AWS Documentation

Release 25.1.0

AdaCore

Nov 12, 2026

CONTENTS

1 Introduction **1**
- 1.1 HTTP/2 . . . 2

2 Building AWS **3**
- 2.1 Requirements . . . 3
- 2.2 AWS.Net.Std . . . 3
- 2.3 Building . . . 4
- 2.4 Building on cross-platforms . . . 4
- 2.5 Demos . . . 5
- 2.6 Installing . . . 6

3 Using AWS **9**
- 3.1 Setting up environment . . . 9
  - 3.1.1 Using environment variables . . . 9
  - 3.1.2 Using GNAT Project Files . . . 9
- 3.2 Basic notions . . . 10
  - 3.2.1 Building an AWS server . . . 10
  - 3.2.2 Callback procedure . . . 12
  - 3.2.3 Form parameters . . . 13
  - 3.2.4 Distribution of an AWS server . . . 14
- 3.3 Building answers . . . 15
  - 3.3.1 Redirection . . . 15
  - 3.3.2 New location for a page . . . 15
  - 3.3.3 Authentication required . . . 15
  - 3.3.4 Sending back an error message . . . 15
  - 3.3.5 Response from a string . . . 15
  - 3.3.6 Response from a Stream_Element_Array . . . 15
  - 3.3.7 Response from a file . . . 15
  - 3.3.8 Response from a stream . . . 16
  - 3.3.9 Response from a on-disk stream . . . 16
  - 3.3.10 Response from a on-disk once stream . . . 17
  - 3.3.11 Response from a memory stream . . . 17
  - 3.3.12 Response from a compressed memory stream . . . 17
  - 3.3.13 Split page . . . 17
  - 3.3.14 Response a from pipe stream . . . 18
- 3.4 Configuration options . . . 18
- 3.5 Session handling . . . 24
- 3.6 HTTP state management . . . 25
- 3.7 Authentication . . . 26
- 3.8 File upload . . . 27

3.9 Communication . . . 27
3.9.1 Communication - client side . . . 27
3.9.2 Communication - server side . . . 28
3.10 Hotplug module . . . 28
3.10.1 Hotplug module - server activation . . . 28
3.10.2 Hotplug module - creation . . . 29
3.11 Server Push . . . 30
3.12 Working with Server sockets . . . 30
3.13 Server Log . . . 31
3.14 Secure server . . . 32
3.14.1 Initialization . . . 33
3.14.2 Verify callback . . . 33
3.14.3 Self-signed certificate . . . 34
3.14.4 Using a Certificate Authority . . . 34
3.14.5 Security level . . . 36
3.14.6 Protocol . . . 37
3.15 Unexpected exception handler . . . 37
3.16 Socket log . . . 38
3.17 Client side . . . 38
3.17.1 Client . . . 38
3.17.2 Secure Client . . . 39

4 High level services **41**
4.1 Directory browser . . . 41
4.2 Dispatchers . . . 41
4.2.1 Callback dispatcher . . . 41
4.2.2 Method dispatcher . . . 41
4.2.3 URI dispatcher . . . 42
4.2.4 Virtual host dispatcher . . . 42
4.2.5 Transient pages dispatcher . . . 42
4.2.6 Timer dispatcher . . . 42
4.2.7 Linker dispatcher . . . 42
4.2.8 SOAP dispatcher . . . 43
4.3 Static Page server . . . 43
4.4 Transient Pages . . . 43
4.5 Split pages . . . 44
4.6 Download Manager . . . 44
4.7 Web Elements . . . 46
4.7.1 Installation . . . 46
4.7.2 Ajax . . . 47
4.8 Web Blocks . . . 53
4.8.1 Web Block example . . . 53
4.8.2 Web Block and Ajax . . . 56
4.8.3 Web Block and templates2ada . . . 57
4.9 Web Cross-References . . . 60
4.10 WebSockets . . . 60
4.10.1 Introduction to WebSockets . . . 60
4.10.2 WebSockets on the client (javascript) . . . 61
4.10.3 WebSockets on the client (Ada) . . . 61
4.10.4 WebSockets on the server . . . 62

5 Using SOAP **65**
5.1 SOAP Client . . . 65
5.2 SOAP Server . . . 66

5.2.1 Step by step instructions . . . 66
5.2.2 SOAP helpers . . . 67

6 Using WSDL **69**
6.1 Creating WSDL documents . . . 69
6.1.1 Using ada2wsdl . . . 69
6.1.2 Ada mapping to WSDL . . . 71
6.1.3 ada2wsdl . . . 77
6.1.4 ada2wsdl limitations . . . 78
6.2 Working with WSDL documents . . . 78
6.2.1 Client side (stub) . . . 78
6.2.2 Server side (skeleton) . . . 79
6.2.3 wsdl2aws . . . 80
6.2.4 wsdl2aws code generator . . . 82
6.2.5 wsdl2aws limitations . . . 84
6.2.6 awsascb . . . 84
6.3 Using ada2wsdl and wsdl2aws together . . . 85

