Transfer-Encoding-Token : constant String := "Transfer-Encoding";
  Upgrade-Token       : constant String := "Upgrade";
  Via-Token           : constant String := "Via";
  Warning-Token       : constant String := "Warning";

  -- Request header tokens RFC 2616
  Accept-Token : constant String := "Accept";
  Accept_Charset-Token : constant String := "Accept-Charset";

```

```

Accept-Encoding-Token      : constant String := "Accept-Encoding";
Accept-Language-Token     : constant String := "Accept-Language";
Authorization-Token       : constant String := "Authorization";
Expect-Token              : constant String := "Expect";
From-Token                : constant String := "From";
Host-Token                : constant String := "Host";
If_Match-Token            : constant String := "If-Match";
If_Modified_Since-Token   : constant String := "If-Modified-Since";
If_None_Match-Token       : constant String := "If-None-Match";
If_Range-Token            : constant String := "If-Range";
If_Unmodified_Since-Token : constant String := "If-Unmodified-Since";
Max_Forwards-Token        : constant String := "Max-Forwards";
Proxy_Authorization-Token  : constant String := "Proxy-Authorization";
Range-Token               : constant String := "Range";
Referer-Token             : constant String := "Referer";
TE-Token                  : constant String := "TE";
User_Agent-Token          : constant String := "User-Agent";

-- Response header tokens RFC 2616
Accept_Ranges-Token       : constant String := "Accept-Ranges";
Age-Token                  : constant String := "Age";
ETag-Token                 : constant String := "ETag";
Location-Token             : constant String := "Location";
Proxy_Authenticate-Token  : constant String := "Proxy-Authenticate";
Retry_After-Token         : constant String := "Retry-After";
Server-Token              : constant String := "Server";
Vary-Token                 : constant String := "Vary";
WWW_Authenticate-Token    : constant String := "WWW-Authenticate";

-- Entity header tokens RFC 2616
Allow-Token                : constant String := "Allow";
Content-Encoding-Token     : constant String := "Content-Encoding";
Content-Language-Token     : constant String := "Content-Language";
Content_Length-Token       : constant String := "Content-Length";
Content_Location-Token     : constant String := "Content-Location";
Content_MD5-Token          : constant String := "Content-MD5";
Content_Range-Token        : constant String := "Content-Range";
Content_Type-Token         : constant String := "Content-Type";
Expires-Token              : constant String := "Expires";
Last_Modified-Token        : constant String := "Last-Modified";

-- Other tokens
Proxy_Connection-Token     : constant String := "Proxy-Connection";
Content_Disposition-Token  : constant String := "Content-Disposition";
Cookie-Token               : constant String := "Cookie";
Set_Cookie-Token           : constant String := "Set-Cookie";
SOAPAction-Token           : constant String := "SOAPAction";
Content_Id-Token           : constant String := "Content-ID";
Content_Transfer-Encoding-Token : constant String
    := "Content-Transfer-Encoding";

-----
-- Status Code --
-----

type Status_Code is
    (S100, S101,
     -- 1xx : Informational - Request received, continuing process

     S200, S201, S202, S203, S204, S205, S206,
     -- 2xx : Success - The action was successfully received, understood and
     -- accepted

     S300, S301, S302, S303, S304, S305, S307,
     -- 3xx : Redirection - Further action must be taken in order to

```

```

-- complete the request

S400, S401, S402, S403, S404, S405, S406, S407, S408, S409,
S410, S411, S412, S413, S414, S415, S416, S417,
-- 4xx : Client Error - The request contains bad syntax or cannot be
-- fulfilled

S500, S501, S502, S503, S504, S505
-- 5xx : Server Error - The server failed to fulfill an apparently
-- valid request
);

function Image (S : in Status_Code) return String;
-- Returns Status_Code image. This value does not contain the leading S.

function Reason_Phrase (S : in Status_Code) return String;
-- Returns the reason phrase for the status code S, see [RFC 2616 - 6.1.1]

-----
-- Content encoding --
-----

type Content_Encoding is (Identity, GZip, Deflate);
-- Encoding mode for the response, Identity means that no encoding is
-- done, Gzip/Deflate to select the Gzip or Deflate encoding algorithm.

-----
-- Cache_Control --
-----

type Cache_Option is new String;

-- Cache_Option is a string and any specific option can be specified. We
-- define three options:
--
-- Unspecified   : No cache option will used.
-- No_Cache      : Ask browser and proxy to not cache data (no-cache,
--                max-age, and s-maxage are specified).
-- No_Store      : Ask browser and proxy to not store any data. This can be
--                used to protect sensitive data.
-- Prevent_Cache : Equivalent to No_Store + No_Cache

Unspecified      : constant Cache_Option;
No_Cache         : constant Cache_Option;
No_Store         : constant Cache_Option;
Prevent_Cache    : constant Cache_Option;

-----
-- HTTP message constructors --
-----

function Accept_Type (Mode : in String) return String;
pragma Inline (Accept_Type);

function Accept_Language (Mode : in String) return String;
pragma Inline (Accept_Language);

function Authorization (Mode, Password : in String) return String;
pragma Inline (Authorization);

function Connection (Mode : in String) return String;
pragma Inline (Connection);

function Content_Length (Size : in Natural) return String;
pragma Inline (Content_Length);

```

```

function Cookie (Value : in String) return String;
pragma Inline (Cookie);

function Content_Type
  (Format   : in String;
   Boundary : in String := "")
  return String;
pragma Inline (Content_Type);

function Cache_Control (Option : in Cache_Option) return String;
pragma Inline (Cache_Control);

function Content_Disposition
  (Format   : in String;
   Name     : in String;
   Filename : in String)
  return String;
pragma Inline (Content_Disposition);
-- Note that this is not part of HTTP/1.1 standard, it is there because
-- there is a lot of implementation around using it. This header is used
-- in multipart data.

function Host (Name : in String) return String;
pragma Inline (Host);

function Last_Modified (Date : in Calendar.Time) return String;
pragma Inline (Last_Modified);

function Location (URL : in String) return String;
pragma Inline (Location);

function Proxy_Authorization (Mode, Password : in String) return String;
pragma Inline (Proxy_Authorization);

function Proxy_Connection (Mode : in String) return String;
pragma Inline (Proxy_Connection);

function SOAPAction (URI : in String) return String;
pragma Inline (SOAPAction);

function Status_Line (Code : in Status_Code) return String;
pragma Inline (Status_Line);

function Transfer-Encoding (Encoding : in String) return String;
pragma Inline (Transfer-Encoding);

function User_Agent (Name : in String) return String;
pragma Inline (User_Agent);

function WWW_Authenticate (Realm : in String) return String;
pragma Inline (WWW_Authenticate);
-- Basic authentication request.

function WWW_Authenticate
  (Realm : in String;
   Nonce : in String;
   Stale : in Boolean)
  return String;
pragma Inline (WWW_Authenticate);
-- Digest authentication request.

-----
-- helper functions --
-----

```

```
function Match (Str, Pattern : in String) return Boolean;
pragma Inline (Match);
-- Returns True if Pattern matches the begining of Str. The test is not
-- case sensitive.

function Does_Not_Match (Str, Pattern : in String) return Boolean;
pragma Inline (Does_Not_Match);
-- Returns True if Pattern does not matches the begining of Str. The test
-- is not case sensitive.

function To_HTTP_Date (Time : in Calendar.Time) return String;
-- Returns an Ada time as a string using the HTTP normalized format.
-- Format is RFC 822, updated by RFC 1123.

function To_Time (HTTP_Date : in String) return Calendar.Time;
-- Returns an Ada time from an HTTP one. This is To_HTTP_Date opposite
-- function.

private
  -- implementation removed
end AWS.Messages;
```


B.22 AWS.MIME

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2005                             --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                     --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                 --
--  General Public License for more details.                                           --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                     --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                  --
-----

package AWS.MIME is

  -- Some content type constants. All of them will be defined into this
  -- package and associated with the right extensions. It is possible to
  -- add new MIME types with the routines below or by placing a file named
  -- aws.mime into the startup directory.
  --
  -- A MIME type is written in two parts: type/format

  -----
  -- Text --
  -----

  Text_CSS          : constant String := "text/css";
  Text_HTML          : constant String := "text/html";
  Text_Plain         : constant String := "text/plain";
  Text_XML           : constant String := "text/xml";
  Text_X_SGML        : constant String := "text/x-sgml";

  -----
  -- Image --
  -----

  Image_Gif          : constant String := "image/gif";
  Image_Jpeg          : constant String := "image/jpeg";
  Image_Png           : constant String := "image/png";
  Image_Tiff          : constant String := "image/tiff";
  Image_X_Portable_Anymap : constant String := "image/x-portable-anymap";
  Image_X_Portable_Bitmap : constant String := "image/x-portable-bitmap";
  Image_X_Portable_Graymap : constant String := "image/x-portable-graymap";
  Image_X_Portable_Pixmap : constant String := "image/x-portable-pixmap";
  Image_X_RGB         : constant String := "image/x-rgb";
  Image_X_Xbitmap     : constant String := "image/x-xbitmap";
  Image_X_Xpixmap     : constant String := "image/x-xpixmap";

```

```

Image_X_Xwindowdump      : constant String := "image/x-xwindowdump";

-----
-- Application --
-----

Application_Postscript    : constant String := "application/postscript";
Application_Pdf            : constant String := "application/pdf";
Application_Zip            : constant String := "application/zip";
Application_Octet_Stream  : constant String := "application/octet-stream";
Application_Form_Data      : constant String
    := "application/x-www-form-urlencoded";
Application_Mac_Binhex40   : constant String := "application/mac-binhex40";
Application_Msword         : constant String := "application/msword";
Application_Powerpoint     : constant String := "application/powerpoint";
Application_Rtf            : constant String := "application/rtf";
Application_X_Compress     : constant String := "application/x-compress";
Application_X_GTar         : constant String := "application/x-gtar";
Application_X_GZip         : constant String := "application/x-gzip";
Application_X_Latex        : constant String := "application/x-latex";
Application_X_Sh           : constant String := "application/x-sh";
Application_X_Shar         : constant String := "application/x-shar";
Application_X_Tar          : constant String := "application/x-tar";
Application_X_Tcl          : constant String := "application/x-tcl";
Application_X_Tex          : constant String := "application/x-tex";
Application_X_Texinfo      : constant String := "application/x-texinfo";
Application_X_Troff        : constant String := "application/x-troff";
Application_X_Troff_Man    : constant String := "application/x-troff-man";

-----
-- Audio --
-----

Audio_Basic               : constant String := "audio/basic";
Audio_Mpeg                : constant String := "audio/mpeg";
Audio_X_Wav               : constant String := "audio/x-wav";
Audio_X_Pn_Realaudio      : constant String := "audio/x-pn-realaudio";
Audio_X_Pn_Realaudio_Plugin : constant String
    := "audio/x-pn-realaudio-plugin";
Audio_X_Realaudio         : constant String := "audio/x-realaudio";

-----
-- Video --
-----

Video_Mpeg                : constant String := "video/mpeg";
Video_Quicktime           : constant String := "video/quicktime";
Video_X_Msvideo           : constant String := "video/x-msvideo";

-----
-- Multipart --
-----

Multipart_Form_Data       : constant String := "multipart/form-data";
Multipart_Byteranges      : constant String := "multipart/byteranges";
Multipart_Related         : constant String := "multipart/related";
Multipart_X_Mixed_Replace : constant String
    := "multipart/x-mixed-replace";

-----
-- Setting --
-----

procedure Add_Extension (Ext : in String; MIME_Type : in String);
-- Add extension Ext (file extension without the dot, e.g. "txt") to the

```

```

-- set of MIME type extension handled by this API. Ext will be mapped to
-- the MIME_Type string.

procedure Add_Regexp (Filename : in String; MIME_Type : in String);
-- Add a specific rule to the MIME type table. Filename is a regular
-- expression and will be mapped to the MIME_Type string.

-----
-- MIME Type --
-----

function Content_Type
  (Filename : in String;
   Default  : in String := Application_Octet_Stream) return String;
-- Returns the MIME Content Type based on filename's extension or if not
-- found the MIME Content type where Filename matches one of the specific
-- rules set by Add_Regexp (see below).
-- Returns Default if the file type is unknown (i.e. no extension and
-- no regular expression match filename).

function Extension (Content_Type : in String) return String;
-- Returns the best guess of the extension to use for the Content Type.
-- Note that extensions added indirectly by Add_Regexp are not searched.

function Is_Text (MIME_Type : in String) return Boolean;
-- Returns True if the MIME_Type is a text data

function Is_Audio (MIME_Type : in String) return Boolean;
-- Returns True if the MIME_Type is an audio data

function Is_Image (MIME_Type : in String) return Boolean;
-- Returns True if the MIME_Type is an image data

function Is_Video (MIME_Type : in String) return Boolean;
-- Returns True if the MIME_Type is a video data

function Is_Application (MIME_Type : in String) return Boolean;
-- Returns True if the MIME_Type is an application data

end AWS.MIME;
```

B.23 AWS.Net

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2006                           --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                  --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

--  This is two implementations for this spec. One for standard sockets and
--  one for SSL socket. Note that the SSL implementation does support standard
--  socket too, this is controlled with the Security boolean on routine
--  below. The corresponding implementation will be selected at build time.

with Ada.Exceptions;
with Ada.Finalization;
with Ada.Streams;

with AWS.Utils;

package AWS.Net is

  use Ada;
  use Ada.Streams;
  use Ada.Exceptions;

  Socket_Error : exception;
  --  Raised by all routines below, a message will indicate the nature of
  --  the error.

  type Socket_Type is abstract tagged private;
  type Socket_Access is access all Socket_Type'Class;

  type Socket_Set is array (Positive range <>) of Socket_Access;

  subtype FD_Type is Integer;
  --  Represents an external socket file descriptor

  type FD_Set (Size : Natural) is abstract tagged private;
  --  Abstract type for waiting of network events on group of sockets FD

  type Event_Type is (Error, Input, Output);
  --  Error - socket is in error state.

```

```

-- Input - socket ready for read.
-- Output - socket available for write.

type Event_Set is array (Event_Type) of Boolean;
-- Type for get result of events waiting

subtype Wait_Event_Type is Event_Type range Input .. Output;
type Wait_Event_Set is array (Wait_Event_Type) of Boolean;
-- Type for set events to wait, note that Error event would be waited
-- anyway.

Forever : constant Duration;
-- The longest delay possible on the implementation

-----
-- Initialize --
-----

function Socket (Security : in Boolean) return Socket_Type'Class;
-- Create an uninitialized socket

function Socket (Security : in Boolean) return Socket_Access;
-- Create a dynamically allocated uninitialized socket

procedure Bind
  (Socket : in out Socket_Type;
   Port   : in   Natural;
   Host   : in   String := "") is abstract;
-- Create the server socket and bind it on the given port.
-- Using 0 for the port will tell the OS to allocate a non-privileged
-- free port. The port can be later retrieved using Get_Port on the
-- bound socket.

procedure Listen
  (Socket : in Socket_Type; Queue_Size : in Positive := 5) is abstract;
-- Set the queue size of the socket

procedure Accept_Socket
  (Socket : in Socket_Type'Class; New_Socket : in out Socket_Type)
  is abstract;
-- Accept a connection on a socket. If it raises Socket_Error, all
-- resources used by new_Socket have been released.
-- There is not need to call Free or Shutdown.

type Socket_Constructor is access
  function (Security : in Boolean) return Socket_Type'Class;

procedure Connect
  (Socket : in out Socket_Type;
   Host   : in   String;
   Port   : in   Positive;
   Wait   : in   Boolean := True) is abstract;
-- Connect a socket on a given host/port. If Wait is True Connect will wait
-- for the connection to be established for timeout seconds, specified by
-- Set_Timeout routine. If Wait is False Connect will return immediately,
-- not waiting for the connection to be established. It is possible to wait
-- for the Connection completion by calling Wait routine with Output set to
-- True in Events parameter.

procedure Socket_Pair (S1, S2 : out Socket_Type);
-- Create 2 sockets and connect them together

procedure Shutdown (Socket : in Socket_Type) is abstract;
-- Shutdown both side of the socket and close it. Does not raise
-- Socket_Error if the socket is not connected.

```

```

procedure Free (Socket : in out Socket_Access);
-- Release memory associated with the socket

-----
-- IO --
-----

procedure Send
  (Socket : in Socket_Type'Class; Data : in Stream_Element_Array);
-- Send Data chunk to the socket

procedure Send
  (Socket : in      Socket_Type;
   Data   : in      Stream_Element_Array;
   Last   : out Stream_Element_Offset) is abstract;
-- Try to place data to Socket's output buffer.
-- If all data cannot be placed to the socket output buffer, Last will
-- be lower than Data'Last, if no data has been placed into the output
-- buffer, Last is set to be out of Data'Range, either Data'First - 1
-- or Stream_Element_Offset'Last is used.

procedure Receive
  (Socket : in      Socket_Type;
   Data   : out Stream_Element_Array;
   Last   : out Stream_Element_Offset) is abstract;
-- Read a chunk of data from the socket and set appropriate Last value.
-- This call always returns some data and will wait for incoming data only
-- if necessary.

function Receive
  (Socket : in Socket_Type'Class;
   Max    : in Stream_Element_Count := 4096) return Stream_Element_Array;
-- Read a chunk of data from the socket and returns it. This call always
-- returns some data and will wait for incoming data only if necessary.

function Pending (Socket : in Socket_Type) return Stream_Element_Count
is abstract;
-- Returns the number of bytes which are available inside socket
-- for immediate read.

-----
-- Others --
-----

function Get_FD (Socket : in Socket_Type) return FD_Type is abstract;
-- Returns the file descriptor associated with the socket

function Peer_Addr (Socket : in Socket_Type) return String is abstract;
-- Returns the peer name/address

function Peer_Port (Socket : in Socket_Type) return Positive is abstract;
-- Returns the port of the peer socket

function Get_Addr (Socket : in Socket_Type) return String is abstract;
-- Returns the name/address of the socket

function Get_Port (Socket : in Socket_Type) return Positive is abstract;
-- Returns the port of the socket

function Host_Name return String;
-- Returns the running host name

procedure Set_Send_Buffer_Size
  (Socket : in Socket_Type; Size : in Natural) is abstract;

```

```

-- Set the internal socket send buffer size.
-- Do not confuse with buffers for the AWS.Net.Buffered operations.

procedure Set_Receive_Buffer_Size
  (Socket : in Socket_Type; Size : in Natural) is abstract;
-- Set the internal socket receive buffer size.
-- Do not confuse with buffers for the AWS.Net.Buffered operations.

function Get_Send_Buffer_Size (Socket : in Socket_Type) return Natural
  is abstract;
-- Returns the internal socket send buffer size.
-- Do not confuse with buffers for the AWS.Net.Buffered operations.

function Get_Receive_Buffer_Size (Socket : in Socket_Type) return Natural
  is abstract;
-- Returns the internal socket receive buffer size.
-- Do not confuse with buffers for the AWS.Net.Buffered operations.

procedure Set_Blocking_Mode
  (Socket : in out Socket_Type; Blocking : in Boolean);
pragma Obsolescent ("Use Set_Timeout instead");
-- Set the blocking mode for the socket

procedure Set_Timeout (Socket : in out Socket_Type; Timeout : in Duration);
pragma Inline (Set_Timeout);
-- Sets the timeout for the socket read/write operations

procedure Set_No_Delay
  (Socket : in Socket_Type; Value : in Boolean := True);
-- Set/clear TCP_NODELAY option on socket

function Wait
  (Socket : in Socket_Type'Class;
   Events : in Wait_Event_Set) return Event_Set;
-- Waiting for Input/Output/Error events.
-- Waiting time is defined by Set_Timeout.
-- Empty event set in result mean that timeout occurred.

function Check
  (Socket : in Socket_Type'Class;
   Events : in Wait_Event_Set) return Event_Set;
-- Check for Input/Output/Error events availability.
-- No wait for socket timeout.

function Poll
  (Socket : in Socket_Type'Class;
   Events : in Wait_Event_Set;
   Timeout : in Duration) return Event_Set;
-- Wait events on socket descriptor for specified Timeout

function Errno (Socket : in Socket_Type) return Integer is abstract;
-- Returns and clears error state in socket

function Is_Timeout (E : in Exception_Occurrence) return Boolean;
-- Returns True if the message associated with the Exception_Occurrence for
-- a Socket_Error is a timeout.

function To_FD_Set
  (Socket : in Socket_Type;
   Events : in Wait_Event_Set;
   Size : in Positive := 1) return FD_Set'Class;
-- Create appropriate socket FD set and put Socket fd there.

-----
-- Socket FD sets --

```

```

-----

type FD_Set_Access is access all FD_Set'Class;

procedure Add
  (FD_Set : in out FD_Set_Access;
   FD      : in      FD_Type;
   Event   : in      Wait_Event_Set);
-- Add FD to the end of FD_Set

procedure Free (FD_Set : in out FD_Set_Access);
pragma Inline (Free);
-- Deallocate the socket FD set

procedure Add
  (FD_Set : in out Net.FD_Set;
   FD      : in      FD_Type;
   Event   : in      Wait_Event_Set) is abstract;
-- Add FD to the end of FD_Set

procedure Set_Mode
  (FD_Set : in out Net.FD_Set;
   Index   : in      Positive;
   Mode    : in      Wait_Event_Set) is abstract;
-- Sets the kind of network events to wait for

function Copy
  (FD_Set : access Net.FD_Set;
   Size    : in      Natural) return FD_Set_Access is abstract;
-- Allocates and copy the given FD_Set with different size

procedure Remove
  (FD_Set : in out Net.FD_Set; Index : in Positive) is abstract;
-- Removes socket FD from Index position.
-- Last socket FD in FD_Set is placed at position Index.

function Length (FD_Set : in Net.FD_Set) return Natural is abstract;
-- Returns number of socket FD elements in FD_Set

procedure Wait
  (FD_Set : in out Net.FD_Set;
   Timeout : in      Duration;
   Count   : out Natural) is abstract;
-- Wait for network events on the sockets FD set. Count value is the
-- number of socket FDs with non empty event set.

procedure Next
  (FD_Set : in Net.FD_Set; Index : in out Positive) is abstract;
-- Looking for an active (for which an event has been detected by routine
-- Wait above) socket FD starting from Index and return Index of the found
-- active socket FD. Use functions Status to retrieve the kind of network
-- events for this socket.

function Status
  (FD_Set : in Net.FD_Set;
   Index   : in Positive) return Event_Set is abstract;
-- Returns events for the socket FD at position Index

private
  -- implementation removed
end AWS.Net;

```


B.24 AWS.Net.Buffered

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2002-2005                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                         --
-----

-- All routines below are buffered both ways (input and output) for better
-- performances.

package AWS.Net.Buffered is

  -----
  -- Output --
  -----

  procedure Put (Socket : in Socket_Type'Class; Item : in String);
  -- Write Item into Socket's buffer. Send the buffer to the socket if full

  procedure Put_Line (Socket : in Socket_Type'Class; Item : in String);
  -- Write Item & CRLF into Socket's buffer. Send the buffer to the socket
  -- if full.

  procedure New_Line (Socket : in Socket_Type'Class);
  pragma Inline (New_Line);
  -- Write CRLF into Socket's buffer. Send the buffer to the socket if full.

  procedure Write
    (Socket : in Socket_Type'Class;
     Item   : in Stream_Element_Array);
  -- Write Item into Socket's buffer. Send the buffer to the socket if full.

  procedure Flush (Socket : in Socket_Type'Class);
  -- Send the buffer to the socket

  -----
  -- Input --
  -----

  procedure Read
    (Socket : in      Socket_Type'Class;

```

```

    Data : out Stream_Element_Array);
pragma Inline (Read);
-- Returns Data array read from the socket

function Read
(Socket : in Socket_Type'Class;
 Max : in Stream_Element_Count := 4096)
return Ada.Streams.Stream_Element_Array;
pragma Inline (Read);
-- Returns an array of bytes read from the socket

procedure Read
(Socket : in Socket_Type'Class;
 Data : out Stream_Element_Array;
 Last : out Stream_Element_Offset);
-- Read any available data from buffered socket.
-- Wait if no data available.
-- Same semantic with Net.Receive procedure.

function Get_Line (Socket : in Socket_Type'Class) return String;
-- Returns a line read from Socket. A line is a set of character
-- terminated by CRLF.

function Get_Char (Socket : in Socket_Type'Class) return Character;
pragma Inline (Get_Char);
-- Returns a single character read from socket

function Peek_Char (Socket : in Socket_Type'Class) return Character;
pragma Inline (Peek_Char);
-- Returns next character that will be read from Socket. It does not
-- actually consume the character, this character will be returned by
-- the next read operation on the socket.

procedure Read_Buffer
(Socket : in Socket_Type'Class;
 Data : out Stream_Element_Array;
 Last : out Stream_Element_Offset);
-- Returns data read from the internal socket's read buffer. No data are
-- read from the socket. This can be useful when switching to non buffered
-- mode.

-----
-- Control --
-----

procedure Shutdown (Socket : in Socket_Type'Class);
-- Shutdown and close the socket. Release all memory and resources
-- associated with it.

end AWS.Net.Buffered;
```

B.25 AWS.Net.Log

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2004-2006                           --
--                                     AdaCore                                           --
--                                                                                       --
--   This library is free software; you can redistribute it and/or modify               --
--   it under the terms of the GNU General Public License as published by               --
--   the Free Software Foundation; either version 2 of the License, or (at            --
--   your option) any later version.                                                    --
--                                                                                       --
--   This library is distributed in the hope that it will be useful, but                --
--   WITHOUT ANY WARRANTY; without even the implied warranty of                        --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                 --
--   General Public License for more details.                                           --
--                                                                                       --
--   You should have received a copy of the GNU General Public License                  --
--   along with this library; if not, write to the Free Software Foundation,          --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                   --
--                                                                                       --
--   As a special exception, if other files instantiate generics from this             --
--   unit, or you link this unit with other files to produce an executable,            --
--   this unit does not by itself cause the resulting executable to be                 --
--   covered by the GNU General Public License. This exception does not                --
--   however invalidate any other reasons why the executable file might be             --
--   covered by the GNU Public License.                                                 --
-----

--   This package handles the Net logging facility for AWS.
--
--   AWS calls the Write procedure which in turn calls the callback routine
--   provided by the user when starting the logging. This feature can help
--   greatly to debug an application.
--
--   This package is thread safe. There will never be two simultaneous calls
--   to the callback routine.

package AWS.Net.Log is

  type Data_Direction is (Sent, Received);
  --   The direction of the data, sent or received to/from the socket

  type Event_Type is (Connect, Accept_Socket, Shutdown);

  type Write_Callback is access procedure
    (Direction : in Data_Direction;
     Socket     : in Socket_Type'Class;
     Data       : in Stream_Element_Array;
     Last       : in Stream_Element_Offset);
  --   The callback procedure which is called for each incoming/outgoing data

  type Event_Callback is access procedure
    (Action : in Event_Type; Socket : in Socket_Type'Class);
  --   The callback procedure which is called for every socket creation,
  --   connect and accept.

  type Error_Callback is access procedure
    (Socket : in Socket_Type'Class; Message : in String);
  --   The callback procedure which is called for every socket error

  procedure Start
    (Write : in Write_Callback;

```

```

    Event : in Event_Callback := null;
    Error : in Error_Callback := null);
--   Activate the logging

function Is_Active return Boolean;
pragma Inline (Is_Active);
--   Returns True if Log is activated and False otherwise

function Is_Write_Active return Boolean;
pragma Inline (Is_Write_Active);
--   Returns True if Write Log is activated and False otherwise

function Is_Event_Active return Boolean;
pragma Inline (Is_Event_Active);
--   Returns True if Event Log is activated and False otherwise

procedure Write
  (Direction : in Data_Direction;
   Socket     : in Socket_Type'Class;
   Data       : in Stream_Element_Array;
   Last       : in Stream_Element_Offset);
--   Write sent/received data indirectly through the callback routine,
--   if activated (i.e. Start routine above has been called). Otherwise this
--   call does nothing.

procedure Event (Action : in Event_Type; Socket : in Socket_Type'Class);
--   Call Event callback if activated (i.e. Start routine above has been
--   called). Otherwise this call does nothing.

procedure Error (Socket : in Socket_Type'Class; Message : in String);
--   Call Error callback if activated (i.e. Start routine above has been
--   called). Otherwise this call does nothing.

procedure Stop;
--   Stop logging activity

end AWS.Net.Log;
```

B.26 AWS.Net.Log.Callbacks

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2004                           --
--                               ACT-Europe                                   --
--                               -----                                   --
--   This library is free software; you can redistribute it and/or modify   --
--   it under the terms of the GNU General Public License as published by   --
--   the Free Software Foundation; either version 2 of the License, or (at --
--   your option) any later version.                                         --
--                                                                           --
--   This library is distributed in the hope that it will be useful, but    --
--   WITHOUT ANY WARRANTY; without even the implied warranty of            --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU      --
--   General Public License for more details.                               --
--                                                                           --
--   You should have received a copy of the GNU General Public License      --
--   along with this library; if not, write to the Free Software Foundation, --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.        --
--                                                                           --
--   As a special exception, if other files instantiate generics from this  --
--   unit, or you link this unit with other files to produce an executable, --
--   this unit does not by itself cause the resulting executable to be     --
--   covered by the GNU General Public License. This exception does not    --
--   however invalidate any other reasons why the executable file might be  --
--   covered by the GNU Public License.                                     --
-----

-- Some ready to use write procedures

package AWS.Net.Log.Callbacks is

  procedure Initialize
    (Filename : in String;
     Callback : in Write_Callback);
  -- Initialize the logging, must be called before using the callbacks below

  procedure Finalize;
  -- Stop logging, close log file

  procedure Text
    (Direction : in Data_Direction;
     Socket     : in Socket_Type'Class;
     Data       : in Stream_Element_Array;
     Last       : in Stream_Element_Offset);
  -- A text output, each chunk is output with an header and footer:
  --   Data sent/received to/from socket <FD> (<size>/<buffer size>)
  --   <data>
  --   Total data sent: <nnn> received: <nnn>

  procedure Binary
    (Direction : in Data_Direction;
     Socket     : in Socket_Type'Class;
     Data       : in Stream_Element_Array;
     Last       : in Stream_Element_Offset);
  -- A binary output, each chunk is output with an header and footer. The
  -- data itself is written using a format close to the Emacs hexl-mode:
  --   Data sent/received to/from socket <FD> (<size>/<buffer size>)
  --   HH HH HH HH HH HH HH HH HH HH HH HH HH  az.rt.mpl..q
  --   Total data sent: <nnn> received: <nnn>
  --
  -- HH is the hex character number, if the character is not printable a dot

```

```
-- is written.  
end AWS.Net.Log.Callbacks;
```

B.27 AWS.Net.SSL

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2002-2006                             --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                     --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU                   --
--  General Public License for more details.                                           --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                     --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                  --
-----

--  This is the SSL based implementation of the Net package. The implementation
--  should depend only on AWS.Net.Std and the SSL library. It is important to
--  not call directly a socket binding here to ease porting.

with AWS.Net.Std;
with SSL.Thin;

package AWS.Net.SSL is

  type Socket_Type is new Net.Std.Socket_Type with private;

  Is_Supported : constant Boolean;
  -- True if SSL supported in the current runtime

  -----
  -- Initialize --
  -----

  procedure Accept_Socket
    (Socket      : in      Net.Socket_Type'Class;
     New_Socket  : in out Socket_Type);
  -- Accept a connection on a socket

  procedure Connect
    (Socket : in out Socket_Type;
     Host   : in      String;
     Port   : in      Positive;
     Wait   : in      Boolean := True);
  -- Connect a socket on a given host/port. If Wait is True Connect will wait
  -- for the connection to be established for timeout seconds, specified by
  -- Set_Timeout routine. If Wait is False Connect will return immediately,
  -- not waiting for the connection to be established and it does not make the
  -- SSL handshake. It is possible to wait for the Connection completion by
  -- calling Wait routine with Output set to True in Events parameter.

```

```

procedure Socket_Pair (S1, S2 : out Socket_Type);
-- Create 2 sockets and connect them together.

procedure Shutdown (Socket : in Socket_Type);
-- Shutdown both side of the socket and close it

procedure Free (Socket : in out Socket_Type);
-- Release memory associated with the socket object

-----
-- IO --
-----

procedure Send
  (Socket : in      Socket_Type;
   Data   : in      Stream_Element_Array;
   Last   : out     Stream_Element_Offset);

procedure Receive
  (Socket : in      Socket_Type;
   Data   : out     Stream_Element_Array;
   Last   : out     Stream_Element_Offset);
pragma Inline (Receive);

function Pending (Socket : in Socket_Type) return Stream_Element_Count;
-- Returns the number of bytes which are available inside socket
-- for immediate read.

-----
-- Initialization --
-----

type Method is
  (SSLv2,  SSLv2_Server,  SSLv2_Client,
   SSLv23, SSLv23_Server, SSLv23_Client,
   TLSv1,  TLSv1_Server, TLSv1_Client,
   SSLv3,  SSLv3_Server, SSLv3_Client);

type Config is private;

Null_Config : constant Config;

procedure Initialize
  (Config           : in out SSL.Config;
   Certificate_Filename : in      String;
   Security_Mode     : in      Method      := SSLv23;
   Key_Filename       : in      String      := "";
   Exchange_Certificate : in      Boolean   := False);
-- Initialize the SSL layer into Config. Certificate_Filename must point
-- to a valid certificate. Security mode can be used to change the
-- security method used by AWS. Key_Filename must be specified if the key
-- is not in the same file as the certificate. The Config object can be
-- associated with all secure sockets sharing the same options. If
-- Exchange_Certificate is True the client will send it's certificate to
-- the server, if False only the server will send its certificate.

procedure Release (Config : in out SSL.Config);
-- Release memory associated with the Config object

procedure Set_Config
  (Socket : in out Socket_Type;
   Config : in      SSL.Config);
-- Set the SSL configuration object for the secure socket

```



```
function Secure_Client
  (Socket : in Net.Socket_Type'Class;
   Config : in SSL.Config := Null_Config) return Socket_Type;
-- Make client side SSL connection from plain socket.
-- SSL handshake does not performed. SSL handshake would be made
-- automatically on first Read/Write, or explicitly by the Do_Handshake
-- call. Do not free or close source socket after this call.

function Secure_Server
  (Socket : in Net.Socket_Type'Class;
   Config : in SSL.Config := Null_Config) return Socket_Type;
-- Make server side SSL connection from plain socket.
-- SSL handshake does not performed. SSL handshake would be made
-- automatically on first Read/Write, or explicitly by the Do_Handshake
-- call. Do not free or close source socket after this call.

procedure Do_Handshake (Socket : in out Socket_Type);
-- Wait for a SSL/TLS handshake to take place. You need to call this
-- routine if you have converted a standard socket to secure one and need
-- to get the peer certificate.

private
  -- implementation removed
end AWS.Net.SSL;
```

B.28 AWS.Net.SSL.Certificate

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2003                           --
--                               ACT-Europe                                    --
--                               --                                             --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                               --                                             --
--  This library is distributed in the hope that it will be useful, but      --
--  WITHOUT ANY WARRANTY; without even the implied warranty of               --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--  General Public License for more details.                                  --
--                               --                                             --
--  You should have received a copy of the GNU General Public License        --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                               --                                             --
--  As a special exception, if other files instantiate generics from this    --
--  unit, or you link this unit with other files to produce an executable,   --
--  this unit does not by itself cause the resulting executable to be        --
--  covered by the GNU General Public License. This exception does not      --
--  however invalidate any other reasons why the executable file might be    --
--  covered by the GNU Public License.                                       --
-----

with Ada.Strings.Unbounded;

package AWS.Net.SSL.Certificate is

  use Ada.Strings.Unbounded;

  type Object is private;

  Undefined : constant Object;

  function Get (Socket : in Socket_Type) return Object;
  -- Returns the certificate used by the SSL

  function Subject (Certificate : in Object) return String;
  -- Returns the certificate's subject

  function Issuer (Certificate : in Object) return String;
  -- Returns the certificate's issuer

private
  -- implementation removed
end AWS.Net.SSL.Certificate;

```

B.29 AWS.Parameters

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2002                         --
--                                     ACT-Europe                                       --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify             --
--  it under the terms of the GNU General Public License as published by             --
--  the Free Software Foundation; either version 2 of the License, or (at          --
--  your option) any later version.                                                  --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but             --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                     --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU             --
--  General Public License for more details.                                         --
--                                                                                       --
--  You should have received a copy of the GNU General Public License               --
--  along with this library; if not, write to the Free Software Foundation,        --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                 --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this          --
--  unit, or you link this unit with other files to produce an executable,         --
--  this unit does not by itself cause the resulting executable to be              --
--  covered by the GNU General Public License. This exception does not             --
--  however invalidate any other reasons why the executable file might be         --
--  covered by the GNU Public License.                                              --
-----

with Ada.Strings.Unbounded;

with AWS.Containers.Tables;

package AWS.Parameters is

  type List is new AWS.Containers.Tables.Table_Type with private;

  subtype VString_Array is AWS.Containers.Tables.VString_Array;

  function URI_Format (Parameter_List : in List) return String;
  -- Returns the list of parameters in the URI format. This can be added
  -- after the ressource to form the complete URI. The format is:
  -- "?name1=value1&name2=value2..."

  -- See AWS.Containers.Tables for inherited routines.

private
  -- implementation removed
end AWS.Parameters;

```

B.30 AWS.POP

```

-----
--                      Ada Web Server                      --
--                      P O P - Post Office Protocol          --
--                                                                --
--                      Copyright (C) 2003-2006                --
--                      AdaCore                                --
--                                                                --
--  This library is free software; you can redistribute it and/or modify --
--  it under the terms of the GNU General Public License as published by --
--  the Free Software Foundation; either version 2 of the License, or (at --
--  your option) any later version.                               --
--                                                                --
--  This library is distributed in the hope that it will be useful, but --
--  WITHOUT ANY WARRANTY; without even the implied warranty of --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU --
--  General Public License for more details.                     --
--                                                                --
--  You should have received a copy of the GNU General Public License --
--  along with this library; if not, write to the Free Software Foundation, --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA. --
--                                                                --
--  As a special exception, if other files instantiate generics from this --
--  unit, or you link this unit with other files to produce an executable, --
--  this unit does not by itself cause the resulting executable to be --
--  covered by the GNU General Public License. This exception does not --
--  however invalidate any other reasons why the executable file might be --
--  covered by the GNU Public License.                            --
-----

with Ada.Finalization;
with Ada.Strings.Unbounded;

with AWS.Headers;
with AWS.Net.Std;
with AWS.Resources.Streams;
with AWS.Utils;

package AWS.POP is

  use Ada.Strings.Unbounded;

  POP_Error : exception;
  -- Raised by all routines when an error has been detected

  -----
  -- Mailbox --
  -----

  Default_POP_Port : constant := 110;

  type Mailbox is private;

  type Authenticate_Mode is (Clear_Text, APOP);

  function Initialize
    (Server_Name : in String;
     User        : in String;
     Password    : in String;
     Authenticate : in Authenticate_Mode := Clear_Text;
     Port        : in Positive          := Default_POP_Port)
    return Mailbox;
  -- Connect on the given Port to Server_Name and open User's Mailbox. This

```

```

-- mailbox object will be used to retrieve messages.

procedure Close (Mailbox : in POP.Mailbox);
-- Close mailbox

function User_Name (Mailbox : in POP.Mailbox) return String;
-- Returns User's name for this mailbox

function Message_Count (Mailbox : in POP.Mailbox) return Natural;
-- Returns the number of messages in the user's mailbox

function Size (Mailbox : in POP.Mailbox) return Natural;
-- Returns the total size in bytes of the user's mailbox

-----
-- Message --
-----

type Message is tagged private;

function Get
  (Mailbox : in POP.Mailbox;
   N       : in Positive;
   Remove  : in Boolean      := False)
  return Message;
-- Retrieve Nth message from the mailbox, let the message on the mailbox
-- if Remove is False.

procedure Delete
  (Mailbox : in POP.Mailbox;
   N       : in Positive);
-- Delete message number N from the mailbox

function Get_Header
  (Mailbox : in POP.Mailbox;
   N       : in Positive)
  return Message;
-- Retrieve headers for the Nth message from the mailbox, let the message
-- on the mailbox. This is useful to build a quick summary of the mailbox.

generic
  with procedure Action
    (Message : in POP.Message;
     Index   : in Positive;
     Quit    : in out Boolean);
procedure For_Every_Message
  (Mailbox : in POP.Mailbox;
   Remove  : in Boolean := False);
-- Calls Action for each message read on the mailbox, delete the message
-- from the mailbox if Remove is True. Set Quit to True to stop the
-- iterator. Index is the mailbox's message index.

generic
  with procedure Action
    (Message : in POP.Message;
     Index   : in Positive;
     Quit    : in out Boolean);
procedure For_Every_Message_Header (Mailbox : in POP.Mailbox);
-- Calls Action for each message read on the mailbox. Only the headers are
-- read from the mailbox. Set Quit to True to stop the iterator. Index is
-- the mailbox's message index.

function Size (Message : in POP.Message) return Natural;
-- Returns the message size in bytes

```

```

function Content (Message : in POP.Message) return Unbounded_String;
-- Returns message's content as an Unbounded_String. Each line are
-- separated by CR+LF characters.

function From (Message : in POP.Message) return String;
-- Returns From header value

function To (Message : in POP.Message; N : Natural := 0) return String;
-- Returns the To header value. If N = 0 returns all recipients separated
-- by a coma otherwise it returns the Nth To recipient.

function To_Count (Message : in POP.Message) return Natural;
-- Returns the number of To recipient for Message. returns 0 if there is
-- no To for this message.

function CC (Message : in POP.Message; N : Natural := 0) return String;
-- Retrurns the CC header value. If N = 0 returns all recipients separated
-- by a coma otherwise it returns the Nth CC recipient.

function CC_Count (Message : in POP.Message) return Natural;
-- Returns the number of CC recipient for Message. Returns 0 if there is
-- no CC for this message.

function Subject (Message : in POP.Message) return String;
-- Returns Subject header value

function Date (Message : in POP.Message) return String;
-- Returns Date header value

function Header
  (Message : in POP.Message;
   Header  : in String)
  return String;
-- Returns header value for header named Header, returns the empty string
-- if such header does not exist.

-----
-- Attachment --
-----

type Attachment is private;

function Attachment_Count (Message : in POP.Message) return Natural;
-- Returns the number of Attachments into Message

function Get
  (Message : in POP.Message'Class;
   Index   : in Positive)
  return Attachment;
-- Returns the Nth Attachment for Message, Raises Constraint_Error if
-- there is not such attachment.

generic
  with procedure Action
    (Attachment : in POP.Attachment;
     Index      : in Positive;
     Quit       : in out Boolean);
procedure For_Every_Attachment (Message : in POP.Message);
-- Calls action for every Attachment in Message. Stop iterator if Quit is
-- set to True, Quit is set to False by default.

function Content
  (Attachment : in POP.Attachment)
  return AWS.Resources.Streams.Stream_Access;
-- Returns Attachment's content as a memory stream. Note that the stream

```

```
-- has already been decoded. Most attachments are MIME Base64 encoded.

function Content (Attachment : in POP.Attachment) return Unbounded_String;
-- Returns Attachment's content as an Unbounded_String. This routine must
-- only be used for non file attachments. Raises Constraint_Error if
-- called for a file attachment.

function Filename (Attachment : in POP.Attachment) return String;
-- Returns the Attachment filename or the empty string if it is not a file
-- but an embedded message.

function Is_File (Attachment : in POP.Attachment) return Boolean;
-- Returns True if Attachment is a file

procedure Write (Attachment : in POP.Attachment; Directory : in String);
-- Writes Attachment's file content into Directory. This must only be used
-- for a file attachment.

private
  -- implementation removed
end AWS.POP;
```

B.31 AWS.Resources

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2002-2005                         --
--                                     AdaCore                                           --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify             --
--  it under the terms of the GNU General Public License as published by             --
--  the Free Software Foundation; either version 2 of the License, or (at          --
--  your option) any later version.                                                  --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but             --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                     --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU              --
--  General Public License for more details.                                         --
--                                                                                       --
--  You should have received a copy of the GNU General Public License               --
--  along with this library; if not, write to the Free Software Foundation,        --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                 --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this          --
--  unit, or you link this unit with other files to produce an executable,         --
--  this unit does not by itself cause the resulting executable to be              --
--  covered by the GNU General Public License. This exception does not             --
--  however invalidate any other reasons why the executable file might be          --
--  covered by the GNU Public License.                                              --
-----

with Ada.Calendar;
with Ada.Streams;
with Ada.Unchecked_Deallocation;

package AWS.Resources is

  use Ada.Streams;

  Resource_Error : exception;

  type File_Type is limited private;

  type File_Instance is (None, Plain, GZip, Both);
  -- None : No instance of this file present.
  -- Plain : A non-compressed version of this file exists.
  -- GZip : A gzip encoded version of this file exists.
  -- Both : Both versions of this file exists.

  subtype Content_Length_Type is Stream_Element_Offset;

  Undefined_Length : constant Content_Length_Type;
  -- Undefined length could be used when we do not know the message stream
  -- length at the start of transfer. The end of message could be determined
  -- by the chunked transfer-encoding in the HTTP/1.1, or by the closing
  -- connection in the HTTP/1.0.

  procedure Open
    (File : out File_Type;
     Name : in String;
     Form : in String := "");
  -- Open file in mode In_File. Only reading from the file is supported.
  -- This procedure open the in-memory (embedded) file if present, otherwise
  -- the file on disk is opened. Note that if Name file is not found, it
  -- checks for Name & ".gz" and unzipped the file content in this case.

```



```

procedure Open
  (File : out File_Type;
   Name : in String;
   Form : in String := "";
   GZip : in out Boolean);
-- Open file in mode In_File. Only reading from the file is supported.
-- This procedure open the in-memory (embedded) file if present, otherwise
-- the file on disk is opened. If GZip parameter is False this call is
-- equivalent to the Open routine above. If GZip is True this routine will
-- first check for the compressed version of the resource (Name & ".gz"),
-- if found GZip output value will remain True. If GZip value is True and
-- the compressed version of the resource does not exist it looks for
-- non-compressed version and set GZip value to False.

procedure Reset (Resource : in out File_Type);
-- Reset the file, reading will restart at the beginning

procedure Set_Index
  (Resource : in out File_Type;
   To : in Stream_Element_Offset);
-- Set the position in the stream, next Read will start at the position
-- whose index is To. If To is outside the content the index is set to
-- Last + 1 to ensure that next End_Of_File will return True.

procedure Close (Resource : in out File_Type);
-- Close the file

procedure Read
  (Resource : in out File_Type;
   Buffer : out Stream_Element_Array;
   Last : out Stream_Element_Offset);
-- Returns a set of bytes from the file

procedure Get_Line
  (Resource : in out File_Type;
   Buffer : out String;
   Last : out Natural);
-- Returns a line from the file. A line is a set of character terminated
-- by ASCII.LF (UNIX style EOF) or ASCII.CR+ASCII.LF (DOS style EOF).

function End_Of_File (Resource : in File_Type) return Boolean;
-- Returns true if there is no more data to read.

function LF_Terminated (Resource : in File_Type) return Boolean;
-- Returns True if last line returned by Get_Line was terminated with a LF
-- or CR+LF on DOS based systems.

function Size (Resource : in File_Type) return Content_Length_Type;
-- Returns the size (in bytes) of the resource. If the size of the
-- resource is not defined, the routine Size returns Undefined_Length
-- value.

function Exist (Name : in String) return File_Instance;
-- Return GZip if only file Name & ".gz" exists.
-- Return Plain if only file Name exists.
-- Return Both if both file Name and Name & ".gz" exists.
-- Return None if files neither Name nor Name & ".gz" exist.

function Is_Regular_File (Name : in String) return Boolean;
-- Returns True if Filename is a regular file and is readable. Checks
-- first for in memory file then for disk file.

function File_Size (Name : in String) return Stream_Element_Offset;
-- Returns Filename's size in bytes. Checks first for in memory file

```

```
-- then for disk file.

function File_Timestamp (Name : in String) return Ada.Calendar.Time;
-- Get the time for last modification to a file in UTC/GMT. Checks first
-- for in memory file then for disk file.

private
-- implementation removed
end AWS.Resources;
```

B.32 AWS.Resources.Embedded

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2002-2004                             --
--                                     ACT-Europe                                           --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                 --
--  it under the terms of the GNU General Public License as published by                 --
--  the Free Software Foundation; either version 2 of the License, or (at               --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                  --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                   --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

with Ada.Streams;

with AWS.Resources.Streams.Memory;

package AWS.Resources.Embedded is

  use Ada;

  subtype Buffer_Access is Streams.Memory.Buffer_Access;

  procedure Open
    (File : out File_Type;
     Name : in String;
     Form : in String := "";
     GZip : in out Boolean);
  -- Open resource from registered data

  procedure Create
    (File : out File_Type;
     Buffer : in Buffer_Access);
  -- Create the resource directly from memory data

  function Exist (Name : in String) return File_Instance;
  -- Return GZip if only file Name & ".gz" exists.
  -- Return Plain if only file Name exists.
  -- Return Both if both file Name and Name & ".gz" exists.
  -- Return None if files neither Name nor Name & ".gz" exist.

  function Is_Regular_File (Name : in String) return Boolean;
  pragma Inline (Is_Regular_File);
  -- Returns True if file named Name has been registered (i.e. it is an
  -- in-memory file).

  function File_Size

```

```
(Name : in String)
  return Ada.Streams.Stream_Element_Offset;

function File_Timestamp (Name : in String) return Ada.Calendar.Time;

procedure Register
  (Name      : in String;
   Content   : in Buffer_Access;
   File_Time : in Calendar.Time);
-- Register a new file named Name into the embedded resources. The file
-- content is pointed to by Content, the File_Time must be the last
-- modification time stamp for the file. If Name ends with ".gz" the
-- embedded resource registered as compressed. If a file is already
-- registered for this name, Content replace the previous one.

end AWS.Resources.Embedded;
```

B.33 AWS.Resources.Files

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2002-2003                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                                                                              --
--  This library is distributed in the hope that it will be useful, but       --
--  WITHOUT ANY WARRANTY; without even the implied warranty of               --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--  General Public License for more details.                                  --
--                                                                              --
--  You should have received a copy of the GNU General Public License        --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--  As a special exception, if other files instantiate generics from this     --
--  unit, or you link this unit with other files to produce an executable,   --
--  this unit does not by itself cause the resulting executable to be        --
--  covered by the GNU General Public License. This exception does not       --
--  however invalidate any other reasons why the executable file might be    --
--  covered by the GNU Public License.                                        --
-----

```

package AWS.Resources.Files is

 procedure Open

```

    (File : out File_Type;
     Name : in String;
     Form : in String := "";
     GZip : in out Boolean);

```

 procedure Open

```

    (File : out File_Type;
     Name : in String;
     Form : in String := "");

```

 function Exist (Name : in String) return File_Instance;

```

-- Return GZip if only file Name & ".gz" exists.
-- Return Plain if only file Name exists.
-- Return Both if both file Name and Name & ".gz" exists.
-- Return None if files neither Name nor Name & ".gz" exist.