7 Working with mails **87**
7.1 Sending e-mail . . . 87
7.2 Retrieving e-mail . . . 88

8 LDAP **91**

9 Jabber **93**
9.1 Jabber presence . . . 93
9.2 Jabber message . . . 93

10 Resources **95**
10.1 Building resources . . . 95
10.2 Using resources . . . 95
10.3 Stream resources . . . 95
10.4 awsres tool . . . 96

11 Status page **97**

12 References **101**

13 AWS API Reference **105**
13.1 AWS . . . 105
13.2 AWS.Attachments . . . 107
13.3 AWS.Client . . . 112
13.4 AWS.Client.Hotplug . . . 124
13.5 AWS.Communication . . . 126
13.6 AWS.Communication.Client . . . 128
13.7 AWS.Communication.Server . . . 129
13.8 AWS.Config . . . 131
13.9 AWS.Config.Ini . . . 141
13.10 AWS.Config.Set . . . 142
13.11 AWS.Containers.Tables . . . 151
13.12 AWS.Cookie . . . 155
13.13 AWS.Default . . . 159
13.14 AWS.Dispatchers . . . 163
13.15 AWS.Dispatchers.Callback . . . 165
13.16 AWS.Exceptions . . . 166

13.17 AWS.Headers . . . 168
13.18 AWS.Headers.Values . . . 171
13.19 AWS.Jabber . . . 174
13.20 AWS.LDAP.Client . . . 175
13.21 AWS.Log . . . 182
13.22 AWS.Messages . . . 186
13.23 AWS.MIME . . . 194
13.24 AWS.Net . . . 198
13.25 AWS.Net.Buffered . . . 207
13.26 AWS.Net.Log . . . 210
13.27 AWS.Net.Log.Callbacks . . . 212
13.28 AWS.Net.SSL . . . 214
13.29 AWS.Net.SSL.Certificate . . . 222
13.30 AWS.Net.WebSocket . . . 225
13.31 AWS.Net.WebSocket.Registry . . . 231
13.32 AWS.Net.WebSocket.Registry.Control . . . 235
13.33 AWS.Parameters . . . 236
13.34 AWS.POP . . . 238
13.35 AWS.Resources . . . 243
13.36 AWS.Resources.Embedded . . . 246
13.37 AWS.Resources.Files . . . 248
13.38 AWS.Resources.Streams . . . 250
13.39 AWS.Resources.Streams.Disk . . . 252
13.40 AWS.Resources.Streams.Disk.Once . . . 254
13.41 AWS.Resources.Streams.Memory . . . 255
13.42 AWS.Resources.Streams.Memory.ZLib . . . 257
13.43 AWS.Resources.Streams.Pipe . . . 259
13.44 AWS.Response . . . 261
13.45 AWS.Response.Set . . . 270
13.46 AWS.Server . . . 275
13.47 AWS.Server.Hotplug . . . 281
13.48 AWS.Server.Log . . . 283
13.49 AWS.Server.Push . . . 286
13.50 AWS.Server.Status . . . 292
13.51 AWS.Services.Callbacks . . . 294
13.52 AWS.Services.Directory . . . 295
13.53 AWS.Services.Dispatchers . . . 298
13.54 AWS.Services.Dispatchers.Linker . . . 300
13.55 AWS.Services.Dispatchers.Method . . . 301
13.56 AWS.Services.Dispatchers.URI . . . 303
13.57 AWS.Services.Dispatchers.Virtual_Host . . . 305
13.58 AWS.Services.Download . . . 307
13.59 AWS.Services.Page_Server . . . 309
13.60 AWS.Services.Split_Pages . . . 311
13.61 AWS.Services.Split_Pages.Alpha . . . 314
13.62 AWS.Services.Split_Pages.Alpha.Bounded . . . 316
13.63 AWS.Services.Split_Pages.Uniform . . . 318
13.64 AWS.Services.Split_Pages.Uniform.Alpha . . . 320
13.65 AWS.Services.Split_Pages.Uniform.Overlapping . . . 322
13.66 AWS.Services.Transient_Pages . . . 323
13.67 AWS.Services.Web_Block . . . 325
13.68 AWS.Services.Web_Block.Context . . . 326
13.69 AWS.Services.Web_Block.Registry . . . 328
13.70 AWS.Session . . . 332