```

 function Is_Regular_File (Name : in String) return Boolean;

 function File_Size

```

    (Name : in String)
    return Ada.Streams.Stream_Element_Offset;

```

 function File_Timestamp (Name : in String) return Ada.Calendar.Time;

end AWS.Resources.Files;

B.34 AWS.Resources.Streams

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2002-2005                             --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                     --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                 --
--  General Public License for more details.                                           --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,           --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                     --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this              --
--  unit, or you link this unit with other files to produce an executable,            --
--  this unit does not by itself cause the resulting executable to be                 --
--  covered by the GNU General Public License. This exception does not                --
--  however invalidate any other reasons why the executable file might be             --
--  covered by the GNU Public License.                                                 --
-----

package AWS.Resources.Streams is

  use Ada;

  type Stream_Type is abstract tagged limited private;

  type Stream_Access is access all Stream_Type'Class;

  function End_Of_File (Resource : in Stream_Type) return Boolean is abstract;

  procedure Read
    (Resource : in out Stream_Type;
     Buffer    : out Stream_Element_Array;
     Last     : out Stream_Element_Offset) is abstract;

  procedure Reset (Resource : in out Stream_Type) is abstract;

  procedure Set_Index
    (Resource : in out Stream_Type;
     To       : in Stream_Element_Offset) is abstract;
  -- Set the position in the stream, next Read will start at the position
  -- whose index is To. If To is outside the content the index is set to
  -- Last + 1 to ensure that next End_Of_File will return True.

  procedure Close (Resource : in out Stream_Type) is abstract;

  function Size (Resource : in Stream_Type) return Stream_Element_Offset;
  -- This default implementation returns Undefined_Length. If the derived
  -- stream implementation knows about the size (in bytes) of the stream
  -- this routine should be redefined.

  function Name (Resource : in Stream_Type) return String;
  -- This default implementation returns the empty string. It is must be
  -- overwritten by file based stream to provide the proper filename

```

```
-- associated with the stream.

procedure Create
  (Resource : out File_Type;
   Stream   : in  Stream_Access);
pragma Inline (Create);
-- Create a resource file from user defined stream

private
  -- implementation removed
end AWS.Resources.Streams;
```

B.35 AWS.Resources.Streams.Disk

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2003-2005                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but       --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License         --
--   along with this library; if not, write to the Free Software Foundation,   --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this     --
--   unit, or you link this unit with other files to produce an executable,    --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not       --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                       --
-----

-- An ready-to-use implementation of the stream API where the stream content
-- is read from an on-disk file.

with Ada.Streams.Stream_IO;

package AWS.Resources.Streams.Disk is

  type Stream_Type is new Streams.Stream_Type with private;

  procedure Open
    (File : out Stream_Type;
     Name : in String;
     Form : in String := "shared=no");

  function End_Of_File (Resource : in Stream_Type) return Boolean;

  procedure Read
    (Resource : in out Stream_Type;
     Buffer : out Stream_Element_Array;
     Last : out Stream_Element_Offset);

  function Size (Resource : in Stream_Type) return Stream_Element_Offset;

  function Name (Resource : in Stream_Type) return String;

  procedure Reset (Resource : in out Stream_Type);

  procedure Set_Index
    (Resource : in out Stream_Type;
     To : in Stream_Element_Offset);

  procedure Close (Resource : in out Stream_Type);

private

```



```
-- implementation removed  
end AWS.Resources.Streams.Disk;
```

B.36 AWS.Resources.Streams.Disk.Once

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2003-2004                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                                                                              --
--  This library is distributed in the hope that it will be useful, but       --
--  WITHOUT ANY WARRANTY; without even the implied warranty of               --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--  General Public License for more details.                                  --
--                                                                              --
--  You should have received a copy of the GNU General Public License         --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--  As a special exception, if other files instantiate generics from this     --
--  unit, or you link this unit with other files to produce an executable,   --
--  this unit does not by itself cause the resulting executable to be        --
--  covered by the GNU General Public License. This exception does not       --
--  however invalidate any other reasons why the executable file might be    --
--  covered by the GNU Public License.                                        --
-----

--  An ready-to-use implementation of the stream API where the stream content
--  is read from an on-disk file. The file is removed from the file system
--  when the transfer is completed.

package AWS.Resources.Streams.Disk.Once is

  type Stream_Type is new Disk.Stream_Type with null record;

  procedure Close (Resource : in out Stream_Type);
  -- Only redefine Close that will not only close the stream but also delete
  -- the file.

end AWS.Resources.Streams.Disk.Once;

```

B.37 AWS.Resources.Streams.Memory

```

-----
--                                     Ada Web Server                                     --
--                                                                                     --
--                                     Copyright (C) 2003-2005                         --
--                                     AdaCore                                           --
--                                                                                     --
--  This library is free software; you can redistribute it and/or modify             --
--  it under the terms of the GNU General Public License as published by             --
--  the Free Software Foundation; either version 2 of the License, or (at          --
--  your option) any later version.                                                  --
--                                                                                     --
--  This library is distributed in the hope that it will be useful, but             --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                     --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU              --
--  General Public License for more details.                                         --
--                                                                                     --
--  You should have received a copy of the GNU General Public License               --
--  along with this library; if not, write to the Free Software Foundation,        --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                 --
--                                                                                     --
--  As a special exception, if other files instantiate generics from this           --
--  unit, or you link this unit with other files to produce an executable,         --
--  this unit does not by itself cause the resulting executable to be              --
--  covered by the GNU General Public License. This exception does not             --
--  however invalidate any other reasons why the executable file might be          --
--  covered by the GNU Public License.                                              --
-----

-- API to handle a memory stream. A memory stream is first created
-- empty. User can add chunk of data using the Append routines. The stream
-- is then read using the Read procedure.

with AWS.Utils;
with Memory_Streams;

package AWS.Resources.Streams.Memory is

  type Stream_Type is new Streams.Stream_Type with private;

  subtype Stream_Element_Access is Utils.Stream_Element_Array_Access;
  type Buffer_Access is access constant Ada.Streams.Stream_Element_Array;

  procedure Append
    (Resource : in out Stream_Type;
     Buffer    : in    Stream_Element_Array;
     Trim     : in    Boolean := False);
  -- Append Buffer into the memory stream

  procedure Append
    (Resource : in out Stream_Type;
     Buffer    : in    Stream_Element_Access);
  -- Append static data pointed to Buffer into the memory stream as is.
  -- The stream will free the memory on close.

  procedure Append
    (Resource : in out Stream_Type;
     Buffer    : in    Buffer_Access);
  -- Append static data pointed to Buffer into the memory stream as is.
  -- The stream would not try to free the memory on close.

  procedure Read
    (Resource : in out Stream_Type;

```

```

    Buffer    :    out Stream_Element_Array;
    Last      :    out Stream_Element_Offset);
-- Returns a chunk of data in Buffer, Last point to the last element
-- returned in Buffer.

function End_Of_File (Resource : in Stream_Type) return Boolean;
-- Returns True if the end of the memory stream has been reached

procedure Clear (Resource : in out Stream_Type);
pragma Inline (Clear);
-- Delete all data from memory stream

procedure Reset (Resource : in out Stream_Type);
-- Reset the streaming data to the first position

procedure Set_Index
  (Resource : in out Stream_Type;
   To       : in      Stream_Element_Offset);
-- Set the position in the stream, next Read will start at the position
-- whose index is To.

function Size (Resource : in Stream_Type) return Stream_Element_Offset;
-- Returns the number of bytes in the memory stream

procedure Close (Resource : in out Stream_Type);
-- Close the memory stream. Release all memory associated with the stream

private
  -- implementation removed
end AWS.Resources.Streams.Memory;
```

B.38 AWS.Resources.Streams.Memory.ZLib

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2003-2004                             --
--                                     ACT-Europe                                           --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                 --
--  it under the terms of the GNU General Public License as published by                 --
--  the Free Software Foundation; either version 2 of the License, or (at               --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                 --
--  General Public License for more details.                                             --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                     --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

--  This API is used as for standard memory stream (see parent package), the
--  only difference is that the stream is compressing/decompressing on append.

with ZLib;

package AWS.Resources.Streams.Memory.ZLib is

  package ZL renames Standard.ZLib;

  type Stream_Type is new Memory.Stream_Type with private;

  subtype Window_Bits_Type is ZL.Window_Bits_Type;
  subtype Header_Type is ZL.Header_Type;
  subtype Compression_Level is ZL.Compression_Level;
  subtype Strategy_Type is ZL.Strategy_Type;
  subtype Compression_Method is ZL.Compression_Method;
  subtype Memory_Level_Type is ZL.Memory_Level_Type;

  Default_Compression : constant Compression_Level := ZL.Default_Compression;
  Default_Header : constant Header_Type := ZL.Default;

  procedure Deflate_Initialize
    (Resource : in out Stream_Type;
     Level : in Compression_Level := ZL.Default_Compression;
     Strategy : in Strategy_Type := ZL.Default_Strategy;
     Method : in Compression_Method := ZL.Deflated;
     Window_Bits : in Window_Bits_Type := ZL.Default_Window_Bits;
     Memory_Level : in Memory_Level_Type := ZL.Default_Memory_Level;
     Header : in Header_Type := ZL.Default);
  pragma Inline (Deflate_Initialize);
  -- Initialize the compression

  procedure Inflate_Initialize
    (Resource : in out Stream_Type;

```

```

    Window_Bits : in      Window_Bits_Type := ZL.Default_Window_Bits;
    Header       : in      Header_Type      := ZL.Default);
pragma Inline (Inflate_Initialize);
-- Initialize the decompression

procedure Append
(Resource : in out Stream_Type;
 Buffer   : in      Stream_Element_Array;
 Trim    : in      Boolean := False);
-- Compress/decompress and Append Buffer into the memory stream

procedure Read
(Resource : in out Stream_Type;
 Buffer   : out Stream_Element_Array;
 Last    : out Stream_Element_Offset);
-- Returns a chunk of data in Buffer, Last point to the last element
-- returned in Buffer.

function Size (Resource : in Stream_Type) return Stream_Element_Offset;
-- Returns the number of bytes in the memory stream

procedure Close (Resource : in out Stream_Type);
-- Close the ZLib stream, release all memory associated with the Resource
-- object.

private
-- implementation removed
end AWS.Resources.Streams.Memory.ZLib;
```

B.39 AWS.Response

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2005                         --
--                                     AdaCore                                           --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify             --
--  it under the terms of the GNU General Public License as published by             --
--  the Free Software Foundation; either version 2 of the License, or (at          --
--  your option) any later version.                                                  --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but              --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                     --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU              --
--  General Public License for more details.                                         --
--                                                                                       --
--  You should have received a copy of the GNU General Public License               --
--  along with this library; if not, write to the Free Software Foundation,        --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                 --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this           --
--  unit, or you link this unit with other files to produce an executable,         --
--  this unit does not by itself cause the resulting executable to be              --
--  covered by the GNU General Public License. This exception does not             --
--  however invalidate any other reasons why the executable file might be          --
--  covered by the GNU Public License.                                              --
-----

--  This package is to be used to build answer to be sent to the client
--  browser. It is also used as the object returned from the client API. So
--  it is either a response built on the server side or the response received
--  on the client side.

with Ada.Strings.Unbounded;
with Ada.Streams;
with Ada.Finalization;
with Ada.Unchecked_Deallocation;

with AWS.Headers;
with AWS.Messages;
with AWS.MIME;
with AWS.Net;
with AWS.Resources.Streams;
with AWS.Status;

package AWS.Response is

  use Ada;

  type Data is private;
  -- Note that this type use a reference counter which is not thread safe.

  type Callback is access function (Request : in Status.Data) return Data;
  -- This is the Web Server Callback procedure. A client must declare and
  -- pass such procedure to the HTTP record.

  type Data_Mode is
    (Header,      -- Send only the HTTP header
     Message,     -- Send a standard HTTP message
     File,        -- Send a file
     File_Once,   -- Send a file once, delete it after sending
     Stream,      -- Send a stream
  );

```

```

    Socket_Taken,    -- Socket has been taken from the server
    No_Data);        -- No data, this is not a response

type Authentication_Mode is (Any, Basic, Digest);
-- The authentication mode.
-- "Basic" and "Digest" mean that server must accept the requested
-- authentication mode. "Any" mean that server could accept any
-- authentication from client.
-- Note the order here should not be changed as it is used in AWS.Client.

subtype Content_Length_Type is Integer range -1 .. Integer'Last;

Undefined_Length : constant Content_Length_Type;
-- Undefined length could be used when we do not know the message length
-- at the start of transfer. The end of message could be determined by the
-- chunked transfer-encoding in the HTTP/1.1, or by the closing connection
-- in the HTTP/1.0.

Default_Moved_Message : constant String;
-- This is a template message, @_ will be replaced by the Location (see
-- function Build with Location below).

Default_Authenticate_Message : constant String;
-- This is the message that will be displayed on the Web Browser if the
-- authentication process fails or is cancelled.

-----
-- Data Constructors --
-----

function Build
(Content_Type : in String;
 Message_Body : in String;
 Status_Code : in Messages.Status_Code := Messages.S200;
 Cache_Control : in Messages.Cache_Option := Messages.Unspecified;
 Encoding : in Messages.Content_Encoding := Messages.Identity)
return Data;

function Build
(Content_Type : in String;
 UString_Message : in Strings.Unbounded.Unbounded_String;
 Status_Code : in Messages.Status_Code := Messages.S200;
 Cache_Control : in Messages.Cache_Option := Messages.Unspecified;
 Encoding : in Messages.Content_Encoding := Messages.Identity)
return Data;
-- Return a message whose body is passed into Message_Body. The
-- Content_Type parameter is the MIME type for the message
-- body. Status_Code is the response status (see Messages.Status_Code
-- definition).

function Build
(Content_Type : in String;
 Message_Body : in Streams.Stream_Element_Array;
 Status_Code : in Messages.Status_Code := Messages.S200;
 Cache_Control : in Messages.Cache_Option := Messages.Unspecified;
 Encoding : in Messages.Content_Encoding := Messages.Identity)
return Data;
-- Idem above, but the message body is a stream element array

type Disposition_Mode is (Attachment, Inline, None);
-- Describes the way a file/stream is sent to the browser.
--
-- Attachment : The file is sent as an attachment, the browser
-- wont display the content even if the MIME type
-- is supported (.txt or .doc on IE for example).
```



```

--
--      Inline      : The file can be displayed inside the browser if
--                    MIME type is supported. If not the browser will
--                    propose to save this file.
--
--      None        : No specific setting is sent to the browser. The
--                    browser default setting will be used. Note that in
--                    this case the browser determine the filename using
--                    the URI. This is the default setting.
--
function File
(Content_Type : in String;
 Filename     : in String;
 Status_Code  : in Messages.Status_Code      := Messages.S200;
 Cache_Control : in Messages.Cache_Option     := Messages.Unspecified;
 Encoding     : in Messages.Content_Encoding := Messages.Identity;
 Once         : in Boolean                    := False;
 Disposition  : in Disposition_Mode          := None;
 User_Filename : in String                    := "")
return Data;
-- Returns a message whose message body is the content of the file. The
-- Content_Type must indicate the MIME type for the file. User_Filename
-- can be used to force the filename on the client side. This can be
-- different from the server side Filename. If Once is set to True the
-- file will be deleted after the download (this includes the case where
-- the download is suspended).

function Stream
(Content_Type : in String;
 Handle       : not null access Resources.Streams.Stream_Type'Class;
 Status_Code  : in Messages.Status_Code      := Messages.S200;
 Cache_Control : in Messages.Cache_Option     := Messages.No_Cache;
 Encoding     : in Messages.Content_Encoding := Messages.Identity;
 Server_Close  : in Boolean                    := True;
 Disposition  : in Disposition_Mode          := None;
 User_Filename : in String                    := "")
return Data;
-- Returns a message whose message body is the content of the user defined
-- stream. The Content_Type must indicate the MIME type for the data
-- stream, Status_Code is the the header status code which should be send
-- back to client's browser. If Server_Close is set to False the server
-- will not close the stream after sending it, it is then user's
-- responsibility to close the stream. User_Filename can be used to force
-- the filename on the client side. This can be different from the server
-- side filename (for file based stream) or can be used to name a non disk
-- based stream.

-----
-- Redirection Constructors --
-----

function URL
(Location      : in String;
 Cache_Control : in Messages.Cache_Option := Messages.Unspecified)
return Data;
-- This ask the server for a redirection to the specified URL. This is
-- a temporary redirection, and the client browser should query the
-- same original URL next time.

function Moved
(Location      : in String;
 Message       : in String                := Default_Moved_Message;
 Cache_Control : in Messages.Cache_Option := Messages.Unspecified)
return Data;
-- This send back a moved message (Messages.S301) with the specified

```

```

-- message body.
-- This is a permanent redirection, and the client browser is encouraged
-- to update links so that the next query for the URL goes directly to
-- the new location.

-----
-- Other Constructors --
-----

function Acknowledge
  (Status_Code   : in Messages.Status_Code;
   Message_Body  : in String := "";
   Content_Type  : in String := MIME.Text_HTML)
  return Data;
-- Returns a message to the Web browser. This routine must be used to
-- send back an error message to the Web browser. For example if a
-- requested resource cannot be served a message with status code S404
-- must be sent.

function Authenticate
  (Realm   : in String;
   Mode    : in Authentication_Mode := Basic;
   Stale   : in Boolean              := False;
   Message : in String               := Default_Authenticate_Message)
  return Data;
-- Returns an authentication message (Messages.S401), the Web browser
-- will then ask for an authentication. Realm string will be displayed
-- by the Web Browser in the authentication dialog box.

function Socket_Taken return Data;
-- Must be used to say that the connection socket has been taken by user
-- inside of user callback. No operations should be performed on this
-- socket, and associated slot should be released for further operations.

function Empty return Data;
-- Returns an empty message (Data_Mode = No_Data and Status_Code is 204).
-- It is used to say that user's handlers were not able to do something
-- with the request. This is used by the callback's chain in the
-- dispatchers and should not be used by users.

--
-- API to retrieve response data
--

-----
-- Header --
-----

function Header
  (D      : in Data;
   Name   : in String;
   N      : in Positive) return String;
pragma Inline (Header);
-- Return the N-th value for header Name

function Header (D : in Data; Name : in String) return String;
pragma Inline (Header);
-- Return all values as a comma-separated string for header Name.
-- See [RFC 2616 - 4.2] last paragraph.

function Header (D : in Data) return AWS.Headers.List;

procedure Send_Header (Socket : in Net.Socket_Type'Class; D : in Data);
pragma Inline (Send_Header);
-- Send all header lines to the socket

```

```

function Status_Code (D : in Data) return Messages.Status_Code;
pragma Inline (Status_Code);
-- Returns the status code

function Content_Length (D : in Data) return Content_Length_Type;
pragma Inline (Content_Length);
-- Returns the content length (i.e. the message body length). A value of 0
-- indicate that there is no message body.

function Content_Type (D : in Data) return String;
pragma Inline (Content_Type);
-- Returns the MIME type for the message body

function Cache_Control (D : in Data) return Messages.Cache_Option;
pragma Inline (Cache_Control);
-- Returns the cache control specified for the response

function Location (D : in Data) return String;
pragma Inline (Location);
-- Returns the location for the new page in the case of a moved
-- message. See Moved constructor above.

-----
-- Data --
-----

function Mode (D : in Data) return Data_Mode;
pragma Inline (Mode);
-- Returns the data mode, either Header, Message or File

function Message_Body (D : in Data) return String;
pragma Inline (Message_Body);
-- Returns the message body content as a string.
-- Message_Body routines could not be used with user defined streams
-- (see. Stream routine in this package). Constraint_Error would be raised
-- on try to get data by the Message_Body from the user defined streams.
-- For get data from user defined streams routine Create_Resource should
-- be used.

function Message_Body
  (D : in Data)
  return Strings.Unbounded.Unbounded_String;
-- Returns message body content as an unbounded_string

function Message_Body (D : in Data) return Streams.Stream_Element_Array;
-- Returns message body as a binary content

procedure Message_Body
  (D      : in      Data;
   File   : out AWS.Resources.File_Type);
-- Returns the message body as a stream

function Filename (D : in Data) return String;
pragma Inline (Filename);
-- Returns the filename which should be sent back

-----
-- Authentication --
-----

function Realm (D : in Data) return String;
pragma Inline (Realm);
-- Returns the Realm for the current authentication request

```

```

function Authentication (D : in Data) return Authentication_Mode;
pragma Inline (Authentication);
-- Returns the authentication mode requested by server

function Authentication_Stale (D : in Data) return Boolean;
pragma Inline (Authentication_Stale);
-- Returns the stale parameter for authentication

-----
-- Resources --
-----

procedure Create_Resource
  (D      : in out Data;
   File   : out AWS.Resources.File_Type;
   GZip   : in Boolean);
pragma Inline (Create_Resource);
-- Creates the resource object (either a file or in-memory object) for
-- the data to be sent to the client. The resource should be closed after
-- use.
-- GZip is true when the http client support GZip decoding,
-- if file or embedded resource is in the GZip format this routine would
-- define Content-Encoding header field value.

function Close_Resource (D : in Data) return Boolean;
-- Returns True if the resource stream must be close

private
  -- implementation removed
end AWS.Response;

```

B.40 AWS.Server

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                     --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                     --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                     --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                        --
-----

with Ada.Calendar;
with Ada.Exceptions;
with Ada.Finalization;

with AWS.Config;
with AWS.Default;
with AWS.Dispatchers;
with AWS.Exceptions;
with AWS.Hotplug;
with AWS.Log;
with AWS.Net.Acceptors;
with AWS.Net.SSL;
with AWS.Response;
with AWS.Status;
with AWS.Utils;

package AWS.Server is

  type HTTP is limited private;
  -- A Web server

  -----
  -- Server initialization --
  -----

  -- Note that starting a secure server if AWS has not been configured to
  -- support HTTPS will raise Program_Error.

  procedure Start
    (Web_Server : in out HTTP;
     Callback   : in   Response.Callback;
     Config     : in   AWS.Config.Object);
  -- Start server using a full configuration object. With this routine it is
  -- possible to control all features of the server. A simplified version of

```

```
-- Start is also provided below with the most common options.
```

```
procedure Start
```

```
(Web_Server : in out HTTP;
  Dispatcher : in    Dispatchers.Handler'Class;
  Config      : in    AWS.Config.Object);
-- Idem, but using the dispatcher tagged type instead of callback. See
-- AWS.Services.Dispatchers and AWS.Dispatchers hierarchies for built-in
-- services and interface to build your own dispatcher models.
-- Note that a copy of the Dispatcher is kept into Web_Server. Any
-- changes done to the Dispatcher object will not be part of the Web
-- server dispatcher.
```

```
procedure Start
```

```
(Web_Server      : in out HTTP;
  Name            : in    String;
  Callback        : in    Response.Callback;
  Max_Connection  : in    Positive := Default.Max_Connection;
  Admin_URI       : in    String   := Default.Admin_URI;
  Port           : in    Positive := Default.Server_Port;
  Security        : in    Boolean  := False;
  Session         : in    Boolean  := False;
  Case_Sensitive_Parameters : in    Boolean := True;
  Upload_Directory : in    String   := Default.Upload_Directory;
  Line_Stack_Size : in    Positive := Default.Line_Stack_Size);
-- Start the Web server. Max_Connection is the number of simultaneous
-- connections the server's will handle (the number of slots in AWS).
-- Name is just a string used to identify the server. This is used
-- for example in the administrative page. Admin_URI must be set to enable
-- the administrative status page. Callback is the procedure to call for
-- each resource requested. Port is the Web server port. If Security is
-- set to True the server will use an HTTPS/SSL connection. If Session is
-- set to True the server will be able to get a status for each client
-- connected. A session Id is used for that, on the client side it is a
-- cookie. Case_Sensitive_Parameters if set to False it means that the CGI
-- parameters name will be handled without case sensitivity. Upload
-- directory point to a directory where uploaded files will be stored.
```

```
-----
-- Server termination --
-----
```

```
procedure Shutdown (Web_Server : in out HTTP);
```

```
-- Stop the server and release all associated memory. This routine can
-- take some time to terminate because it waits for all tasks to terminate
-- properly before releasing the memory. The log facilities will be
-- automatically stopped by calling Stop_Log below.
```

```
type Termination is (No_Server, Q_Key_Pressed, Forever);
```

```
procedure Wait (Mode : in Termination := No_Server);
```

```
-- The purpose of this procedure is to control the main procedure
-- termination. This procedure will return only when no server are running
-- (No_Server mode) or the 'q' key has been pressed. If mode is set to
-- Forever, Wait will never return and the process will have to be killed.
```

```
-----
-- Server configuration --
-----
```

```
function Config (Web_Server : in HTTP) return AWS.Config.Object;
```

```
-- Returns configuration object for Web_Server
```

```
procedure Set_Unexpected_Exception_Handler
```

```
(Web_Server : in out HTTP;
```

```

    Handler      : in      Exceptions.Unexpected_Exception_Handler);
-- Set the unexpected exception handler. It is called whenever an
-- unrecoverable error has been detected. The default handler just display
-- (on standard output) an error message with the location of the
-- error. By changing this handler it is possible to log or display full
-- symbolic stack backtrace if needed.

procedure Set
(Web_Server : in out HTTP;
 Dispatcher : in      Dispatchers.Handler'Class);
-- Dynamically associate a new dispatcher object to the server. With the
-- feature it is possible to change server behavior at runtime. The
-- complete set of callback procedures will be changed when calling this
-- routine.

procedure Set_Security
(Web_Server      : in out HTTP;
 Certificate_Filename : in      String;
 Security_Mode    : in      Net.SSL.Method := Net.SSL.SSLv23_Server;
 Key_Filename     : in      String        := "");
-- Set security option for AWS. Certificate_Filename is the name of a file
-- containing a certificate. Key_Filename is the name of the file
-- containing the key, if the empty string the key will be taken from the
-- certificate filename. This must be called before starting the secure
-- server otherwise the default security options or options set in the
-- config files will be used. After that the call will have no effect.

procedure Set_Socket_Constructor
(Web_Server      : in out HTTP;
 Socket_Constructor : in      Net.Socket_Constructor);
-- Set the socket constructor routine to use when creating new sockets on
-- the server. By calling this routine it is possible to replace the
-- default AWS communication layer used. The default constructor is
-- AWS.Net.Socket. Note that this routine must be called before starting
-- the server. It is also important to note that sockets returned by the
-- constructor must have the cache properly initialized. See AWS.Net.Cache
-- for more information.

type HTTP_Access is access all HTTP;

function Get_Current return HTTP_Access;
-- Get current server. This can be used in a callback procedure to
-- retrieve the running HTTP server. It is needed when a callback
-- procedure is shared by multiple servers.

function Get_Status return Status.Data;
-- Returns the current status data. This is useful to get the full status
-- in a templates engine callback procedure for example.

-----
-- Other API --
-----

procedure Give_Back_Socket
(Web_Server : in out HTTP; Socket : in Net.Socket_Type'Class);
-- Give the socket back to the server. Socket must have been taken from
-- the server using the Response.Socket_Taken routine in a callback.

procedure Give_Back_Socket
(Web_Server : in out HTTP; Socket : in Net.Socket_Access);
-- Idem.
-- Use Socket_Access to avoid memory reallocation for already allocated
-- sockets.

procedure Set_Field (Id, Value : in String);

```

```
-- Set the extended log field value for the server the controlling the
-- current task.

private
  -- implementation removed
end AWS.Server;
```


B.41 AWS.Server.Hotplug

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2004                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--   This library is free software; you can redistribute it and/or modify     --
--   it under the terms of the GNU General Public License as published by     --
--   the Free Software Foundation; either version 2 of the License, or (at   --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                       --
-----

with AWS.Hotplug;

package AWS.Server.Hotplug is

  -- Messages used to register/unregister hotplug modules

  Register_Message      : constant String := "REGISTER";
  Unregister_Message    : constant String := "UNREGISTER";
  Request_Nonce_Message : constant String := "REQUEST_NONCE";

  -- The Authorization_File below is a file that contains authorizations
  -- for the hotplug modules. Only modules that have an entry into this
  -- file will be able to register to server. Each line on this file must
  -- have the following format:
  --
  -- <module_name>:<md5_password>:<host>:<port>
  --
  -- module_name : The name of the module that will register
  -- md5_password : The corresponding password, use hotplug_password
  --                  tool to generate such password
  -- host         : The host name where requests will be redirected
  -- port         : and the corresponding port

  procedure Activate
    (Web_Server      : in HTTP_Access;
     Port            : in Positive;
     Authorization_File : in String;
     Register_Mode    : in AWS.Hotplug.Register_Mode := AWS.Hotplug.Add);
  -- Start hotplug server listening at the specified Port for the Web_Server.
  -- Only client modules listed in the authorization file will be able to
  -- connect to this server. For better security the host of redirection
  -- must also be specified.

  procedure Shutdown;

```

```
-- Shutdown hotplug server  
end AWS.Server.Hotplug;
```

B.42 AWS.Server.Log

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2003-2004                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--   This library is free software; you can redistribute it and/or modify     --
--   it under the terms of the GNU General Public License as published by     --
--   the Free Software Foundation; either version 2 of the License, or (at   --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                       --
-----

package AWS.Server.Log is

  -----
  -- Standard Log --
  -----

  procedure Start
    (Web_Server      : in out HTTP;
     Split_Mode      : in      AWS.Log.Split_Mode := AWS.Log.None;
     Filename_Prefix : in      String              := "";
     Auto_Flush      : in      Boolean             := False);
  -- Activate server's logging activity. See AWS.Log. If Auto_Flush is True
  -- the file will be flushed after all written data.

  function Name (Web_Server : in HTTP) return String;
  -- Return the name of the Log or an empty string if one is not active

  procedure Stop (Web_Server : in out HTTP);
  -- Stop server's logging activity. See AWS.Log

  function Is_Active
    (Web_Server : in HTTP) return Boolean;
  -- Returns True if the Web Server log has been activated

  procedure Flush (Web_Server : in out HTTP);
  -- Flush the server log.
  -- Note that error log does not need to be flushed because it is always
  -- flushed by default.

  -----
  -- Error Log --
  -----

  procedure Start_Error

```

```
(Web_Server      : in out HTTP;
 Split_Mode      : in      AWS.Log.Split_Mode := AWS.Log.None;
 Filename_Prefix : in      String              := "");
-- Activate server's logging activity. See AWS.Log

function Error_Name (Web_Server : in HTTP) return String;
-- Return the name of the Error Log or an empty string if one is not active

procedure Stop_Error (Web_Server : in out HTTP);
-- Stop server's logging activity. See AWS.Log

function Is_Error_Active (Web_Server : in HTTP) return Boolean;
-- Returns True if the Web Server error log has been activated

end AWS.Server.Log;
```

B.43 AWS.Server.Push

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but       --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License         --
--   along with this library; if not, write to the Free Software Foundation,   --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.           --
--                                                                              --
--   As a special exception, if other files instantiate generics from this     --
--   unit, or you link this unit with other files to produce an executable,    --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not       --
--   however invalidate any other reasons why the executable file might be     --
--   covered by the GNU Public License.                                        --
-----

-- Package to support Server Push feature. This is only supported by Netscape
-- browsers. It will not work with Microsoft Internet Explorer.
-- For Microsoft Internet Explorer complementary active components
-- should be used like java applets or ActiveX controls.

with Ada.Containers.Indefinite_Hashed_Maps;
with Ada.Strings.Hash;
with Ada.Streams;
with Ada.Strings.Unbounded;

with AWS.Net;

generic

  type Client_Output_Type (<>) is private;
  -- Data type client want to send through server push.

  type Client_Environment is private;
  -- Data type to keep client context. This context will be passed to the
  -- conversion routine below.

  with function To_Stream_Array
    (Output : in Client_Output_Type;
     Client : in Client_Environment)
    return Ada.Streams.Stream_Element_Array;
  -- Function used for convert Client_Output_Type to Stream_Output_Type.
  -- This is used by the server to prepare the data to be sent to the
  -- clients.

package AWS.Server.Push is

  use Ada.Strings.Unbounded;

  Client_Gone : exception;

```

```

-- Raised when a client is not responding

Closed : exception;
-- Raised when trying to register to a closed push server

Duplicate_Client_Id : exception;
-- Raised in trying to register an already registered client

type Object is limited private;
-- This is the push server object. A push server has two modes, either it
-- is Open or Closed. When open it will send data to registered
-- clients. No data will be sent to registered client if the server is
-- Closed.

type Mode is (Plain, Multipart, Chunked);
-- Described the mode to communicate with the client.
-- Plain      : no transformation is done, the data are sent as-is
-- Multipart  : data are MIME encoded.
-- Chunked    : data are chunked, a piece of data is sent in small pieces.

subtype Client_Key is String;
-- The Client Id key representation. In a server each client must have a
-- uniq ID. This Id is used for registration and for sending data to
-- specific client.

type Group_Set is array (Positive range <>) of Unbounded_String;

Empty_Group : constant Group_Set := (1 .. 0 => Null_Unbounded_String);

procedure Register
  (Server      : in out Object;
   Client_Id   : in    Client_Key;
   Socket      : in    Net.Socket_Type'Class;
   Environment : in    Client_Environment;
   Init_Data   : in    Client_Output_Type;
   Init_Content_Type : in String := "";
   Kind        : in    Mode      := Plain;
   Duplicated_Age : in    Duration := Duration'Last;
   Groups      : in    Group_Set := Empty_Group);
-- Add client identified by Client_Id to the server subscription
-- list and send the Init_Data (as a Data_Content_Type mime content) to
-- him. After registering this client will be able to receive pushed data
-- from the server in broadcasting mode.
-- If Duplicated_Age less than age of the already registered same Client_Id
-- then old one will be unregistered first (no exception will be raised).

procedure Register
  (Server      : in out Object;
   Client_Id   : in    Client_Key;
   Socket      : in    Net.Socket_Type'Class;
   Environment : in    Client_Environment;
   Kind        : in    Mode      := Plain;
   Duplicated_Age : in    Duration := Duration'Last;
   Groups      : in    Group_Set := Empty_Group);
-- Same as above but without sending initial data

procedure Unregister
  (Server      : in out Object;
   Client_Id   : in    Client_Key;
   Close_Socket : in    Boolean := True);
-- Removes client Client_Id from server subscription list. The associated
-- client's socket will be closed if Close_Socket is True. No exception is
-- raised if Client_Id was not registered.

procedure Unregister_Clients

```

```

(Server      : in out Object;
  Close_Sockets : in Boolean := True);
-- Remove all registered clients from the server. Closes if Close_Sockets
-- is set to True (default) otherwise the sockets remain open. After this
-- call the sever will still in running mode. Does nothing if there is no
-- client registered.

procedure Send_To
(Server      : in out Object;
  Client_Id  : in Client_Key;
  Data       : in Client_Output_Type;
  Content_Type : in String := "");
-- Push data to a specified client identified by Client_Id

procedure Send
(Server      : in out Object;
  Data       : in Client_Output_Type;
  Group_Id   : in String := "";
  Content_Type : in String := "");
-- Push data to group of clients (broadcast) subscribed to the server.
-- If Group_Id is empty, data transferred to each client.

procedure Send
(Server      : in out Object;
  Data       : in Client_Output_Type;
  Client_Gone : access procedure (Client_Id : in String);
  Group_Id   : in String := "";
  Content_Type : in String := "");
-- Push data to group of clients (broadcast) subscribed to the server.
-- If Group_Id is empty, data transferred to each client.
-- Call Client_Gone for each client with broken socket.

generic
  with procedure Client_Gone (Client_Id : in String);
procedure Send_G
(Server      : in out Object;
  Data       : in Client_Output_Type;
  Group_Id   : in String := "";
  Content_Type : in String := "");
-- Same like before, but generic for back compatibility.

function Count (Server : in Object) return Natural;
-- Returns the number of registered clients in the server

function Is_Open (Server : in Object) return Boolean;
-- Return True if the server is open, meaning server is still running,
-- ready to accept client's registration and still sending data to
-- clients.

-- Shutdown routines put the server in a Closed mode. The routines below
-- provides a way to eventually close the socket, to send some
-- finalisation data.

procedure Shutdown
(Server      : in out Object;
  Close_Sockets : in Boolean := True);
-- Unregistered all clients and close all associated connections (socket) if
-- Close_Socket is True. The server will be in Closed mode. After this
-- call any client trying to register will get the Closed exception. It is
-- possible to reactivate the server with Restart.

procedure Shutdown
(Server      : in out Object;
  Final_Data : in Client_Output_Type;
  Final_Content_Type : in String := "");

```

```
-- Idem as above but it send Final_Data (as a Data_Content_Type mime
-- content) before closing connections.

procedure Shutdown_If_Empty
  (Server : in out Object;
   Open   : out Boolean);
-- Server will be shutdown (close mode) if there is no more active clients
-- (Count = 0). Returns new server status in Open (Open will be True if
-- server is in Open mode and False otherwise). After this call any client
-- trying to register will get the Closed exception. It is possible to
-- reactivate the server with Restart.

procedure Restart (Server : in out Object);
-- Set server to Open mode. Server will again send data to registered
-- clients. It does nothing if server was already open.

private
  -- implementation removed
end AWS.Server.Push;
```


B.44 AWS.Server.Status

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2003-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                        --
-----

--   This package provides routine to retrieve server's internal status

with Ada.Calendar;

with AWS.Templates;

package AWS.Server.Status is

  function Translations (Server : in HTTP) return Templates.Translate_Table;
  -- Returns a translate table to be used with a template file. This table
  -- contains all internal server's data. This table is used by the server
  -- internal status page for example.

  function Start_Time (Server : in HTTP) return Ada.Calendar.Time;
  -- Returns the server's start time

  function Resources_Served (Server : in HTTP) return Natural;
  -- Returns the total number of resources (static file, templates,
  -- in-memory string) served by the server.

  function Socket (Server : in HTTP) return Net.Socket_Type'Class;
  -- Returns the server's socket

  function Current_Connections (Server : in HTTP) return Natural;
  -- Returns the current number of connections

  function Is_Session_Activated (Server : in HTTP) return Boolean;
  -- Returns True if the session feature has been activated

  function Is_Security_Activated (Server : in HTTP) return Boolean;
  -- Returns True if the HTTPS protocol is used

  function Is_Shutdown (Server : in HTTP) return Boolean;
  -- Returns True if server has been stopped (the server could still be in

```

```
    -- the shutdown phase).  
end AWS.Server.Status;
```

B.45 AWS.Services.Callbacks

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2004                               --
--                                     ACT-Europe                                       --
--                                     -----                                       --
--  This library is free software; you can redistribute it and/or modify             --
--  it under the terms of the GNU General Public License as published by             --
--  the Free Software Foundation; either version 2 of the License, or (at          --
--  your option) any later version.                                                 --
--                                                                                     --
--  This library is distributed in the hope that it will be useful, but             --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                     --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU              --
--  General Public License for more details.                                         --
--                                                                                     --
--  You should have received a copy of the GNU General Public License               --
--  along with this library; if not, write to the Free Software Foundation,        --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                 --
--                                                                                     --
--  As a special exception, if other files instantiate generics from this           --
--  unit, or you link this unit with other files to produce an executable,         --
--  this unit does not by itself cause the resulting executable to be              --
--  covered by the GNU General Public License. This exception does not             --
--  however invalidate any other reasons why the executable file might be          --
--  covered by the GNU Public License.                                              --
-----

--  Services to be used to declare aliases based on URI. This is mostly
--  designed to be used with AWS.services.Dispatchers.URI.

with AWS.Response;
with AWS.Status;

package AWS.Services.Callbacks is

  use AWS;

  generic
    Prefix    : String; -- the prefix found in the URI
    Directory : String; -- the directory where the file is
  function File (Request : in Status.Data) return Response.Data;
  --  This is a callback function where URL:
  --    http://<host>/<prefix>toto
  --  references the file:
  --    <directory>/toto
  --
  --  If the URL does not start with Prefix it returns a 404 error page.
  --  This is designed to be use with AWS.Services.Dispatchers.URI.

end AWS.Services.Callbacks;

```

B.46 AWS.Services.Directory

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2003                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--   This library is free software; you can redistribute it and/or modify     --
--   it under the terms of the GNU General Public License as published by     --
--   the Free Software Foundation; either version 2 of the License, or (at   --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of              --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU       --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License       --
--   along with this library; if not, write to the Free Software Foundation, --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.         --
--                                                                              --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,  --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be   --
--   covered by the GNU Public License.                                       --
-----

with AWS.Status;
with Templates_Parser;

--   This service can be used to browse a file system. The browsing mechanism
--   will gather information (filename, size, directory...) from a specified
--   directory name and will fill a translation table. This table will be used
--   with a template file to render the HTML document. You can design your own
--   browsing template file, here is a description of all tag variables defined
--   in the translation table:
--
--   URI (discrete)
--       The URI pointing to the directory parsed.
--
--   VERSION (discrete)
--       AWS version string.
--
--   IS_DIR_V (vector)
--       A list of booleans, indicate if Nth entry is a directory or not.
--
--   NAME_V (vector)
--       A list of filenames. Nth name is a directory if Nth entry in IS_DIR
--       is set to true.
--
--   SIZE_V (vector)
--       A list of sizes. Nth entry is the file size of the Nth entry in
--       NAMES.
--
--   TIME_V (vector)
--       A list of last modification times. Nth entry is is the last
--       modification time of the Nth entry in NAMES.
--
--   NAME_ORDR
--       Rule to either set ordering on name or to revert current name
--       ordering.

```

```

--
-- SNME_ORDR
-- Rule to either set ordering on name (case sensitive) or to revert
-- current name (case sensitive) ordering.
--
-- EXT_ORDR
-- Rule to either set ordering on extension or to revert current
-- extension ordering.
--
-- SEXT_ORDR
-- Rule to either set ordering on extension (case sensitive) or to
-- revert current extension (case sensitive) ordering.
--
-- MIME_ORDR
-- Rule to either set ordering on MIME type or to revert current MIME
-- type ordering.
--
-- DIR_ORDR
-- Rule to either set ordering on directory or to revert current
-- directory ordering.
--
-- SIZE_ORDR
-- Rule to either set ordering on size or to revert current size
-- ordering.
--
-- TIME_ORDR
-- Rule to either set ordering on time or to revert current time
-- ordering.
--
-- ORIG_ORDR
-- Rule to either set original ordering (file order as read on the file
-- system) or to revert current original ordering.
--
-- DIR_NAME_ORDR
-- Rule to either set ordering on directory/name or to revert current
-- directory/name ordering.
--
-- DIR_SNME_ORDR
-- Rule to either set ordering on directory/name (case sensitive) or to
-- revert current directory/name (case sensitive) ordering.
--
-- DIR_TIME_ORDR
-- Rule to either set ordering on directory/time or to revert current
-- directory/time ordering.
--

package AWS.Services.Directory is

  use Templates_Parser;

  function Browse
    (Directory_Name : in String;
     Request        : in AWS.Status.Data)
    return Translate_Table;
  -- Returns a translation table containing information parsed from
  -- Directory_Name. This is supposed to be used with a directory template.

  function Browse
    (Directory_Name      : in String;
     Template_Filename   : in String;
     Request              : in AWS.Status.Data;
     Translations         : in Translate_Table := No_Translation)
    return String;
  -- Parses directory Directory_Name and use Templates_Parser to fill in the
  -- template Template_Filename. It is possible to specified some specifics

```

```
-- tags in Translations.  
end AWS.Services.Directory;
```

B.47 AWS.Services.Dispatchers

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2005                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                               --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of                --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                               --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                               --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                        --
-----

package AWS.Services.Dispatchers is

  pragma Pure;

  -- Services on the Dispatcher tree are to help building big servers.
  -- Experiences shows that a lot of user's code is to check the value of a
  -- specific URI or request method to call the right callback that will
  -- handle the request. This code is a big "if/elsif/end if" that just hide
  -- the real job. A dispatcher is to replace this code. Currently there is
  -- five of them:
  --
  -- URI (AWS.Services.Dispatchers.URI)
  --   to dispatch to a callback depending of the ressource name.
  --
  -- Method (AWS.Services.Dispatchers.Method)
  --   to dispatch to a callback depending of the request method.
  --
  -- Virtual_Host (AWS.Services.Dispatchers.Virtual_Host)
  --   to dispatch to a callback depending on the host name. This is known
  --   as virtual hosting and permit to have multiple servers on the same
  --   machine using the same port.
  --
  -- Transient_Pages (AWS.Services.Dispatchers.Transient_Pages)
  --   to handle transient pages, if the default user's callback returns
  --   404 this dispatcher checks if the requested resource is a transient
  --   page.
  --
  -- Timer (AWS.Services.Dispatchers.Timer)
  --   to dispatch to a specific callback depending on the current time.
  --
  -- Linker (AWS.Services.Dispatchers.Linker)
  --   to link two dispatchers together, if the first one retruns 404 tries
  --   the second one.

```

```
end AWS.Services.Dispatchers;
```


B.48 AWS.Services.Dispatchers.Linker

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2005                               --
--                                     AdaCore                                           --
--                                     -----
--  This library is free software; you can redistribute it and/or modify
--  it under the terms of the GNU General Public License as published by
--  the Free Software Foundation; either version 2 of the License, or (at
--  your option) any later version.
--
--  This library is distributed in the hope that it will be useful, but
--  WITHOUT ANY WARRANTY; without even the implied warranty of
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
--  General Public License for more details.
--
--  You should have received a copy of the GNU General Public License
--  along with this library; if not, write to the Free Software Foundation,
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
--
--  As a special exception, if other files instantiate generics from this
--  unit, or you link this unit with other files to produce an executable,
--  this unit does not by itself cause the resulting executable to be
--  covered by the GNU General Public License. This exception does not
--  however invalidate any other reasons why the executable file might be
--  covered by the GNU Public License.
-----

--  Link two dispatchers together

with AWS.Dispatchers;
with AWS.Response;
with AWS.Status;

package AWS.Services.Dispatchers.Linker is

  type Handler is new AWS.Dispatchers.Handler with private;

  function Dispatch
    (Dispatcher : in Handler;
     Request    : in Status.Data)
    return Response.Data;
  -- Dispatch to the first dispatcher, if the resources is not found (status
  -- code 404 returned) there try on the second one.

  procedure Register
    (Dispatcher : in out Handler;
     First, Second : in AWS.Dispatchers.Handler'Class);
  -- Set the dispatcher first and second handler. The First handler will be
  -- looked for before the second.

private
  -- implementation removed
end AWS.Services.Dispatchers.Linker;

```

B.49 AWS.Services.Dispatchers.Method

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2006                           --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU                   --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

-- Dispatch a specific request to a callback depending on the request method.

with AWS.Dispatchers;
with AWS.Response;
with AWS.Status;

package AWS.Services.Dispatchers.Method is

  type Handler is new AWS.Dispatchers.Handler with private;

  function Dispatch
    (Dispatcher : in Handler;
     Request    : in Status.Data)
    return Response.Data;
  -- Dispatch to the corresponding method callback, if no such callback
  -- registered it dispatches to the default callback. If there is no default
  -- callback it returns an error message (code 404).

  procedure Register
    (Dispatcher : in out Handler;
     Method     : in     Status.Request_Method;
     Action     : in     AWS.Dispatchers.Handler'Class);
  -- Register callback to use for a specific request method.