13.71 AWS.SMTP . . . 337
13.72 AWS.SMTP.Client . . . 340
13.73 AWS.Status . . . 345
13.74 AWS.Templates . . . 353
13.75 AWS.Translator . . . 354
13.76 AWS.URL . . . 358
13.77 SOAP . . . 362
13.78 SOAP.Client . . . 363
13.79 SOAP.Dispatchers . . . 365
13.80 SOAP.Dispatchers.Callback . . . 367
13.81 SOAP.Message . . . 369
13.82 SOAP.Message.XML . . . 371
13.83 SOAP.Parameters . . . 373
13.84 SOAP.Types . . . 377

Index **395**

CHAPTER

ONE

INTRODUCTION

AWS stands for *Ada Web Server*. It is an Ada implementation of the *HTTP/1.1* and *HTTP/2* protocols as defined in the RFC-2616 from June 1999 and RFC-7640 from May 2015 respectivelly.

The goal is not to build a full Web server but more to make it possible to use a Web browser (like Firefox or Chrome) 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 implements 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 either *OpenSSL*, *LibreSSL* or *GNUTLS* two Open Source SSL implementations.

Major supported features are:

- HTTP/1.1 and HTTP/2 (aka h2c) implementation
- HTTPS/1.1 and HTTPS/2 (aka h2) (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 (server side)
- HTTP state management (client side cookies)
- 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

1.1 HTTP/2

The *HTTP/2* protocol has been designed with speed and security in mind. It is a binary protocol making the exchanged frames less verbose and has support for header compression to even more reduces the payload size. The header compression format is called HPACK (described in RFC-7541 from May 2015) and permits to efficiently represent HTTP header fields by using an Huffman encoding specifically designed for HTTP header's information.

The *HTTP/2* protocol has some specific configuration options. See *Configuration options*. All of them are starting with the *HTTP2* prefix.

Finally the *HTTP/2* protocol is enabled by default and can be disabled by setting the option *HTTP2_Activated* to *false*.

Note that disabling the *HTTP/2* protocol will not make the server unusable, just that during the handshake with the client it won't be selected as not advertised as supported. In this case, AWS server will continue working using the *HTTP/1.1* protocol.

AWS also provides client side supports for the *HTTP/2* protocol. A parameter in the client API can be used to request the *HTTP/1.1* or the *HTTP/2* protocol to be used or to let the client and server decide about the protocol to be used during the handshake.

CHAPTER

TWO

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) ;

  Obviously an Ada compiler is mandatory. Only GNAT is tested, the code should be fairly portable but has never been tested on another compiler. See INSTALL file distributed with AWS for specific versions supported.

- OpenSSL (**optional**) ;

  OpenSSL is an Open Source toolkit implementing the *Secure Sockets Layer* (SSL v2 and v3 and TLS 1.1, 1.2) and much more. It is possible to download the OpenSSL source distribution from *http://www.openssl.org <http://www.openssl.org>* and build it. A Windows binary distribution may also be downloaded there.

- LibreSSL (**optional**) ;

  LibreSSL is an Open Source toolkit implementing the *Secure Sockets Layer* which is fully compatible with OpenSSL. It is possible to download the LibreSSL source distribution from *https://www.libressl.org/>* and build it.

- GNUTLS (**optional**) ;

  GNUTLS is an Open Source toolkit implementing the *Secure Sockets Layer* (SSL v3 and TLS 1.1, 1.2) and much more. It is necessary to install the developers libraries to use it in AWS.

- 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 <http://www.openldap.org>*.

2.2 AWS.Net.Std

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

GNAT
: Version based on *GNAT.Sockets* from GNAT version 20 and later with IPv6 support. This is the **default implementation** used.

IPv6
: Compartible with GNAT before version 20 socket implementation with IPv6 support:

```
$ make setup NETLIB=ipv6
```

IPv4

Compartible with GNAT before version 20 socket implementation based on GNAT.Sockets package without IPv6 support:

```
$ make setup NETLIB=ipv4
```

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 some demos require that *AWS* be built with *SSL* support. If you want to activate *SSL* you must have installed the necessary developers libraries. It is possible to specify the *SSL* implementation to use with the *SOCKET* variable.