  procedure Register
    (Dispatcher : in out Handler;
     Method     : in     Status.Request_Method;
     Action     : in     Response.Callback);
  -- Idem as above but take a callback procedure as parameter.

  procedure Unregister
    (Dispatcher : in out Handler;
     Method     : in     Status.Request_Method);
  -- Removes Method from the list of request method to handle.

```

```
procedure Register_Default_Callback
  (Dispatcher : in out Handler;
   Action      : in      AWS.Dispatchers.Handler'Class);
-- Register the default callback. This will be used if no request method
-- have been activated.

private
  -- implementation removed
end AWS.Services.Dispatchers.Method;
```

B.50 AWS.Services.Dispatchers.URI

```

-----
--                                     Ada Web Server                                     --
--                                                                                     --
--                                     Copyright (C) 2000-2006                         --
--                                     AdaCore                                           --
--                                                                                     --
--  This library is free software; you can redistribute it and/or modify             --
--  it under the terms of the GNU General Public License as published by             --
--  the Free Software Foundation; either version 2 of the License, or (at          --
--  your option) any later version.                                                  --
--                                                                                     --
--  This library is distributed in the hope that it will be useful, but             --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                     --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU              --
--  General Public License for more details.                                         --
--                                                                                     --
--  You should have received a copy of the GNU General Public License               --
--  along with this library; if not, write to the Free Software Foundation,        --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                 --
--                                                                                     --
--  As a special exception, if other files instantiate generics from this          --
--  unit, or you link this unit with other files to produce an executable,         --
--  this unit does not by itself cause the resulting executable to be              --
--  covered by the GNU General Public License. This exception does not             --
--  however invalidate any other reasons why the executable file might be          --
--  covered by the GNU Public License.                                              --
-----

-- Dispatch a specific request to a callback depending on the URI

with Ada.Strings.Unbounded;

with Ada.Containers.Vectors;

with AWS.Dispatchers;
with AWS.Response;
with AWS.Status;

package AWS.Services.Dispatchers.URI is

  type Handler is new AWS.Dispatchers.Handler with private;

  function Dispatch
    (Dispatcher : in Handler;
     Request    : in Status.Data)
    return Response.Data;
  -- Dispatch will return the value returned by the first callback matching
  -- the request. Note that if a callback returns the Response.Empty message,
  -- Dispatch will just continue to the next matching callback. In any case,
  -- if no handler matches it will call the default callback. If no default
  -- callback is registered an error (code 404) HTML message is returned.

  procedure Register
    (Dispatcher : in out Handler;
     URI        : in String;
     Action     : in AWS.Dispatchers.Handler'Class;
     Prefix     : in Boolean := False);
  -- Register URI to use the specified dispatcher. URI is the full string
  -- that must match the ressource requested (with the leading /). If Prefix
  -- is True, only the URI prefix is checked.

  procedure Register

```

```

    (Dispatcher : in out Handler;
     URI        : in      String;
     Action     : in      Response.Callback;
     Prefix     : in      Boolean := False);
-- Idem as above but take a callback procedure as parameter

procedure Register_Regexp
  (Dispatcher : in out Handler;
   URI        : in      String;
   Action     : in      AWS.Dispatchers.Handler'Class);
-- Register URI to use the specified dispatcher. URI is a regular
-- expression that must match the ressource requested (with the leading /).

procedure Register_Regexp
  (Dispatcher : in out Handler;
   URI        : in      String;
   Action     : in      Response.Callback);
-- Idem as above but take a callback procedure as parameter

procedure Unregister
  (Dispatcher : in out Handler;
   URI        : in      String);
-- Removes URI from the list. URI is either a name or a regexp and must
-- have exactaly the value used with Register.

procedure Register_Default_Callback
  (Dispatcher : in out Handler;
   Action     : in      AWS.Dispatchers.Handler'Class);
-- Register the default callback. This will be used if no URI match
-- the request.

private
  -- implementation removed
end AWS.Services.Dispatchers.URI;

```

B.51 AWS.Services.Dispatchers.Virtual_Host

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2006                             --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                          --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                  --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                  --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

with Ada.Strings.Unbounded;

with Strings_Maps;

with AWS.Dispatchers;
with AWS.Response;
with AWS.Status;

package AWS.Services.Dispatchers.Virtual_Host is

  type Handler is new AWS.Dispatchers.Handler with private;

  function Dispatch
    (Dispatcher : in Handler;
     Request    : in Status.Data)
    return Response.Data;
  -- Returns an error message (code 404) if there is no match for the request

  procedure Register
    (Dispatcher      : in out Handler;
     Virtual_Hostname : in String;
     Hostname        : in String);
  -- Register Virtual_Hostname to be a redirection to the specified
  -- hostname.

  procedure Register
    (Dispatcher      : in out Handler;
     Virtual_Hostname : in String;
     Action          : in AWS.Dispatchers.Handler'Class);
  -- Register Virtual_Hostname to use the specified callback.

  procedure Register
    (Dispatcher      : in out Handler;
     Virtual_Hostname : in String;

```

```
        Action      : in      Response.Callback);  
-- Idem as above but take a callback procedure as parameter.  
  
procedure Unregister  
  (Dispatcher      : in out Handler;  
   Virtual_Hostname : in      String);  
-- Removes Virtual_Hostname from the list of virtual hostnames to handle.  
  
procedure Register_Default_Callback  
  (Dispatcher : in out Handler;  
   Action      : in      AWS.Dispatchers.Handler'Class);  
-- Register the default callback. This will be used if no Virtual_Hostname  
-- match the request.  
  
private  
  -- implementation removed  
end AWS.Services.Dispatchers.Virtual_Host;
```

B.52 AWS.Services.Download

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2005-2006                           --
--                                     AdaCore                                           --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                     --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU                   --
--  General Public License for more details.                                           --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                     --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                  --
-----

--  This is a download manager service, can be used to avoid polluting the main
--  server slot with long downloads. A single task is used in this
--  implementation.

with AWS.Config;
with AWS.Dispatchers;
with AWS.Resources.Streams;
with AWS.Response;
with AWS.Services.Dispatchers.Linker;
with AWS.Status;

package AWS.Services.Download is

  procedure Start
    (Server_Dispatcher      : in    AWS.Dispatchers.Handler'Class;
     Main_Dispatcher        : out   Services.Dispatchers.Linker.Handler;
     Max_Concurrent_Download : in    Positive :=
       Config.Max_Concurrent_Download);
    -- Start the download manager server. Server_Dispatcher is the dispatcher
    -- for the Web server. Main_Dispatcher is the dispatcher that must be used
    -- with the main server start routine. This dispatcher handles the standard
    -- web server resources and the download manager ones.
    -- Max_Concurrent_Download contains the number of simultaneous download
    -- that can be handled, request past this limit are queued. Note that a
    -- single task is used for this implementation. Using a download manager is
    -- useful to avoid the standard Web server to be busy with long downloads.

  procedure Stop;
    -- Stop the download server, all current download are interrupted

  function Build
    (Request : in Status.Data;
     Name     : in String;
     Resource : not null access Resources.Streams.Stream_Type'Class)

```



```
    return Response.Data;
-- Queue a download request. If there is room on the download manager the
-- template page aws_download_manager_start.thtml is used to build the
-- answer otherwise the template page aws_download_manager_waiting.thtml is
-- used. Name is the resource name and will be the default name used on the
-- user side to save the file on disk. Resource is a stream on which the
-- data to be sent are read.
--
-- Templates tags description:
--
-- aws_download_manager_waiting.thtml
--     NAME      the name of the resource as pass to build
--     RES_URI   the resource URI unique to the download server
--     POSITION   the position on the waiting queue
-- aws_download_manager_start.thtml
--     NAME      the name of the resource as pass to build
--     RES_URI   the resource URI unique to the download server
--
-- Note that both template pages must contain a refresh meta-tag:
--
--     <meta http-equiv="refresh" content="2">
end AWS.Services.Download;
```

B.53 AWS.Services.Page_Server

```

-----
--                                Ada Web Server                                --
--                                Copyright (C) 2000-2001                        --
--                                ACT-Europe                                     --
--                                -----                                     --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                                                                              --
--  This library is distributed in the hope that it will be useful, but      --
--  WITHOUT ANY WARRANTY; without even the implied warranty of               --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--  General Public License for more details.                                  --
--                                                                              --
--  You should have received a copy of the GNU General Public License        --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--  As a special exception, if other files instantiate generics from this    --
--  unit, or you link this unit with other files to produce an executable,  --
--  this unit does not by itself cause the resulting executable to be       --
--  covered by the GNU General Public License. This exception does not      --
--  however invalidate any other reasons why the executable file might be   --
--  covered by the GNU Public License.                                       --
-----

--  The Callback is an implementation of a simple static Web page server. It
--  will return the Web pages found in the Web server directory. If directory
--  browsing is activated, it will be possible to browse directory content if
--  the requested ressource is a directory. There is two specials files that
--  are recognized:

--      404.shtml                  The Web page returned if the requested page is
--                                not found. This is a template with a single tag
--                                variable named PAGE. It will be replaced by the
--                                ressource which was not found.

--                                Note that on Microsoft IE this page will be
--                                displayed only if the total page size is bigger
--                                than 512 bytes or it includes at least one
--                                image.

--      aws_directory.shtml        The template page used for directory browsing.
--                                See AWS.Services.Directory for a full description
--                                of this template usage.

with AWS.Status;
with AWS.Response;

package AWS.Services.Page_Server is

  procedure Directory_Browsing (Activated : in Boolean);
  -- If Activated is set to True the directory browsing faciity will be
  -- activated. By default this feature is not activated.

  function Callback (Request : in AWS.Status.Data) return AWS.Response.Data;
  -- This is the AWS callback for the simple static Web pages server.

end AWS.Services.Page_Server;

```

B.54 AWS.Services.Split_Pages

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2003-2004                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--   This library is free software; you can redistribute it and/or modify     --
--   it under the terms of the GNU General Public License as published by     --
--   the Free Software Foundation; either version 2 of the License, or (at   --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                       --
-----

with Ada.Strings.Unbounded;
with AWS.Response;
with AWS.Templates;

package AWS.Services.Split_Pages is

  use Ada.Strings.Unbounded;

  Splitter_Error : exception;

  -- This package provides an API to split a big table in multiple pages
  -- using the transient Web Pages support.

  type Page_Range is record
    First : Positive;
    Last  : Natural; -- For an empty range, Last < First
  end record;

  type Ranges_Table is array (Positive range <>) of Page_Range;
  type URI_Table    is array (Positive range <>) of Unbounded_String;

  type Splitter is abstract tagged limited private;
  -- This is the (abstract) root class of all splitters
  -- Two operations are necessary: Get_Page_Ranges and Get_Translations
  -- The following tags are always defined by the Parse function; however,
  -- if a splitter redefines them in Get_Translations, the new definition
  -- will replace the standard one:
  -- NUMBER_PAGES  Number of pages generated.
  -- PAGE_NUMBER   Position of the current page in all pages
  -- OFFSET        Current table line offset real table line can be computed
  --               using: @_"+"(OFFSET):TABLE_LINE_@

  function Get_Page_Ranges
    (This : in Splitter;

```

```

    Table : in Templates.Translate_Set)
    return Ranges_Table is abstract;
-- Get_Page_Ranges is called to define the range (in lines) of each split
-- page. Note that the ranges may overlap and need not cover the full
-- table.

function Get_Translations
  (This      : in Splitter;
   Page      : in Positive;
   URIs      : in URI_Table;
   Ranges    : in Ranges_Table)
  return Templates.Translate_Set is abstract;
-- Get_Translations builds the translation table for use with the splitter

function Parse
  (Template      : in String;
   Translations : in Templates.Translate_Set;
   Table         : in Templates.Translate_Set;
   Split_Rule    : in Splitter'Class;
   Cached        : in Boolean := True)
  return Response.Data;

function Parse
  (Template      : in String;
   Translations : in Templates.Translate_Table;
   Table         : in Templates.Translate_Table;
   Split_Rule    : in Splitter'Class;
   Cached        : in Boolean := True)
  return Response.Data;
-- Parse the Template file and split the result in multiple pages.
-- Translations is a standard Translate_Set used for all pages. Table
-- is the Translate_Set containing data for the table to split in
-- multiple pages. This table will be analysed and according to the
-- Split_Rule, a set of transient pages will be created.
-- If Cached is True the template will be cached (see Templates_Parser
-- documentation).
-- Each Split_Rule define a number of specific tags for use in the template
-- file.

function Parse
  (Template      : in String;
   Translations : in Templates.Translate_Table;
   Table         : in Templates.Translate_Table;
   Max_Per_Page : in Positive := 25;
   Max_In_Index : in Positive := 20;
   Cached        : in Boolean := True)
  return Response.Data;
-- Compatibility function with previous version of AWS.
-- Uses the Uniform_Splitter
-- Note that the Max_In_Index parameter is ignored.
-- The same effect can be achieved by using the bounded_index.html
-- template for displaying the index.

private
  -- implementation removed
end AWS.Services.Split_Pages;

```

B.55 AWS.Services.Split_Pages.Alpha

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2004                               --
--                                     ACT-Europe                                         --
--                                                                                       --
--   This library is free software; you can redistribute it and/or modify             --
--   it under the terms of the GNU General Public License as published by             --
--   the Free Software Foundation; either version 2 of the License, or (at          --
--   your option) any later version.                                                  --
--                                                                                       --
--   This library is distributed in the hope that it will be useful, but             --
--   WITHOUT ANY WARRANTY; without even the implied warranty of                     --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU              --
--   General Public License for more details.                                         --
--                                                                                       --
--   You should have received a copy of the GNU General Public License               --
--   along with this library; if not, write to the Free Software Foundation,        --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                 --
--                                                                                       --
--   As a special exception, if other files instantiate generics from this          --
--   unit, or you link this unit with other files to produce an executable,         --
--   this unit does not by itself cause the resulting executable to be              --
--   covered by the GNU General Public License. This exception does not             --
--   however invalidate any other reasons why the executable file might be          --
--   covered by the GNU Public License.                                              --
-----

package AWS.Services.Split_Pages.Alpha is

  -- Split in (at most) 28 pages, one for empty fields, one for all fields
  -- that start with a digit, and one for each different initial letter.
  -- Note that leading spaces in the key field are ignored; this means that a
  -- key field containing only spaces is treated as an empty field.
  -- The key field is set by calling Set_Key. If no key is defined, or no
  -- corresponding association is found in Table, or the association is not a
  -- vector, Splitter_Error is raised.
  -- The key field must be sorted, and all values must be empty or start with
  -- a digit or letter (case ignored). Otherwise, Splitter_Error is raised.
  -- Letters that do not appear in the key field are associated to the empty
  -- string; an Href can be specified instead by calling Set_Default_Href.
  --
  -- Tags:
  -- NEXT           The href to the next page.
  -- PREVIOUS       The href to the previous page.
  -- FIRST          The href to the first page.
  -- LAST           The href to the last page.
  -- PAGE_INDEX     Position of the current page in the INDEXES_V vector
  -- HREFS_V         A vector tag containing a set of href to pages, or "" if
  --                 their is no page for the corresponding letter.
  -- INDEXES_V      A vector tag (synchronized with HREFS_V) containing ' '
  --                 and the letters 'A' .. 'Z'
  --
  -- HREFS_V and INDEXES_V can be used to create an index to the generated
  -- pages.

  type Splitter is new Split_Pages.Splitter with private;

  function Get_Page_Ranges
    (This : in Splitter;
     Table : in Templates.Translate_Set)
    return Ranges_Table;

```

```
function Get_Translations
  (This      : in Splitter;
   Page      : in Positive;
   URIs      : in URI_Table;
   Ranges    : in Ranges_Table)
  return Templates.Translate_Set;

procedure Set_Key (This : in out Splitter; Key : in String);
-- Set the key field, this is the name of the vector association in the
-- translate_set that will be used to create the index.

procedure Set_Default_Href (This : in out Splitter; Href : in String);
-- Href to use for letter having no entry in the key, if not specified the
-- empty string is used.

private
  -- implementation removed
end AWS.Services.Split_Pages.Alpha;
```

B.56 AWS.Services.Split_Pages.Alpha.Bounded

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2004                               --
--                                     ACT-Europe                                         --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify               --
--  it under the terms of the GNU General Public License as published by               --
--  the Free Software Foundation; either version 2 of the License, or (at            --
--  your option) any later version.                                                    --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                       --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                --
--  General Public License for more details.                                           --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                 --
--  along with this library; if not, write to the Free Software Foundation,          --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                   --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this             --
--  unit, or you link this unit with other files to produce an executable,           --
--  this unit does not by itself cause the resulting executable to be                 --
--  covered by the GNU General Public License. This exception does not                --
--  however invalidate any other reasons why the executable file might be            --
--  covered by the GNU Public License.                                                 --
-----

package AWS.Services.Split_Pages.Alpha.Bounded is

  -- Same as the alpha splitter, but pages larger than Max_Per_Page are
  -- further splitted.
  -- A secondary index is generated that gives the various pages for a given
  -- letter.
  --
  -- Tags (in addition to those of the alpha splitter):
  -- S_NEXT      The href to the next page.
  -- S_PREVIOUS  The href to the previous page.
  -- S_FIRST     The href to the first page.
  -- S_LAST      The href to the last page.
  -- S_PAGE_INDEX Position of the current page in the S_INDEXES_V vector
  --              Note that for this splitter, this is also the page number.
  -- S_HREFS_V   A vector tag containing a set of href to the different
  --              pages for the current letter.
  -- S_INDEXES_V A vector tag (synchronized with S_HREFS_V) containing the
  --              page numbers for the hrefs.
  --
  -- HREFS_V and INDEXES_V can be used to create an index to the generated
  -- pages. S_HREFS_V and S_INDEXES_V can be used to create a secondary
  -- alphabetical index that points directly to the corresponding element.

  type Splitter (Max_Per_Page : Positive) is new Alpha.Splitter with private;

  function Get_Page_Ranges
    (This : in Splitter;
     Table : in Templates.Translate_Set)
    return Ranges_Table;

  function Get_Translations
    (This : in Splitter;
     Page : in Positive;
     URIs : in URI_Table;

```

```
      Ranges : in Ranges_Table)
    return Templates.Translate_Set;

private
  -- implementation removed
end AWS.Services.Split_Pages.Alpha.Bounded;
```


B.57 AWS.Services.Split_Pages.Uniform

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2004                          --
--                               ACT-Europe                                   --
--                               -----                                   --
--   This library is free software; you can redistribute it and/or modify   --
--   it under the terms of the GNU General Public License as published by   --
--   the Free Software Foundation; either version 2 of the License, or (at  --
--   your option) any later version.                                         --
--                                                                           --
--   This library is distributed in the hope that it will be useful, but    --
--   WITHOUT ANY WARRANTY; without even the implied warranty of             --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU      --
--   General Public License for more details.                               --
--                                                                           --
--   You should have received a copy of the GNU General Public License      --
--   along with this library; if not, write to the Free Software Foundation, --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.        --
--                                                                           --
--   As a special exception, if other files instantiate generics from this   --
--   unit, or you link this unit with other files to produce an executable, --
--   this unit does not by itself cause the resulting executable to be      --
--   covered by the GNU General Public License. This exception does not     --
--   however invalidate any other reasons why the executable file might be   --
--   covered by the GNU Public License.                                     --
-----

package AWS.Services.Split_Pages.Uniform is

  -- Split in pages of length Max_Per_Page (except the last one)
  --
  -- Tags:
  -- NEXT           The href to the next page.
  -- PREVIOUS       The href to the previous page.
  -- FIRST          The href to the first page.
  -- LAST           The href to the last page.
  -- PAGE_INDEX     Position of the current page in the INDEXES_V vector
  --                Note that for this splitter, this is also the page number.
  -- HREFS_V        A vector tag containing a set of href to pages.
  -- INDEXES_V      A vector tag (synchronized with HREFS_V) containing the
  --                page numbers for the hrefs.
  --
  -- HREFS_V and INDEXES_V can be used to create an index to the generated
  -- pages.

  type Splitter (Max_Per_Page : Positive)
    is new Split_Pages.Splitter with private;

  function Get_Page_Ranges
    (This : in Splitter;
     Table : in Templates.Translate_Set)
    return Ranges_Table;

  function Get_Translations
    (This : in Splitter;
     Page : in Positive;
     URIs : in URI_Table;
     Ranges : in Ranges_Table)
    return Templates.Translate_Set;

private

```

```
-- implementation removed  
end AWS.Services.Split_Pages.Uniform;
```

B.58 AWS.Services.Split_Pages.Uniform.Alpha

```

-----
--                                     Ada Web Server                                     --
--                                                                                     --
--                                     Copyright (C) 2004                             --
--                                     ACT-Europe                                       --
--                                                                                     --
--  This library is free software; you can redistribute it and/or modify             --
--  it under the terms of the GNU General Public License as published by             --
--  the Free Software Foundation; either version 2 of the License, or (at           --
--  your option) any later version.                                                  --
--                                                                                     --
--  This library is distributed in the hope that it will be useful, but              --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                     --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU              --
--  General Public License for more details.                                         --
--                                                                                     --
--  You should have received a copy of the GNU General Public License               --
--  along with this library; if not, write to the Free Software Foundation,         --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                 --
--                                                                                     --
--  As a special exception, if other files instantiate generics from this           --
--  unit, or you link this unit with other files to produce an executable,         --
--  this unit does not by itself cause the resulting executable to be              --
--  covered by the GNU General Public License. This exception does not             --
--  however invalidate any other reasons why the executable file might be          --
--  covered by the GNU Public License.                                              --
-----

package AWS.Services.Split_Pages.Uniform.Alpha is

  -- Same as the uniform splitter, but builds in addition an alphabetical
  -- secondary index from a key field.
  -- For the references from the index to work, each line of the @TABLE@
  -- statement must include the following:
  --   <a name="@_TABLE_LINE_@">
  -- The alphabetical index will include one entry for empty fields, one
  -- entry for all fields that start with a digit, and one entry for each
  -- different initial letter.
  -- Note that leading spaces in the key field are ignored; this means that a
  -- key field containing only spaces is treated as an empty field.
  -- The key field is set by calling Set_Key. If no key is defined, or no
  -- corresponding association is found in Table, or the association is not a
  -- vector, Splitter_Error is raised.
  -- The key field must be sorted, and all values must be empty or start with
  -- a digit or letter (case ignored). Otherwise, Splitter_Error is raised.
  --
  -- Tags (in addition to those of the uniform splitter):
  -- S_HREFS_V      A vector tag containing a set of href to pages in the form
  --                <page>#<line>.
  -- S_INDEXES_V    A vector tag (synchronized with S_HREFS_V) containing
  --                "<>", "0..9", and the letters 'A' .. 'Z'
  --
  -- HREFS_V and INDEXES_V can be used to create an index to the generated
  -- pages. S_HREFS_V and S_INDEXES_V can be used to create a secondary
  -- alphabetical index that points directly to the corresponding element.

  type Splitter is new Uniform.Splitter with private;

  function Get_Page_Ranges
    (This : in Splitter;
     Table : in Templates.Translate_Set)
    return Ranges_Table;

```

```
function Get_Translations
  (This      : in Splitter;
   Page      : in Positive;
   URIs      : in URI_Table;
   Ranges    : in Ranges_Table)
  return Templates.Translate_Set;

procedure Set_Key (This : in out Splitter; Key : in String);
-- Set the key field, this is the name of the vector association in the
-- translate_set that will be used to create the index.

private
  -- implementation removed
end AWS.Services.Split_Pages.Uniform.Alpha;
```

B.59 AWS.Services.Split_Pages.Uniform.Overlapping

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2004                               --
--                                     ACT-Europe                                         --
--                                     -----                                         --
--  This library is free software; you can redistribute it and/or modify               --
--  it under the terms of the GNU General Public License as published by               --
--  the Free Software Foundation; either version 2 of the License, or (at            --
--  your option) any later version.                                                    --
--                                                                                      --
--  This library is distributed in the hope that it will be useful, but                --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                       --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                --
--  General Public License for more details.                                          --
--                                                                                      --
--  You should have received a copy of the GNU General Public License                  --
--  along with this library; if not, write to the Free Software Foundation,          --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                   --
--                                                                                      --
--  As a special exception, if other files instantiate generics from this             --
--  unit, or you link this unit with other files to produce an executable,           --
--  this unit does not by itself cause the resulting executable to be                --
--  covered by the GNU General Public License. This exception does not               --
--  however invalidate any other reasons why the executable file might be            --
--  covered by the GNU Public License.                                                --
-----

package AWS.Services.Split_Pages.Uniform.Overlapping is

  -- Same as the uniform splitter, but pages (except the first one)
  -- repeat Overlap lines from the previous page in addition to the
  -- Max_Per_Page lines
  --
  -- Tags:
  -- Same as the Uniform splitter

  type Splitter
    (Max_Per_Page : Positive;
     Overlap      : Natural) is new Uniform.Splitter with private;

  function Get_Page_Ranges
    (This : in Splitter;
     Table : in Templates.Translate_Set)
    return Ranges_Table;

private
  -- implementation removed
end AWS.Services.Split_Pages.Uniform.Overlapping;

```

B.60 AWS.Services.Transient_Pages

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2003-2004                             --
--                                     ACT-Europe                                           --
--                                     -----
--  This library is free software; you can redistribute it and/or modify
--  it under the terms of the GNU General Public License as published by
--  the Free Software Foundation; either version 2 of the License, or (at
--  your option) any later version.
--
--  This library is distributed in the hope that it will be useful, but
--  WITHOUT ANY WARRANTY; without even the implied warranty of
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
--  General Public License for more details.
--
--  You should have received a copy of the GNU General Public License
--  along with this library; if not, write to the Free Software Foundation,
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
--
--  As a special exception, if other files instantiate generics from this
--  unit, or you link this unit with other files to produce an executable,
--  this unit does not by itself cause the resulting executable to be
--  covered by the GNU General Public License. This exception does not
--  however invalidate any other reasons why the executable file might be
--  covered by the GNU Public License.
-----

with AWS.Default;
with AWS.Resources.Streams;

package AWS.Services.Transient_Pages is

  function Get_URI return String;
  -- Create a unique URI, must be used to register a transient web page

  procedure Register
    (URI      : in String;
     Resource : in AWS.Resources.Streams.Stream_Access;
     Lifetime : in Duration := Default.Transient_Lifetime);
  -- Register a new transient page, this page will be deleted after Lifetime
  -- seconds.

  function Get (URI : in String) return AWS.Resources.Streams.Stream_Access;
  -- Returns the stream access for the URI or null if this URI has not been
  -- registered.

private
  -- implementation removed
end AWS.Services.Transient_Pages;

```

B.61 AWS.Session

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2005                             --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                  --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                  --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

--  This is the API to handle session data for each client connected.

with Ada.Calendar;

package AWS.Session is

  type Id is private;

  No_Session : constant Id;

  function Create return Id;
  --  Create a new uniq Session Id

  procedure Delete (SID : in Id);
  --  Delete session, does nothing if SID does not exists.
  --  In most cases, the client browser will still send the cookie identifying
  --  the session on its next request. In such a case, the function
  --  AWS.Status.Timed_Out will return True, same as when the session was
  --  deleted automatically by AWS when it expired.
  --  The recommended practice is therefore to call
  --  AWS.Response.Set.Clear_Session when you send a response to the customer
  --  after deleting the session, so that the cookie is not sent again.

  function Image (SID : in Id) return String;
  pragma Inline (Image);
  --  Return ID image

  function Value (SID : in String) return Id;
  pragma Inline (Value);
  --  Build an ID from a String, returns No_Session if SID is not recongnized
  --  as an AWS session ID.

  function Exist (SID : in Id) return Boolean;
  --  Returns True if SID exist

```

```

procedure Touch (SID : in Id);
-- Update to current time the timestamp associated with SID. Does nothing
-- if SID does not exists.

procedure Set
  (SID   : in Id;
   Key   : in String;
   Value : in String);
-- Set key/pair value for the SID

procedure Set
  (SID   : in Id;
   Key   : in String;
   Value : in Integer);
-- Set key/pair value for the SID

procedure Set
  (SID   : in Id;
   Key   : in String;
   Value : in Float);
-- Set key/pair value for the SID

procedure Set
  (SID   : in Id;
   Key   : in String;
   Value : in Boolean);
-- Set key/pair value for the SID

function Get (SID : in Id; Key : in String) return String;
pragma Inline (Get);
-- Returns the Value for Key in the session SID or the empty string if
-- key does not exist.

function Get (SID : in Id; Key : in String) return Integer;
pragma Inline (Get);
-- Returns the Value for Key in the session SID or the integer value 0 if
-- key does not exist or is not an integer.

function Get (SID : in Id; Key : in String) return Float;
pragma Inline (Get);
-- Returns the Value for Key in the session SID or the float value 0.0 if
-- key does not exist or is not a float.

function Get (SID : in Id; Key : in String) return Boolean;
pragma Inline (Get);
-- Returns the Value for Key in the session SID or the boolean False if
-- key does not exist or is not a boolean.

generic
  type Data is private;
  Null_Data : Data;
package Generic_Data is

  procedure Set
    (SID   : in Id;
     Key   : in String;
     Value : in Data);
  -- Set key/pair value for the SID

  function Get (SID : in Id; Key : in String) return Data;
  pragma Inline (Get);
  -- Returns the Value for Key in the session SID or Null_Data if
  -- key does not exist.

```



```

end Generic_Data;

procedure Remove
  (SID : in Id;
   Key : in String);
-- Removes Key from the specified session

function Exist
  (SID : in Id;
   Key : in String)
  return Boolean;
-- Returns True if Key exist in session SID

procedure Clear;
-- Removes all sessions data

-----
-- Iterators --
-----

generic
  with procedure Action
    (N          : in Positive;
     SID        : in Id;
     Time_Stamp : in Ada.Calendar.Time;
     Quit       : in out Boolean);
procedure For_Every_Session;
-- Iterator which call Action for every active session. N is the SID
-- order. Time_Stamp is the time when SID was updated for the last
-- time. Quit is set to False by default, it is possible to control the
-- iterator termination by setting its value to True. Note that in the
-- Action procedure it is possible to use routines that read session's
-- data (Get, Exist) but any routines which modify the data will block
-- (i.e. Touch, Set, Remove, Delete will dead lock).

generic
  with procedure Action
    (N          : in Positive;
     Key, Value : in String;
     Quit       : in out Boolean);
procedure For_Every_Session_Data (SID : in Id);
-- Iterator which returns all the key/value pair defined for session SID.
-- Quit is set to False by default, it is possible to control the iterator
-- termination by setting its value to True. Note that in the Action
-- procedure it is possible to use routines that read session's data (Get,
-- Exist) but any routines which modify the data will block (i.e. Touch,
-- Set, Remove, Delete will dead lock).

-----
-- Lifetime --
-----

procedure Set_Lifetime (Seconds : in Duration);
-- Set the lifetime for session data. At the point a session is deleted,
-- reusing the session ID makes AWS.Status.Session_Timed_Out return True.

function Get_Lifetime return Duration;
-- Get current session lifetime for session data

function Has_Expired (SID : in Id) return Boolean;
-- Returns true if SID should be considered as expired (ie there hasn't
-- been any transaction on it since Get_Lifetime seconds. Such a session
-- should be deleted. Calling this function is mostly internal to AWS, and
-- sessions are deleted automatically when they expire.

```

```

-----
-- Session Callback --
-----

type Callback is access procedure (SID : in Id);
-- Callback procedure called when a session is deleted from the server

procedure Set_Callback (Callback : in Session.Callback);
-- Set the callback procedure to call when a session is deleted from the
-- server. If Callback is Null the session's callback will be removed.

-----
-- Session IO --
-----

procedure Save (File_Name : in String);
-- Save all sessions data into File_Name

procedure Load (File_Name : in String);
-- Restore all sessions data from File_Name

private
-- implementation removed
end AWS.Session;

```

B.62 AWS.SMTP

```

-----
--                               Ada Web Server                               --
--                               S M T P - Simple Mail Transfer Protocol         --
--                               Copyright (C) 2000-2004                         --
--                               ACT-Europe                                     --
--                               -----                                     --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                     --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                     --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                     --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                        --
-----

--   This library implement the Simple Mail Transfer Protocol. Only part of the
--   RFC 821 is covered. There is no support to send a message to a console for
--   example.

with Ada.Strings.Unbounded;

with AWS.Net;

package AWS.SMTP is

  Server_Error : exception;
  --   Raised when an unrecoverable error is found.

  Reply_Code_Error : exception;
  --   Raised when a reply code error is not known.

  Default_SMTP_Port : constant := 25;

  type Receiver is private;
  --   The receiver part (i.e. a server) of SMTP messages as defined in
  --   RFC 821. This is the SMTP server

  -----
  -- Reply_Code --
  -----

  type Reply_Code is range 200 .. 554;

  Service_Ready      : constant Reply_Code := 220;
  Service_Closing    : constant Reply_Code := 221;
  Requested_Action_Ok : constant Reply_Code := 250;
  Start_Mail_Input   : constant Reply_Code := 354;
  Syntax_Error       : constant Reply_Code := 500;

```

```

function Image (R : in Reply_Code) return String;
-- Returns the reply code as a string. Raises Reply_Code_Error if R is
-- not a valid reply code.

function Name (R : in Reply_Code) return String;
-- Returns the reply code reason string. Raises Reply_Code_Error if R is
-- not a valid reply code.

function Message (R : in Reply_Code) return String;
-- This returns the value: Image (R) & ' ' & Name (R)

-----
-- Status --
-----

type Status is private;

function Is_Ok (Status : in SMTP.Status) return Boolean;
pragma Inline (Is_Ok);
-- Return True is status if Ok (no problem) or false if a problem has been
-- detected. This is not an error (in that case Error is raised) but a
-- warning because something wrong (but not unrecoverable) has happen.

function Status_Message (Status : in SMTP.Status) return String;
-- If Is_Ok is False, this function return the reason of the problem. The
-- return message is the error message as reported by the server.

function Status_Code (Status : in SMTP.Status) return Reply_Code;
pragma Inline (Status_Code);
-- Returns the code replied by the server.

procedure Clear (Status : in out SMTP.Status);
pragma Inline (Clear);
-- Clear Status value. Code is set to Requested_Action_Ok and message
-- string to null.

-----
-- E-Mail_Data --
-----

type E-Mail_Data is private;

type Address_Mode is (Full, Name, Address);

function Image
  (E-Mail : in E-Mail_Data;
   Mode   : in Address_Mode := Full) return String;
-- Returns E-Mail only (Mode = Address), recipient name only (Mode = Name)
-- or Name and e-mail (Mode = Full).

function E-Mail (Name : in String; Address : in String)
  return E-Mail_Data;
-- Returns an e-mail address

function Parse (E-Mail : in String) return E-Mail_Data;
-- Parse an e-mail with format "Name <address>" or "address (Name)"
-- and Returns the corresponding E-Mail_Data. Raises Constraint_Error
-- if E-Mail can't be parsed.

type Recipients is array (Positive range <>) of E-Mail_Data;

private
-- implementation removed
end AWS.SMTP;

```


B.63 AWS.SMTP.Client

```

-----
--                               Ada Web Server                               --
--                               S M T P - Simple Mail Transfer Protocol        --
--                               Copyright (C) 2000-2005                       --
--                               AdaCore                                         --
--                               -----                                     --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but       --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                       --
-----

--
--   This unit implement API to send mail message. It is possible to send
--   simple mail [RFC 821] and mail with MIME attachments [RFC 2045 & 2049].
--
--   How to send a mail:
--
--   1) Initialize a Server to send the messages.
--
--       Wanadoo : SMTP.Server := SMTP.Client.Initialize ("smtp.wanadoo.fr");
--
--   2) Send a message via this server.
--
--       Result : SMTP.Status;
--
--       SMTP.Client.Send
--       (Server => Wanadoo,
--        From   => SMTP.E-Mail ("Pascal Obry", "p.obry@wanadoo.fr"),
--        To     => SMTP.E-Mail
--              ("Dmitriy Anisimkov", "anisimkov@omsynet.ru"),
--        Subject => "Latest Ada news",
--        Message => "now Ada can send SMTP mail!",
--        Status => Result);

with Ada.Strings.Unbounded;

package AWS.SMTP.Client is

  use Ada.Strings.Unbounded;

  function Initialize
    (Server_Name : in String;
     Port        : in Positive := Default_SMTP_Port)
    return Receiver;

```

```

-- Create a Server composed of the Name and the Port (default SMTP port
-- is 25), this server will be used to send SMTP message.

procedure Send
  (Server : in      Receiver;
   From   : in      E-Mail_Data;
   To     : in      E-Mail_Data;
   Subject : in      String;
   Message : in      String;
   Status : out SMTP.Status);
-- Send a message via Server. The mail is a simple message composed of a
-- subject and a text message body. Raise Server_Error in case of an
-- unrecoverable error (e.g. can't contact the server).

type Attachment is private;
-- This is an attachment object, either a File or a Base64 content.

function File (Filename : in String) return Attachment;
-- Returns a file attachment. Filename point to a file on the file system.

function Base64_Data (Name, Content : in String) return Attachment;
-- Returns a base64 attachment. Content is the Base64 encoded
-- data. Attachment is named Name. This is a way to send a file attachment
-- from in-memory data.

type Attachment_Set is array (Positive range <>) of Attachment;
-- A set of file attachments.

procedure Send
  (Server : in      Receiver;
   From   : in      E-Mail_Data;
   To     : in      E-Mail_Data;
   Subject : in      String;
   Message : in      String;
   Attachments : in Attachment_Set;
   Status : out SMTP.Status);
-- Send a message via Server. The mail is a MIME message composed of a
-- subject, a message and a set of files MIME encoded. Raise Server_Error
-- in case of an unrecoverable error (e.g. can't contact the server).
-- Raises Constraint_Error is a file attachment cannot be opened.

type Message_File is new String;

procedure Send
  (Server : in      Receiver;
   From   : in      E-Mail_Data;
   To     : in      E-Mail_Data;
   Subject : in      String;
   Filename : in      Message_File;
   Status : out SMTP.Status);
-- Send filename content via Server. The mail is a message composed of a
-- subject and a message body coming from a file. Raises Server_Error in
-- case of an unrecoverable error (e.g. can't contact the server). Raises
-- Constraint_Error if Filename cannot be opened.

--
-- Extended interfaces to send a message to many recipients.
--

procedure Send
  (Server : in      Receiver;
   From   : in      E-Mail_Data;
   To     : in      Recipients;
   Subject : in      String;
   Message : in      String;

```

```

    Status : out SMTP.Status);
-- Send a message via Server. The mail is a simple message composed of a
-- subject and a text message body. Raise Server_Error in case of an
-- unrecoverable error (e.g. can't contact the server).

procedure Send
  (Server      : in    Receiver;
   From        : in    E-Mail_Data;
   To          : in    Recipients;
   Subject     : in    String;
   Message     : in    String;
   Attachments : in    Attachment_Set;
   Status      : out SMTP.Status);
-- Send a message via Server. The mail is a MIME message composed of a
-- subject, a message and a set of files MIME encoded. Raise Server_Error
-- in case of an unrecoverable error (e.g. can't contact the server).
-- Raises Constraint_Error is a file attachment cannot be opened.

private
  -- implementation removed
end AWS.SMTP.Client;
```


B.64 AWS.Status

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2006                             --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                  --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

--  This package is used to keep the HTTP protocol status. Client can then
--  request the status for various values like the requested URI, the
--  Content_Length and the Session ID for example.

with Ada.Calendar;
with Ada.Streams;
with Ada.Strings.Unbounded;

with AWS.Attachments;
with AWS.Headers;
with AWS.Messages;
with AWS.Net;
with AWS.Parameters;
with AWS.Session;
with AWS.URL;
with AWS.Utils;

package AWS.Status is

  type Data is private;

  type Request_Method is (GET, HEAD, POST, PUT);

  type Authorization_Type is (None, Basic, Digest);

  -----
  -- Request-Line --
  -----

  function Method          (D : in Data) return Request_Method;
  pragma Inline (Method);
  -- Returns the request method

  function URI             (D : in Data) return String;

```

```

pragma Inline (URI);
-- Returns the requested resource

function URI (D : in Data) return URL.Object;
pragma Inline (URI);
-- As above but return an URL object

function Parameters (D : in Data) return Parameters.List;
pragma Inline (Parameters);
-- Returns the list of parameters for the request. This list can be empty
-- if there was no form or URL parameters.

function Parameter (D : in Data; Name : in String; N : in Positive := 1) return String;
pragma Inline (Parameter);

function HTTP_Version (D : in Data) return String;
pragma Inline (HTTP_Version);
-- Returns the HTTP version used by the client

function Request_Time (D : in Data) return Ada.Calendar.Time;
pragma Inline (Request_Time);
-- Returns the time of the request.

-----
-- Header --
-----

function Header (D : in Data) return Headers.List;
pragma Inline (Header);
-- Returns the list of header lines for the request

function Accept-Encoding (D : in Data) return String;
pragma Inline (Accept-Encoding);
-- Get the value for "Accept-Encoding:" header

function Connection (D : in Data) return String;
pragma Inline (Connection);
-- Get the value for "Connection:" parameter

function Content_Length (D : in Data) return Natural;
pragma Inline (Content_Length);
-- Get the value for "Content-Length:" header, this is the number of
-- bytes in the message body.

function Content_Type (D : in Data) return String;
pragma Inline (Content_Type);
-- Get value for "Content-Type:" header

function Host (D : in Data) return String;
pragma Inline (Host);
-- Get value for "Host:" header

function If_Modified_Since (D : in Data) return String;
pragma Inline (If_Modified_Since);
-- Get value for "If-Modified-Since:" header

function Keep_Alive (D : in Data) return Boolean;
pragma Inline (Keep_Alive);
-- Returns the flag if the current HTTP connection is keep-alive

function User_Agent (D : in Data) return String;
pragma Inline (User_Agent);
-- Get value for "User-Agent:" header

```

```

function Referer          (D : in Data) return String;
pragma Inline (Referer);
-- Get value for "Referer:" header

function Is_Supported
  (D          : in Data;
   Encoding   : in Messages.Content_Encoding)
  return Boolean;
-- Returns True if the content encoding scheme is sported by the client

function Preferred_Coding (D : in Data) return Messages.Content_Encoding;
-- Returns supported by AWS coding preferred by client from the
-- Accept-Coding header.

-----
-- Connection --
-----

function Peername          (D : in Data) return String;
pragma Inline (Peername);
-- Returns the name of the peer (the name of the client computer)

function Socket            (D : in Data) return Net.Socket_Type'Class;
pragma Inline (Socket);
-- Returns the socket used to transfert data between the client and
-- server.

function Socket            (D : in Data) return Net.Socket_Access;
pragma Inline (Socket);
-- Returns the socket used to transfert data between the client and
-- server. Use Socket_Access to avoid memory allocation if we would need
-- socket access further.

-----
-- Data --
-----

function Multipart_Boundary (D : in Data) return String;
pragma Inline (Multipart_Boundary);
-- Get value for the boundary part in "Content-Type: ...; boundary=..."
-- parameter. This is a string that will be used to separate each chunk of
-- data in a multipart message.

subtype Stream_Element_Array is Ada.Streams.Stream_Element_Array;

function Binary_Data (D : in Data) return Stream_Element_Array;
pragma Inline (Binary_Data);
-- Returns the binary data message content.
-- Note that only the root part of a multipart/related message is returned.

-----
-- Attachments --
-----

function Attachments       (D : in Data) return AWS.Attachments.List;
pragma Inline (Attachments);
-- Returns the list of Attachments for the request

-----
-- Session --
-----

function Has_Session       (D : in Data) return Boolean;
pragma Inline (Has_Session);
-- Returns true if a session ID has been received

```

```

function Session              (D : in Data) return Session.Id;
pragma Inline (Session);
-- Returns the Session ID for the request. Raises Constraint_Error if
-- server's session support not activated.

function Session_Created      (D : in Data) return Boolean;
-- Returns True if session was just created and is going to be sent to
-- client.

function Session_Timed_Out    (D : in Data) return Boolean;
-- Returns True if a previous session was timeout (even if a new session
-- has been created).

-----
-- SOAP --
-----

function Is_SOAP              (D : in Data) return Boolean;
pragma Inline (Is_SOAP);
-- Returns True if it is a SOAP request. In this case SOAPAction return
-- the SOAPAction header and Payload returns the XML SOAP Payload message.

function SOAPAction           (D : in Data) return String;
pragma Inline (SOAPAction);
-- Get value for "SOAPAction:" parameter. This is a standard header to
-- support SOAP over HTTP protocol.

function Payload               (D : in Data) return String;
pragma Inline (Payload);
-- Returns the XML Payload message. XML payload is the actual SOAP
-- request. This is the root part of multipart/related SOAP message.

-----
-- HTTPS --
-----

function Check_Digest
  (D          : in Data;
   Password   : in String)
  return Messages.Status_Code;
-- This function is used by the digest authentication to check if the
-- client password and authentication parameters are correct.
-- The password is not transferred between the client and the server,
-- the server check that the client knows the right password using the
-- MD5 checksum.
-- Returns Messages.S200 in case of successful authentication,
-- Messages.S400 in case of wrong authentication request
-- (RFC 2617 3.2.2, 3.2.2.5),
-- and Messages.S401 in case of authentication error.

function Check_Digest (D : in Data; Password : in String) return Boolean;
-- The same as above, but do not distinguish wrong requests and
-- authentication errors.

function Authorization_Mode    (D : in Data) return Authorization_Type;
pragma Inline (Authorization_Mode);
-- Returns the type of the "Authorization:" parameter

function Authorization_Name    (D : in Data) return String;
pragma Inline (Authorization_Name);
-- Returns "username" value in the "Authorization:" parameter

function Authorization_URI     (D : in Data) return String;
pragma Inline (Authorization_URI);

```

```

-- Returns "uri" value in the "Authorization:" parameter
-- Note, it could differ from HTTP URI field, for example Mozilla browser
-- places http parameters to the authorization uri field.

function Authorization_Password (D : in Data) return String;
pragma Inline (Authorization_Password);
-- Returns "password" value in the "Authorization:" parameter

function Authorization_Realm (D : in Data) return String;
pragma Inline (Authorization_Realm);
-- Returns "realm" value in the "Authorization:" parameter

function Authorization_Nonce (D : in Data) return String;
pragma Inline (Authorization_Nonce);
-- Returns "nonce" value in the "Authorization:" parameter

function Authorization_NC (D : in Data) return String;
pragma Inline (Authorization_NC);
-- Returns "nc" value in the "Authorization:" parameter

function Authorization_CNonce (D : in Data) return String;
pragma Inline (Authorization_CNonce);
-- Returns "cnonce" value in the "Authorization:" parameter

function Authorization_QOP (D : in Data) return String;
pragma Inline (Authorization_QOP);
-- Returns "qop" value in the "Authorization:" parameter

function Authorization_Response (D : in Data) return String;
pragma Inline (Authorization_Response);
-- Returns "response" value in the "Authorization:" parameter

function Authorization_Tail (D : in Data) return String;
pragma Inline (Authorization_Tail);
-- Returns precalculated part of digest composed of
-- Nonce, NC, CNonce, QOP, Method, URI authorization fields.
-- To build a full authorization response you can use:
--
-- MD5.Digest
-- (MD5.Digest (Username & ':' & Realm & ':' & Password)
-- & Authorization_Tail);
--
-- This method can be used to avoid sending a password over the network.

private
-- implementation removed
end AWS.Status;
```

B.65 AWS.Templates

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2001                       --
--                               ACT-Europe                                     --
--                               -----                                     --
--   This library is free software; you can redistribute it and/or modify     --
--   it under the terms of the GNU General Public License as published by     --
--   the Free Software Foundation; either version 2 of the License, or (at   --
--   your option) any later version.                                          --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of              --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU       --
--   General Public License for more details.                                --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,  --
--   this unit does not by itself cause the resulting executable to be       --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be   --
--   covered by the GNU Public License.                                       --
-----

with Templates_Parser;

package AWS.Templates renames Templates_Parser;

```

B.66 AWS.Translator

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2000-2003                             --
--                                     ACT-Europe                                           --
--                                     -----
--  This library is free software; you can redistribute it and/or modify
--  it under the terms of the GNU General Public License as published by
--  the Free Software Foundation; either version 2 of the License, or (at
--  your option) any later version.
--
--  This library is distributed in the hope that it will be useful, but
--  WITHOUT ANY WARRANTY; without even the implied warranty of
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
--  General Public License for more details.
--
--  You should have received a copy of the GNU General Public License
--  along with this library; if not, write to the Free Software Foundation,
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
--
--  As a special exception, if other files instantiate generics from this
--  unit, or you link this unit with other files to produce an executable,
--  this unit does not by itself cause the resulting executable to be
--  covered by the GNU General Public License. This exception does not
--  however invalidate any other reasons why the executable file might be
--  covered by the GNU Public License.
-----

with Ada.Streams;
with Ada.Strings.Unbounded;

with AWS.Resources.Streams.Memory.ZLib;
with AWS.Utils;

package AWS.Translator is

  package ZL renames AWS.Resources.Streams.Memory.ZLib;

  -----
  -- Base64 --
  -----

  function Base64_Encode
    (Data : in Ada.Streams.Stream_Element_Array)
    return String;
  -- Encode Data using the base64 algorithm

  function Base64_Encode (Data : in String) return String;
  -- Same as above but takes a string as input

  function Base64_Decode
    (B64_Data : in String)
    return Ada.Streams.Stream_Element_Array;
  -- Decode B64_Data using the base64 algorithm

  -----
  -- QP --
  -----

  function QP_Decode
    (QP_Data : in String)
    return String;

```

```

-- Decode QP_Data using the Quoted Printable algorithm

-----
-- String to Stream_Element_Array --
-----

function To_String
  (Data : in Ada.Streams.Stream_Element_Array)
  return String;
pragma Inline (To_String);
-- Convert a Stream_Element_Array to a string. Note that as this routine
-- returns a String it should not be used with large array as this could
-- break the stack size limit. Use the routine below for large array.

function To_Stream_Element_Array
  (Data : in String)
  return Ada.Streams.Stream_Element_Array;
pragma Inline (To_Stream_Element_Array);
-- Convert a String to a Stream_Element_Array.

function To_Unbounded_String
  (Data : in Ada.Streams.Stream_Element_Array)
  return Ada.Strings.Unbounded.Unbounded_String;
-- Convert a Stream_Element_Array to an Unbounded_String.

-----
-- Compress/Decompress --
-----

subtype Compression_Level is ZL.Compression_Level;

Default_Compression : constant Compression_Level := ZL.Default_Compression;

function Compress
  (Data : in Ada.Streams.Stream_Element_Array;
   Level : in Compression_Level := Default_Compression;
   Header : in ZL.Header_Type := ZL.Default_Header)
  return Utils.Stream_Element_Array_Access;
-- Returns Data compressed with a standard deflate algorithm based on the
-- zlib library. The result is dynamically allocated and must be
-- explicitly freed.

function Decompress
  (Data : in Ada.Streams.Stream_Element_Array;
   Header : in ZL.Header_Type := ZL.Default_Header)
  return Utils.Stream_Element_Array_Access;
-- Returns Data decompressed based on the zlib library. The results is
-- dynamically allocated and must be explicitly freed.

end AWS.Translator;

```


B.67 AWS.URL

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2004                       --
--                               ACT-Europe                                    --
--                               -----                                    --
--   This library is free software; you can redistribute it and/or modify     --
--   it under the terms of the GNU General Public License as published by     --
--   the Free Software Foundation; either version 2 of the License, or (at   --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but      --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this    --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not      --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                       --
-----

with Ada.Strings.Maps;
with Ada.Strings.Unbounded;

package AWS.URL is

  use Ada;

  -- The general URL form as described in RFC2616 is:
  --
  -- http_URL = "http:" "/" host [ ":" port ] [ abs_path [ "?" query ] ]
  --
  -- Note also that there are different RFC describing URL like the 2616 and
  -- 1738 but they use different terminologies. Here we try to follow the
  -- names used in RFC2616 but we have implemented some extensions at the
  -- end of this package. For example the way Path and File are separated or
  -- the handling of user/password which is explicitly not allowed in the
  -- RFC but are used and supported in many browsers. Here are the extended
  -- URL supported:
  --
  -- http://username:password@www.here.com:80/dir1/dir2/xyz.html?p=8&x=doh
  -- |           |           | |           |           |
  -- protocol    host port path      file      parameters
  --
  --                                     <-- pathname -->

  type Object is private;

  URL_Error : exception;

  Default_HTTP_Port  : constant := 80;
  Default_HTTPS_Port : constant := 443;

  function Parse
    (URL      : in String;
     pathname : in String;

```

```

    Check_Validity : in Boolean := True;
    Normalize      : in Boolean := False)
    return Object;
-- Parse an URL and return an Object representing this URL. It is then
-- possible to extract each part of the URL with the services bellow.
-- Raises URL_Error if Check_Validity is true and the URL reference a
-- resource above the web root directory.

procedure Normalize (URL : in out Object);
-- Removes all occurrences to parent directory ".." and current directory
-- ".". Raises URL_Error if the URL reference a resource above the Web
-- root directory.

function Is_Valid (URL : in Object) return Boolean;
-- Returns True if the URL is valid (does not reference directory above
-- the Web root).

function URL (URL : in Object) return String;
-- Returns full URL string, this can be different to the URL passed if it
-- has been normalized.

function Protocol_Name (URL : in Object) return String;
-- Returns "http" or "https" depending on the protocol used by URL.

function Host (URL : in Object) return String;
-- Returns the hostname

function Port (URL : in Object) return Positive;
-- Returns the port as a positive

function Port (URL : in Object) return String;
-- Returns the port as a string

function Abs_Path
  (URL      : in Object;
   Encode   : in Boolean := False)
  return String;
-- Returns the absolute path. This is the complete resource reference
-- without the query part.

function Query
  (URL      : in Object;
   Encode   : in Boolean := False)
  return String;
-- Returns the Query part of the URL or the empty string if none was
-- specified. Note that character '?' is not part of the Query and is
-- therefore not returned.

--
-- Below are extended API not part of the RFC 2616 URL specification
--

function User (URL : in Object) return String;
-- Returns user name part of the URL. Returns the empty string if user was
-- not specified.

function Password (URL : in Object) return String;
-- Returns user's password part of the URL. Returns the empty string if
-- password was not specified.

function Server_Name (URL : in Object) return String renames Host;

function Security (URL : in Object) return Boolean;
-- Returns True if it is a Secure HTTP (HTTPS) URL

```

```

function Path (URL : in Object; Encode : in Boolean := False) return String;
-- Returns the Path (including the leading slash). If Encode is True then
-- the URL will be encoded using the Encode routine.

function File (URL : in Object; Encode : in Boolean := False) return String;
-- Returns the File. If Encode is True then the URL will be encoded using
-- the Encode routine. Not that by File here we mean the latest part of
-- the URL, it could be a real file or a directory into the filesystem.
-- Parent and current directories are part of the path.

function Parameters
  (URL      : in Object;
   Encode   : in Boolean := False)
  return String;
-- Returns the Parameters (including the starting ? character). If Encode
-- is True then the URL will be encoded using the Encode routine.

function Pathname
  (URL      : in Object;
   Encode   : in Boolean := False)
  return String;
renames Abs_Path;

function Pathname_And_Parameters
  (URL      : in Object;
   Encode   : in Boolean := False)
  return String;
-- Returns the pathname and the parameters. This is equivalent to:
-- Pathname & Parameters.

--
-- URL Encoding and Decoding
--

Default-Encoding_Set : constant Strings.Maps.Character_Set;

function Encode
  (Str      : in String;
   Encoding_Set : in Strings.Maps.Character_Set := Default-Encoding_Set)
  return String;
-- Encode Str into a URL-safe form. Many characters are forbidden into an
-- URL and needs to be encoded. A character is encoded by %XY where XY is
-- the character's ASCII hexadecimal code. For example a space is encoded
-- as %20.

function Decode (Str : in String) return String;
-- This is the opposite of Encode above

private
-- implementation removed
end AWS.URL;

```

B.68 SOAP

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2000-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--  This library is free software; you can redistribute it and/or modify      --
--  it under the terms of the GNU General Public License as published by      --
--  the Free Software Foundation; either version 2 of the License, or (at    --
--  your option) any later version.                                           --
--                                                                              --
--  This library is distributed in the hope that it will be useful, but      --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU       --
--  General Public License for more details.                                  --
--                                                                              --
--  You should have received a copy of the GNU General Public License         --
--  along with this library; if not, write to the Free Software Foundation,  --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.           --
--                                                                              --
--  As a special exception, if other files instantiate generics from this     --
--  unit, or you link this unit with other files to produce an executable,    --
--  this unit does not by itself cause the resulting executable to be        --
--  covered by the GNU General Public License. This exception does not       --
--  however invalidate any other reasons why the executable file might be    --
--  covered by the GNU Public License.
-----

```

package SOAP is

```

--  This is the root package for the SOAP implementation. It supports
--  SOAP 1.1 specifications.

SOAP_Error : exception;
--  Will be raised when an error occurs in the SOAP implementation. The
--  exception message will described the problem.

Version : constant String := "1.4.0";
--  Version number for this implementation

No_SOAPAction : constant String := (1 => ASCII.Nul);
--  Value used to specify that there was no SOAPAction specified

```

end SOAP;

B.69 SOAP.Client

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2000-2004                             --
--                                     ACT-Europe                                           --
--                                     -----
--  This library is free software; you can redistribute it and/or modify
--  it under the terms of the GNU General Public License as published by
--  the Free Software Foundation; either version 2 of the License, or (at
--  your option) any later version.
--
--  This library is distributed in the hope that it will be useful, but
--  WITHOUT ANY WARRANTY; without even the implied warranty of
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
--  General Public License for more details.
--
--  You should have received a copy of the GNU General Public License
--  along with this library; if not, write to the Free Software Foundation,
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
--
--  As a special exception, if other files instantiate generics from this
--  unit, or you link this unit with other files to produce an executable,
--  this unit does not by itself cause the resulting executable to be
--  covered by the GNU General Public License. This exception does not
--  however invalidate any other reasons why the executable file might be
--  covered by the GNU Public License.
-----

with AWS.Client;
with SOAP.Message.Payload;
with SOAP.Message.Response;

package SOAP.Client is

  Not_Specified : String renames AWS.Client.No_Data;

  function Call
    (URL      : in String;
     P        : in Message.Payload.Object;
     SOAPAction : in String := No_SOAPAction;
     User     : in String   := Not_Specified;
     Pwd      : in String   := Not_Specified;
     Proxy    : in String   := Not_Specified;
     Proxy_User : in String := Not_Specified;
     Proxy_Pwd : in String   := Not_Specified;
     Timeouts : in AWS.Client.Timeouts_Values := AWS.Client.No_Timeout)
    return Message.Response.Object'Class;
  -- Send a SOAP HTTP request to URL address. The P is the Payload and
  -- SOAPAction is the required HTTP field. If it is not specified then the
  -- URI (URL resource) will be used for the SOAPAction field. The complete
  -- format is "URL & '#' & Procedure_Name" (Procedure_Name is retrieved
  -- from the Payload object.

  function Call
    (Connection : in AWS.Client.HTTP_Connection;
     SOAPAction : in String;
     P          : in Message.Payload.Object)
    return Message.Response.Object'Class;
  -- Idem as above, but use an already opened connection

end SOAP.Client;

```

B.70 SOAP.Dispatchers

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2003                           --
--                               ACT-Europe                                    --
--                               -----                                    --
--   This library is free software; you can redistribute it and/or modify    --
--   it under the terms of the GNU General Public License as published by    --
--   the Free Software Foundation; either version 2 of the License, or (at   --
--   your option) any later version.                                         --
--                                                                           --
--   This library is distributed in the hope that it will be useful, but     --
--   WITHOUT ANY WARRANTY; without even the implied warranty of             --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU      --
--   General Public License for more details.                                --
--                                                                           --
--   You should have received a copy of the GNU General Public License       --
--   along with this library; if not, write to the Free Software Foundation, --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.         --
--                                                                           --
--   As a special exception, if other files instantiate generics from this   --
--   unit, or you link this unit with other files to produce an executable,  --
--   this unit does not by itself cause the resulting executable to be      --
--   covered by the GNU General Public License. This exception does not     --
--   however invalidate any other reasons why the executable file might be   --
--   covered by the GNU Public License.                                     --
-----

-- Dispatcher for SOAP requests.

with AWS.Dispatchers;
with AWS.Response;
with AWS.Status;
with SOAP.Message.Payload;

package SOAP.Dispatchers is

  type Handler is abstract new AWS.Dispatchers.Handler with private;
  -- This dispatcher will send SOAP and HTTP requests to different routines

  type SOAP_Callback is
    access function (SOAPAction : in String;
                    Payload      : in Message.Payload.Object;
                    Request      : in AWS.Status.Data)
                    return AWS.Response.Data;
  -- This is the SOAP Server callback type. SOAPAction is the HTTP header
  -- SOAPAction value, Payload is the parsed XML payload, request is the
  -- HTTP request status.

  function Dispatch_SOAP
    (Dispatcher : in Handler;
     SOAPAction : in String;
     Payload     : in Message.Payload.Object;
     Request     : in AWS.Status.Data)
    return AWS.Response.Data is abstract;
  -- This dispatch function is called for SOAP requests

  function Dispatch_HTTP
    (Dispatcher : in Handler;
     Request     : in AWS.Status.Data)
    return AWS.Response.Data is abstract;
  -- This dispatch function is called for standard HTTP requests

```

```
private
  -- implementation removed
end SOAP.Dispatchers;
```

B.71 SOAP.Dispatchers.Callback

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2003                               --
--                                     ACT-Europe                                         --
--                                     -----                                         --
--  This library is free software; you can redistribute it and/or modify              --
--  it under the terms of the GNU General Public License as published by              --
--  the Free Software Foundation; either version 2 of the License, or (at           --
--  your option) any later version.                                                  --
--                                                                                      --
--  This library is distributed in the hope that it will be useful, but              --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                      --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU              --
--  General Public License for more details.                                         --
--                                                                                      --
--  You should have received a copy of the GNU General Public License                --
--  along with this library; if not, write to the Free Software Foundation,         --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                  --
--                                                                                      --
--  As a special exception, if other files instantiate generics from this            --
--  unit, or you link this unit with other files to produce an executable,          --
--  this unit does not by itself cause the resulting executable to be               --
--  covered by the GNU General Public License. This exception does not              --
--  however invalidate any other reasons why the executable file might be           --
--  covered by the GNU Public License.                                              --
-----

-- Dispatch on a Callback procedures.

package SOAP.Dispatchers.Callback is

  type Handler is new Dispatchers.Handler with private;
  -- This is a simple wrapper around standard callback procedure (access to
  -- function). It will be used to build dispatchers services and for the
  -- main server callback.

  function Create
    (HTTP_Callback : in AWS.Response.Callback;
     SOAP_Callback : in Dispatchers.SOAP_Callback)
    return Handler;
  -- Build a dispatcher for the specified callback.

private
  -- implementation removed
end SOAP.Dispatchers.Callback;

```


B.72 SOAP.Message

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2005                             --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                     --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU                  --
--  General Public License for more details.                                           --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                   --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                     --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                 --
-----

with Ada.Strings.Unbounded;

with SOAP.Name_Space;
with SOAP.Parameters;

package SOAP.Message is

  use Ada.Strings.Unbounded;

  type Object is tagged private;

  function XML_Image (M : in Object) return Unbounded_String;
  -- Returns the XML image for the wrapper and parameters. This is designed
  -- to be used by Payload and Response object.

  function Name_Space (M : in Object'Class) return SOAP.Name_Space.Object;
  -- Returns message Namespace

  function Wrapper_Name (M : in Object'class) return String;
  -- Returns wrapper name

  function Parameters (M : in Object'class) return SOAP.Parameters.List;
  -- Returns the parameter

  procedure Set_Name_Space
    (M : in out Object'Class;
     NS : in SOAP.Name_Space.Object);
  -- Set message's Namespace

  procedure Set_Wrapper_Name
    (M : in out Object'Class;
     Name : in String);
  -- Set message's wrapper name

```

```
procedure Set_Parameters
  (M      : in out Object'Class;
   P_Set : in      SOAP.Parameters.List);
-- Set message's parameters

private
-- implementation removed
end SOAP.Message;
```

B.73 SOAP.Message.XML

```

-----
--                                     Ada Web Server                                     --
--                                     Copyright (C) 2000-2005                             --
--                                     AdaCore                                             --
--                                     -----
--  This library is free software; you can redistribute it and/or modify
--  it under the terms of the GNU General Public License as published by
--  the Free Software Foundation; either version 2 of the License, or (at
--  your option) any later version.
--
--  This library is distributed in the hope that it will be useful, but
--  WITHOUT ANY WARRANTY; without even the implied warranty of
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
--  General Public License for more details.
--
--  You should have received a copy of the GNU General Public License
--  along with this library; if not, write to the Free Software Foundation,
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
--
--  As a special exception, if other files instantiate generics from this
--  unit, or you link this unit with other files to produce an executable,
--  this unit does not by itself cause the resulting executable to be
--  covered by the GNU General Public License. This exception does not
--  however invalidate any other reasons why the executable file might be
--  covered by the GNU Public License.
-----

with AWS.Client;

with SOAP.Message.Payload;
with SOAP.Message.Response;

package SOAP.Message.XML is

  function Load_Payload (XML : in String) return Message.Payload.Object;
  -- Build a Payload object by parsing the XML payload string

  function Load_Response
    (Connection : in AWS.Client.HTTP_Connection)
    return Message.Response.Object'Class;
  -- Build a Response object (either a standard response or an error
  -- response) by parsing the HTTP client connection output.

  function Load_Response
    (XML : in String) return Message.Response.Object'Class;
  -- Build a Response object (either a standard response or an error
  -- response) by parsing the XML response string.

  function Load_Response
    (XML : in Unbounded_String) return Message.Response.Object'Class;
  -- As above but using an Unbounded_String

  function Image (O : in Object'Class) return String;
  -- Returns XML representation of object O

  function Image (O : in Object'Class) return Unbounded_String;
  -- Idem as above but returns an Unbounded_String instead of a String

end SOAP.Message.XML;

```

B.74 SOAP.Parameters

```

-----
--                                     Ada Web Server                                     --
--                                                                                       --
--                                     Copyright (C) 2000-2005                             --
--                                     AdaCore                                             --
--                                                                                       --
--  This library is free software; you can redistribute it and/or modify                --
--  it under the terms of the GNU General Public License as published by                --
--  the Free Software Foundation; either version 2 of the License, or (at              --
--  your option) any later version.                                                       --
--                                                                                       --
--  This library is distributed in the hope that it will be useful, but                 --
--  WITHOUT ANY WARRANTY; without even the implied warranty of                         --
--  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU                   --
--  General Public License for more details.                                              --
--                                                                                       --
--  You should have received a copy of the GNU General Public License                  --
--  along with this library; if not, write to the Free Software Foundation,            --
--  Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.                      --
--                                                                                       --
--  As a special exception, if other files instantiate generics from this               --
--  unit, or you link this unit with other files to produce an executable,             --
--  this unit does not by itself cause the resulting executable to be                  --
--  covered by the GNU General Public License. This exception does not                 --
--  however invalidate any other reasons why the executable file might be              --
--  covered by the GNU Public License.                                                    --
-----

with Ada.Calendar;
with Ada.Strings.Unbounded;

with SOAP.Types;

package SOAP.Parameters is

  use Ada.Strings.Unbounded;

  Max_Parameters : constant := 50;
  -- This is the maximum number of parameters supported by this
  -- implementation.

  type List is private;

  function Argument_Count (P : in List) return Natural;
  -- Returns the number of parameters in P

  function Argument (P : in List; Name : in String) return Types.Object'Class;
  -- Returns parameters named Name in P. Raises Types.Data_Error if not
  -- found.

  function Argument (P : in List; N : in Positive) return Types.Object'Class;
  -- Returns Nth parameters in P. Raises Types.Data_Error if not found

  function Exist (P : in List; Name : in String) return Boolean;
  -- Returns True if parameter named Name exist in P and False otherwise

  function Get (P : in List; Name : in String) return Types.Long;
  -- Returns parameter named Name in P as a Long value. Raises
  -- Types.Data_Error if this parameter does not exist or is not a Long.

  function Get (P : in List; Name : in String) return Integer;
  -- Returns parameter named Name in P as an Integer value. Raises

```

```

-- Types.Data_Error if this parameter does not exist or is not an Integer.

function Get (P : in List; Name : in String) return Types.Short;
-- Returns parameter named Name in P as a Short value. Raises
-- Types.Data_Error if this parameter does not exist or is not an Short.

function Get (P : in List; Name : in String) return Types.Byte;
-- Returns parameter named Name in P as a Byte value. Raises
-- Types.Data_Error if this parameter does not exist or is not a Byte.

function Get (P : in List; Name : in String) return Long_Float;
-- Returns parameter named Name in P as a Float value. Raises
-- Types.Data_Error if this parameter does not exist or is not a Float.

function Get (P : in List; Name : in String) return Long_Long_Float;
-- Returns parameter named Name in P as a Float value. Raises
-- Types.Data_Error if this parameter does not exist or is not a Double.

function Get (P : in List; Name : in String) return String;
-- Returns parameter named Name in P as a String value. Raises
-- Types.Data_Error if this parameter does not exist or is not a String.

function Get (P : in List; Name : in String) return Unbounded_String;
-- Idem as above, but return an Unbounded_String

function Get (P : in List; Name : in String) return Boolean;
-- Returns parameter named Name in P as a Boolean value. Raises
-- Types.Data_Error if this parameter does not exist or is not a Boolean.

function Get (P : in List; Name : in String) return Ada.Calendar.Time;
-- Returns parameter named Name in P as a Time value. Raises
-- Types.Data_Error if this parameter does not exist or is not a time.

function Get (P : in List; Name : in String) return Types.Unsigned_Long;
-- Returns parameter named Name in P as a Unsigned_Long value. Raises
-- Types.Data_Error if this parameter does not exist or is not an
-- Unsigned_Long.

function Get (P : in List; Name : in String) return Types.Unsigned_Int;
-- Returns parameter named Name in P as a Unsigned_Int value. Raises
-- Types.Data_Error if this parameter does not exist or is not an
-- Unsigned_Int.

function Get (P : in List; Name : in String) return Types.Unsigned_Short;
-- Returns parameter named Name in P as a Unsigned_Short value. Raises
-- Types.Data_Error if this parameter does not exist or is not an
-- Unsigned_Short.

function Get (P : in List; Name : in String) return Types.Unsigned_Byte;
-- Returns parameter named Name in P as a Unsigned_Byte value. Raises
-- Types.Data_Error if this parameter does not exist or is not an
-- Unsigned_Byte.

function Get (P : in List; Name : in String) return Types.SOAP_Base64;
-- Returns parameter named Name in P as a SOAP Base64 value. Raises
-- Types.Data_Error if this parameter does not exist or is not a SOAP
-- Base64.

function Get (P : in List; Name : in String) return Types.SOAP_Record;
-- Returns parameter named Name in P as a SOAP Struct value. Raises
-- Types.Data_Error if this parameter does not exist or is not a SOAP
-- Struct.

function Get (P : in List; Name : in String) return Types.SOAP_Array;
-- Returns parameter named Name in P as a SOAP Array value. Raises

```

```

-- Types.Data_Error if this parameter does not exist or is not a SOAP
-- Array.

-----
-- Constructors --
-----

function "&" (P : in List; O : in Types.Object'Class) return List;
function "+" (O : in Types.Object'Class) return List;

-----
-- Validation --
-----

procedure Check (P : in List; N : in Natural);
-- Checks that there is exactly N parameters or raise Types.Data_Error

procedure Check_Integer (P : in List; Name : in String);
-- Checks that parameter named Name exist and is an Integer value

procedure Check_Float (P : in List; Name : in String);
-- Checks that parameter named Name exist and is a Float value

procedure Check_Boolean (P : in List; Name : in String);
-- Checks that parameter named Name exist and is a Boolean value

procedure Check_Time_Instant (P : in List; Name : in String);
-- Checks that parameter named Name exist and is a Time_Instant value

procedure Check_Base64 (P : in List; Name : in String);
-- Checks that parameter named Name exist and is a Base64 value

procedure Check_Null (P : in List; Name : in String);
-- Checks that parameter named Name exist and is a Null value

procedure Check_Record (P : in List; Name : in String);
-- Checks that parameter named Name exist and is a Record value

procedure Check_Array (P : in List; Name : in String);
-- Checks that parameter named Name exist and is an Array value

private
-- implementation removed
end SOAP.Parameters;

```

B.75 SOAP.Types

```

-----
--                               Ada Web Server                               --
--                               Copyright (C) 2001-2006                       --
--                               AdaCore                                         --
--                               -----                                         --
--   This library is free software; you can redistribute it and/or modify      --
--   it under the terms of the GNU General Public License as published by      --
--   the Free Software Foundation; either version 2 of the License, or (at    --
--   your option) any later version.                                           --
--                                                                              --
--   This library is distributed in the hope that it will be useful, but       --
--   WITHOUT ANY WARRANTY; without even the implied warranty of               --
--   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU        --
--   General Public License for more details.                                  --
--                                                                              --
--   You should have received a copy of the GNU General Public License        --
--   along with this library; if not, write to the Free Software Foundation,  --
--   Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.          --
--                                                                              --
--   As a special exception, if other files instantiate generics from this     --
--   unit, or you link this unit with other files to produce an executable,   --
--   this unit does not by itself cause the resulting executable to be        --
--   covered by the GNU General Public License. This exception does not       --
--   however invalidate any other reasons why the executable file might be    --
--   covered by the GNU Public License.                                        --
-----

--   This package contains all SOAP types supported by this implementation.
--   Here are some notes about adding support for a new SOAP type (not a
--   container) and the corresponding WSDL support:
--
--   1. Add new type derived from scalar in this package. Implements all
--      inherited routines (Image, XML_Image and XML_Type). Implements also
--      a constructor for this new type and a routine named V to get the
--      value as an Ada type.
--
--   2. In SOAP.Parameters add corresponding Get routine.
--
--   3. In SOAP.WSDL, add the new type name in Parameter_Type.
--
--   4. Add support for this new type in all SOAP.WSDL routines. All routines
--      are using a case statement to be sure that it won't compile without
--      fixing it first. For obvious reasons, only SOAP.WSDL.To_Type and
--      SOAP.WSDL.From_Ada are not using a case statement, be sure to do the
--      right Change There.
--
--   5. Finally add support for this type in SOAP.Message.XML. Add this type
--      into Type_State, write the corresponding parse procedure and fill entry
--      into Handlers. Again after adding the proper type into Type_State the
--      compiler will issue errors where changes are needed.

with Ada.Calendar;
with Ada.Finalization;
with Ada.Strings.Unbounded;

with SOAP.Name_Space;

package SOAP.Types is

  use Ada.Strings.Unbounded;

```

```

Data_Error : exception;
-- Raised when a variable has not the expected type

type Object is abstract tagged private;
-- Root type for all SOAP types defined in this package

type Object_Access is access all Object'Class;

type Object_Safe_Pointer is tagged private;
-- A safe pointer to a SOAP object, such objects are controlled so the
-- memory is freed automatically.

type Object_Set is array (Positive range <>) of Object_Safe_Pointer;
-- A set of SOAP types. This is used to build arrays or records. We use
-- Positive for the index to have the item index map the SOAP array
-- element order.

function Image (O : in Object) return String;
-- Returns O value image.

function XML_Image (O : in Object) return String;
-- Returns O value encoded for use by the Payload object or Response
-- object.

function XML_Type (O : in Object) return String;
-- Returns the XML type for the object.

function Name (O : in Object'Class) return String;
-- Returns name for object O.

function "+" (O : in Object'Class) return Object_Safe_Pointer;
-- Allocate an object into the heap and return a safe pointer to it.

function "-" (O : in Object_Safe_Pointer) return Object'Class;
-- Returns the object associated with the safe pointer.

type Scalar is abstract new Object with private;
-- Scalar types are using a by-copy semantic.

type Composite is abstract new Object with private;
-- Composite types are using a by-reference semantic for efficiency
-- reason. Not that these types are not thread safe.

function V (C : in Composite) return Object_Set is abstract;

-----
-- Any Type --
-----

XML_Any_Type : aliased constant String := "xsd:anyType";

type XSD_Any_Type is new Object with private;

function Image      (O : in XSD_Any_Type) return String;
function XML_Image  (O : in XSD_Any_Type) return String;
function XML_Type   (O : in XSD_Any_Type) return String;

function Any
  (V      : in Object'Class;
   Name : in String := "item") return XSD_Any_Type;

function V (O : in XSD_Any_Type) return Object_Access;

-----
-- Array --

```



```

-----

XML_Array      : constant String := "soapenc:Array";
XML_Undefined  : aliased constant String := "xsd:ur-type";

type SOAP_Array is new Composite with private;

function Image      (O : in SOAP_Array) return String;
function XML_Image  (O : in SOAP_Array) return String;
function XML_Type   (O : in SOAP_Array) return String;

function A
  (V      : in Object_Set;
   Name    : in String;
   Type_Name : in String := "")
  return SOAP_Array;
-- Type_Name of the array's elements, if not specified it will be computed
-- based on element's name.

function Size (O : in SOAP_Array) return Natural;
-- Returns the number of item into the array

function V (O : in SOAP_Array; N : in Positive) return Object'Class;
-- Returns SOAP_Array item at position N

function V (O : in SOAP_Array) return Object_Set;

-----
-- Base64 --
-----

XML_Base64      : aliased constant String := "soapenc:base64";
XML_Base64_Binary : constant String := "xsd:base64Binary";

type SOAP_Base64 is new Scalar with private;

function Image      (O : in SOAP_Base64) return String;
function XML_Image  (O : in SOAP_Base64) return String;
function XML_Type   (O : in SOAP_Base64) return String;

function B64
  (V      : in String;
   Name : in String := "item")
  return SOAP_Base64;

function V (O : in SOAP_Base64) return String;

-----
-- Boolean --
-----

XML_Boolean : aliased constant String := "xsd:boolean";

type XSD_Boolean is new Scalar with private;

function Image      (O : in XSD_Boolean) return String;
function XML_Image  (O : in XSD_Boolean) return String;
function XML_Type   (O : in XSD_Boolean) return String;

function B (V : in Boolean; Name : in String := "item") return XSD_Boolean;
function V (O : in XSD_Boolean) return Boolean;

-----
-- Byte --
-----

```

```

type Byte is range -2**7 .. 2**7 - 1;

XML_Byte : aliased constant String := "xsd:byte";

type XSD_Byte is new Scalar with private;

function Image      (O : in XSD_Byte) return String;
function XML_Image  (O : in XSD_Byte) return String;
function XML_Type   (O : in XSD_Byte) return String;

function B (V : in Byte; Name : in String := "item") return XSD_Byte;
function V (O : in XSD_Byte) return Byte;

-----
-- Double --
-----

XML_Double : aliased constant String := "xsd:double";

type XSD_Double is new Scalar with private;

function Image      (O : in XSD_Double) return String;
function XML_Image  (O : in XSD_Double) return String;
function XML_Type   (O : in XSD_Double) return String;

function D
  (V      : in Long_Long_Float;
   Name : in String      := "item")
  return XSD_Double;

function V (O : in XSD_Double) return Long_Long_Float;

-----
-- Float --
-----

XML_Float : aliased constant String := "xsd:float";

type XSD_Float is new Scalar with private;

function Image      (O : in XSD_Float) return String;
function XML_Image  (O : in XSD_Float) return String;
function XML_Type   (O : in XSD_Float) return String;

function F (V : in Long_Float; Name : in String := "item") return XSD_Float;
function V (O : in XSD_Float) return Long_Float;

-----
-- Integer --
-----

XML_Int : aliased constant String := "xsd:int";

type XSD_Integer is new Scalar with private;

function Image      (O : in XSD_Integer) return String;
function XML_Image  (O : in XSD_Integer) return String;
function XML_Type   (O : in XSD_Integer) return String;

function I (V : in Integer; Name : in String := "item") return XSD_Integer;
function V (O : in XSD_Integer) return Integer;

-----
-- Long --

```

```

-----

type Long is range -2**63 .. 2**63 - 1;

XML_Long : aliased constant String := "xsd:long";

type XSD_Long is new Scalar with private;

function Image      (O : in XSD_Long) return String;
function XML_Image  (O : in XSD_Long) return String;
function XML_Type   (O : in XSD_Long) return String;

function L (V : in Long; Name : in String := "item") return XSD_Long;
function V (O : in XSD_Long) return Long;

-----
-- Null --
-----

XML_Null : constant String := "1";

type XSD_Null is new Scalar with private;

function XML_Image (O : in XSD_Null) return String;
function XML_Type  (O : in XSD_Null) return String;

function N (Name : in String := "item") return XSD_Null;

-----
-- Record --
-----

type SOAP_Record is new Composite with private;

function Image      (O : in SOAP_Record) return String;
function XML_Image  (O : in SOAP_Record) return String;
function XML_Type   (O : in SOAP_Record) return String;

function R
  (V      : in Object_Set;
   Name   : in String;
   Type_Name : in String := "")
  return SOAP_Record;
-- If Type_Name is omitted then the type name is the name of the record.
-- Type_Name must be specified for item into an array for example.

function V (O : in SOAP_Record; Name : in String) return Object'Class;
-- Returns SOAP_Record field named Name

function V (O : in SOAP_Record) return Object_Set;

-----
-- Short --
-----

type Short is range -2**15 .. 2**15 - 1;

XML_Short : aliased constant String := "xsd:short";

type XSD_Short is new Scalar with private;

function Image      (O : in XSD_Short) return String;
function XML_Image  (O : in XSD_Short) return String;
function XML_Type   (O : in XSD_Short) return String;

```

```

function S (V : in Short; Name : in String := "item") return XSD_Short;
function V (O : in XSD_Short) return Short;

-----
-- String --
-----

XML_String : aliased constant String := "xsd:string";

type XSD_String is new Scalar with private;

function Image      (O : in XSD_String) return String;
function XML_Image  (O : in XSD_String) return String;
function XML_Type   (O : in XSD_String) return String;

function S
  (V      : in String;
   Name   : in String := "item")
  return XSD_String;

function S
  (V      : in Unbounded_String;
   Name   : in String  := "item")
  return XSD_String;

function V (O : in XSD_String) return String;

function V (O : in XSD_String) return Unbounded_String;

-----
-- TimeInstant --
-----

XML_Time_Instant : aliased constant String := "xsd:timeInstant";
XML_Date_Time    : constant String := "xsd:dateTime";

type XSD_Time_Instant is new Scalar with private;

function Image      (O : in XSD_Time_Instant) return String;
function XML_Image  (O : in XSD_Time_Instant) return String;
function XML_Type   (O : in XSD_Time_Instant) return String;

subtype TZ is Integer range -11 .. +11;
GMT : constant TZ := 0;

function T
  (V      : in Ada.Calendar.Time;
   Name   : in String           := "item";
   Timezone : in TZ             := GMT)
  return XSD_Time_Instant;

function V (O : in XSD_Time_Instant) return Ada.Calendar.Time;
-- Returns a GMT date and time

-----
-- Unsigned_Long --
-----

type Unsigned_Long is mod 2**64;

XML_Unsigned_Long : aliased constant String := "xsd:unsignedLong";

type XSD_Unsigned_Long is new Scalar with private;

function Image      (O : in XSD_Unsigned_Long) return String;

```

```

function XML_Image (O : in XSD_Unsigned_Long) return String;
function XML_Type  (O : in XSD_Unsigned_Long) return String;

function UL
  (V      : in Unsigned_Long;
   Name : in String := "item") return XSD_Unsigned_Long;
function V (O : in XSD_Unsigned_Long) return Unsigned_Long;

-----
-- Unsigned_Int --
-----

type Unsigned_Int is mod 2**32;

XML_Unsigned_Int : aliased constant String := "xsd:unsignedInt";

type XSD_Unsigned_Int is new Scalar with private;

function Image      (O : in XSD_Unsigned_Int) return String;
function XML_Image  (O : in XSD_Unsigned_Int) return String;
function XML_Type   (O : in XSD_Unsigned_Int) return String;

function UI
  (V      : in Unsigned_Int;
   Name : in String := "item") return XSD_Unsigned_Int;
function V (O : in XSD_Unsigned_Int) return Unsigned_Int;

-----
-- Unsigned_Short --
-----

type Unsigned_Short is mod 2**16;

XML_Unsigned_Short : aliased constant String := "xsd:unsignedShort";

type XSD_Unsigned_Short is new Scalar with private;

function Image      (O : in XSD_Unsigned_Short) return String;
function XML_Image  (O : in XSD_Unsigned_Short) return String;
function XML_Type   (O : in XSD_Unsigned_Short) return String;

function US
  (V      : in Unsigned_Short;
   Name : in String := "item") return XSD_Unsigned_Short;
function V (O : in XSD_Unsigned_Short) return Unsigned_Short;

-----
-- Unsigned_Byte --
-----

type Unsigned_Byte is mod 2**8;

XML_Unsigned_Byte : aliased constant String := "xsd:unsignedByte";

type XSD_Unsigned_Byte is new Scalar with private;

function Image      (O : in XSD_Unsigned_Byte) return String;
function XML_Image  (O : in XSD_Unsigned_Byte) return String;
function XML_Type   (O : in XSD_Unsigned_Byte) return String;

function UB
  (V      : in Unsigned_Byte;
   Name : in String := "item") return XSD_Unsigned_Byte;
function V (O : in XSD_Unsigned_Byte) return Unsigned_Byte;

```

```

-----
-- Enumeration --
-----

type SOAP_Enumeration is new Scalar with private;

function Image      (O : in SOAP_Enumeration) return String;
function XML_Image  (O : in SOAP_Enumeration) return String;
function XML_Type   (O : in SOAP_Enumeration) return String;

function E
  (V          : in String;
   Type_Name  : in String;
   Name       : in String := "item")
  return SOAP_Enumeration;

function V (O : in SOAP_Enumeration) return String;

-----
-- Get --
-----

-- It is possible to pass an XSD_Any_Type to all get routines below. The
-- proper value will be returned if the XSD_Any_Type is actually of this
-- type.

function Get (O : in Object'Class) return XSD_Any_Type;
-- Returns O value as an XSD_Any_Type. Raises Data_Error if O is not a
-- SOAP anyType.

function Get (O : in Object'Class) return Long;
-- Returns O value as a Long. Raises Data_Error if O is not a SOAP
-- Long.

function Get (O : in Object'Class) return Integer;
-- Returns O value as an Integer. Raises Data_Error if O is not a SOAP
-- Integer.

function Get (O : in Object'Class) return Short;
-- Returns O value as a Short. Raises Data_Error if O is not a SOAP
-- Short.

function Get (O : in Object'Class) return Byte;
-- Returns O value as a Byte. Raises Data_Error if O is not a SOAP
-- Byte.

function Get (O : in Object'Class) return Long_Float;
-- Returns O value as a Long_Float. Raises Data_Error if O is not a SOAP
-- Float.

function Get (O : in Object'Class) return Long_Long_Float;
-- Returns O value as a Long_Long_Float. Raises Data_Error if O is not a
-- SOAP Double.

function Get (O : in Object'Class) return String;
-- Returns O value as a String. Raises Data_Error if O is not a SOAP
-- String.

function Get (O : in Object'Class) return Unbounded_String;
-- As above but returns an Unbounded_String

function Get (O : in Object'Class) return Boolean;
-- Returns O value as a Boolean. Raises Data_Error if O is not a SOAP
-- Boolean.

```

```

function Get (O : in Object'Class) return Ada.Calendar.Time;
-- Returns O value as a Time. Raises Data_Error if O is not a SOAP
-- Time.

function Get (O : in Object'Class) return Unsigned_Long;
-- Returns O value as a Unsigned_Long. Raises Data_Error if O is not a SOAP
-- Unsigned_Long.

function Get (O : in Object'Class) return Unsigned_Int;
-- Returns O value as a Unsigned_Byte. Raises Data_Error if O is not a SOAP
-- Unsigned_Int.

function Get (O : in Object'Class) return Unsigned_Short;
-- Returns O value as a Unsigned_Short. Raises Data_Error if O is not a
-- SOAP Unsigned_Short.

function Get (O : in Object'Class) return Unsigned_Byte;
-- Returns O value as a Unsigned_Byte. Raises Data_Error if O is not a SOAP
-- Unsigned_Byte.

function Get (O : in Object'Class) return SOAP_Base64;
-- Returns O value as a SOAP Base64. Raises Data_Error if O is not a SOAP
-- Base64 object.

function Get (O : in Object'Class) return SOAP_Record;
-- Returns O value as a SOAP Struct. Raises Data_Error if O is not a SOAP
-- Struct.

function Get (O : in Object'Class) return SOAP_Array;
-- Returns O value as a SOAP Array. Raises Data_Error if O is not a SOAP
-- Array.

-----
-- Name space --
-----

procedure Set_Name_Space
(O : in out Object'Class;
 NS : in Name_Space.Object);
-- Set the name space for object O

function Name_Space (O : in Object'Class) return Name_Space.Object;
-- Returns name space associated with object O

private
-- implementation removed
end SOAP.Types;

```


Index

A

ABORTABLE_V 69
 Accept_Queue_Size 16
 ACCEPT_QUEUE_SIZE 69
 ACTIVITY_COUNTER_V 69
 ACTIVITY_TIME_STAMP_V 69
 ada2wsdl 47
 ada2wsdl limitations 52
 Admin_URI 9, 16
 Ajax 36
 authentication 20
 AWS 77
 AWS.Attachments 78
 AWS.Client 80
 AWS.Client.Hotplug 86
 AWS.Communication 87
 AWS.Communication.Client 88
 AWS.Communication.Server 89
 AWS.Config 90
 AWS.Config.Ini 96
 AWS.Config.Set 97
 AWS.Containers.Tables 102
 AWS.Default 104
 AWS.Dispatchers 107
 AWS.Dispatchers.Callback 109
 AWS.Exceptions 110
 AWS.Headers 112
 AWS.Headers.Values 114
 aws.ini 16
 AWS.Jabber 116
 AWS.LDAP.Client 118
 AWS.Log 123
 AWS.Messages 126
 AWS.MIME 131
 AWS.Net 134
 AWS.Net.Buffered 139
 AWS.Net.Log 141
 AWS.Net.Log.Callbacks 143
 AWS.Net.SSL 145
 AWS.Net.SSL.Certificate 148
 AWS.Net.Std 3
 AWS.Parameters 149
 AWS.POP 150
 AWS.Resources 154
 AWS.Resources.Embedded 157
 AWS.Resources.Files 159
 AWS.Resources.Streams 160
 AWS.Resources.Streams.Disk 162
 AWS.Resources.Streams.Disk.Once 164
 AWS.Resources.Streams.Memory 165
 AWS.Resources.Streams.Memory.ZLib 167
 AWS.Response 169
 AWS.Server 175
 AWS.Server.Hotplug 179
 AWS.Server.Log 181
 AWS.Server.Push 183
 AWS.Server.Status 187
 AWS.Services.Callbacks 189
 AWS.Services.Directory 190
 AWS.Services.Dispatchers 193
 AWS.Services.Dispatchers.Linker 195

AWS.Services.Dispatchers.Method 196
 AWS.Services.Dispatchers.URI 198
 AWS.Services.Dispatchers.Virtual.Host 200
 AWS.Services.Download 202
 AWS.Services.Page_Server 204
 AWS.Services.Split_Pages 205
 AWS.Services.Split_Pages.Alpha 207
 AWS.Services.Split_Pages.Alpha.Bounded 209
 AWS.Services.Split_Pages.Alpha.Uniform 211
 AWS.Services.Split_Pages.Alpha.Uniform.Alpha 213
 AWS.Services.Split_Pages.Alpha.Uniform.Overlapping 215
 AWS.Services.Transient_Pages 216
 AWS.Session 217
 AWS.SMTP 221
 AWS.SMTP.Client 224
 AWS.Status 227
 AWS.Templates 232
 AWS.Translator 233
 AWS.URL 235
 aws_action_clear.tjs 36
 aws_action_replace.tjs 36
 awsres 67

B

basic 20
 Building 4
 Building resources 67

C

Callback 9, 10
 Callback procedure 10
 Callback, dispatcher 31
 Callback, dispatcher API 109
 Case_Sensitive_Parameters 9, 16
 certificate 27
 Certificate (string) 16
 Check_URL_Validity 16
 Cleaner_Client_Data_Timeout 16
 CLEANER_CLIENT_DATA_TIMEOUT 69
 Cleaner_Client_Header_Timeout 16
 CLEANER_CLIENT_HEADER_TIMEOUT 69
 Cleaner_Server_Response_Timeout 17
 CLEANER_SERVER_RESPONSE_TIMEOUT 69
 Cleaner_Wait_For_Client_Timeout 16
 CLEANER_WAIT_FOR_CLIENT_TIMEOUT 69
 client HTTP 29
 Client protocol 29
 Communication 21
 Communication, Client 22
 Communication, Server 22
 Configuration options 16

D

digest	20
Directory browser	31
Directory_Browser_Page	17
Dispatchers	31
Dispatchers callback	31
Dispatchers Linker	32
Dispatchers method	31
Dispatchers SOAP	32
Dispatchers Timer	32
Dispatchers Transient pages	32
Dispatchers URI	31
Dispatchers virtual host	31
Distributing	12
Down_Image	17
Download Manager	34
draft 302	74

E

exception handler	28
Exceptions, Unexpected exceptions, Exceptions handler	110
Exchange_Certificate	17

F

File upload	21
Force_Client_Data_Timeout	17
FORCE_CLIENT_DATA_TIMEOUT	69
Force_Client_Header_Timeout	17
FORCE_CLIENT_HEADER_TIMEOUT	69
Force_Server_Response_Timeout	17
FORCE_SERVER_RESPONSE_TIMEOUT	69
Force_Wait_For_Client_Timeout	17
FORCE_WAIT_FOR_CLIENT_TIMEOUT	69
Form parameters	11
Free_Slots_Keep_Alive_Limit	17

G

GNAT	3
GNU/Ada	3

H

Hello world	10
hotplug	22
Hotplug.Port	17
HTML File Upload	73
HTTP Authentication	74
HTTP declaration	8
HTTP/1.0	73
HTTP/1.1	74
HTTPS	27

I

IMAP/POP	74
ini file	16
Installing	6

J

Jabber	65
Jabber Binding	116
Jabber message	65
Jabber presence	65

K

Key	17
KEYS_M	69

L

LDAP	63
LDAP Binding	118
LDAP Directory	63
libcrypto.a	4
libssl.a	4
Lightweight Directory Access Protocol	63
Line_Stack_Size	17
linker, dispatcher	32
LOG	70
Log.Flush	25
Log.Start	25
Log.Start_Error	25
Log.Stop	25
Log.Stop_Error	25
Log_Extended_Fields	18
LOG_FILE	70
Log_File_Directory	18
Log_Filename_Prefix	17, 18
LOG_MODE	70
Log_Split_Mode	17, 18
Logo	18
LOGO	69
logs	25

M

Max_Connection	9, 18
MAX_CONNECTION	70
method, dispatcher	31
MIME	73

O

OPENED_V	70
OpenLDAP	3
OpenSSL	3

P

Page server	11, 33
pages, split	34
pages, transient	33
Parameters	11
Parameters Get	12
Parameters Get_Name	12
Payload	44
PEERNAME_V	70
POP	59, 73
Port	9
Post Office Protocol	59
program_name.ini	16

Push 24

Q

QUIT_V 70

R

Receive_Timeout 18
 RECEIVE_TIMEOUT 70
 References 73
 resources 10
 Resources 67
 Retrieving e-mail 59
 RFC 0821 73
 RFC 1867 73
 RFC 1939 73
 RFC 1945 73
 RFC 2049 73
 RFC 2109 73
 RFC 2195 74
 RFC 2554 74
 RFC 2616 74
 RFC 2617 74

S

Secure server 27
 Security 9
 SECURITY 70
 Security_Mode 18
 Self dependant 67
 Send 25
 Send_Timeout 18
 SEND_TIMEOUT 70
 Send_To 25
 Sending e-mail 59
 Sending message 21
 Server Push 24
 Server_Host 18
 Server_Name 18
 SERVER_NAME 70
 Server_Port 18
 SERVER_PORT 70
 SERVER SOCK 70
 Session 9, 19
 SESSION_CLEANUP_INTERVAL 70
 Session_Cleanup_Interval (duration) 19
 SESSION_LIFETIME 70
 Session_Lifetime (duration) 19
 SESSIONS_TERMINATE_V 70
 SESSIONS_TS_V 70
 SESSIONS_V 70
 Simple Mail Transfer Protocol 59
 Simple Object Access Protocol 43
 Simple Page server 33
 Simple server 11
 SLOT_ACTIVITY_COUNTER_V 70
 SMTP 59, 73
 SMTP Authentication 74
 SOAP 43
 SOAP (API) 238
 SOAP 1.1 75
 SOAP Client 43

SOAP Dispatcher 46
 SOAP Server 44
 SOAP, dispatcher 32
 SOAP.Client 239
 SOAP.Dispatchers 240
 SOAP.Dispatchers.Callback 46, 242
 SOAP.Message 243
 SOAP.Message.XML 245
 SOAP.Parameters 246
 SOAP.Types 249
 SOAPAction 44
 SOCK_V 71
 socket binding 3
 Socket log 29
 split pages 34
 SSL 27, 74
 starting server 8
 Static Page server 33
 Status 69
 Status_Page 19
 STATUS_PAGE 71
 Stream resources 67

T

timer, dispatcher 32
 TLS 27
 transient pages 33
 transient pages, dispatcher 32
 Transient_Cleanup_Interval 19
 Transient_Lifetime 19

U

Up_Image 19
 upload, client 30
 upload, server 21
 Upload_Directory 19
 UPLOAD_DIRECTORY 71
 URI, dispatcher 31
 Using resources 67
 Utils.SOAP_Wrapper 45

V

VALUES_M 71
 VERSION 71
 virtual host, dispatcher 31

W

we_icons 36
 we_js 36
 Web Elements 36
 Web Service Definition Language 43, 47
 Web_Server 9
 Working with Server sockets 25
 WSDL 43, 47
 WSDL, Client 52
 WSDL, Server 53
 wsdl2aws 47, 54
 wsdl2aws limitations 56
 WWW_Root 19