To build with *GNUTLS*:

```
$ make SOCKET=gnutls setup
$ make build
```

To build with *OpenSSL* or *LibreSSL*:

```
$ make SOCKET=openssl setup
$ make build
```

It is is possible to build *AWS* in debug mode by setting *DEBUG* make's variable:

```
$ make DEBUG=true 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
```

2.4 Building on cross-platforms

To build for a cross platform the TARGET makefile variable must be set with the cross toolchain to be used. The value must be the triplet of the toolchain to use.

For example, to build on VxWorks:

```
$ make TARGET=powerpc-wrs-vxworks setup build
```

Note that on cross-environment one need to build the demos manually. See demos/README.

2.5 Demos

AWS comes with some ready to use demos. The demos are a good way to learn how to use *AWS*.

Here are a short description of them:

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.

autobahn
A demo to validate the WebSocket implementation against the autobahn test suite.

cert
A secure server using a Certificate Authority and validating clients with certificate. This is the highest security level possible.

com
Two simples program that uses the *AWS* communication service.

dispatch
A simple demo using the dispatcher facility. see *URI dispatcher*.

hello_world
The famous Hello World program. This is a server that will always return a Web page saying 'Hello World!'.

hello_wsdl
An hello world kind of application using a WSDL document for describing the messages format.

hotplug
A simple test for the hotplug feature.

https
A simple secure server enforcing TLS 1.2 protocol to be used by the Web Browser. This demo also uses a signed server's key and proper setup hand over the password to the secure layer.

interoplab
A WSDL based demo that test most of the *SOAP* features.

jabber_demo
A simple Jabber command line client to check the presence of a JID (Jabber ID). This uses the Jabber API, see *AWS.Jabber*.

multiple_sessions
A demo of two embedded servers using different sessions.

res_demo
A demo using the resource feature. This Web Server embedded a *PNG* image and an *HTML* page. The executable is self contained.

runme
An example that test many *AWS* features.

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

soap_disp
Like above but use a *SOAP* dispatcher.

soap_vs
A client and server that implement seven *SOAP* procedures for testing purpose.

split
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.

test_ldap
A simple *LDAP* demo which access a public *LDAP* server and display some information.

test_mail
A simple application that send a set of *SMTP* messages with different kind of attachments.

text_input
A simple demo which handle textarea and display the content.

vh_demo
Two servers on the same machine… virtual hosting demo. see *Virtual host dispatcher*.

web_block
A simple Web Bock based counter.

web_block_ajax
As above but using also *Ajax*.

web_block_ajax_templates
As above but using also the *templates2ada* tool which create a tight coupling between the web templates and the *Ada* code.

web_elements
A driver to browse the Web Elements library and see some examples.

web_mail
A simple Web Mail implementation that works on a *POP* mailbox.

websockets
A simple WebSocket demo.

wps
A very simple static Web page server based on *AWS.Services.Page_Server*. see *Static Page server*.

ws
A static Web page server and push enabled server.

ws_candy
A WebSocket demo using many of the WebSocket's features.

zdemo
A simple demo of the Gzip content encoding feature.

For build instructions see `demos/README`.

2.6 Installing

When the build is done you must install *AWS* at a specific location. The target directory is defined with the *prefix* `makefile.conf` variable. The default value is set to the compiler root directory. Note that the previously installed version is automatically removed before installing the new one. To install:

```
$ make install
```

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

```
$ make prefix=/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 *prefix* to point to *GNAT* root directory:

```
$ make prefix=/opt/gnatpro/6.1.1 install
```

Now you are ready to use *AWS* !

CHAPTER

THREE

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*.

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/static*. 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 (setting *ADA_INCLUDE_PATH* and *ADA_OBJECTS_PATH*) for all external libraries that you will be using.

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");

```

(continues on next page)

(continued from previous page)

```
  for Object_Dir use "obj";

end Simple;
```

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 *AWS.Client*.

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:

- declare the HTTP Web Server:

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

- Start the server

  You need to start the server before using it. This is done by calling *AWS.Server.Start* (see AWS.Server):

```
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.
```

The procedure *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.thtml**'), see *Status page*.

The administrative page returns many information about the server. It is possible to configure the server via two configuration files see *Configuration options*.

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* or *GNUTLS* 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 AWS.Server. And there is also the version using a *Config.Object* which is the most generic one.

- 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 *AWS.Status*.).

  For example, to know what URI has been asked:

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

if URI = "/whatever" then
```

(continues on next page)

(continued from previous page)

```
   ...
end if;
```

Then this function should return an answer using one of the constructors in *AWS.Response* (see *AWS.Response*.). 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 := AWS.Status.URI (Request);
  begin
     if URI = "/hello" then
        return AWS.Response.Build ("text/html", "<p>Hello world !");
```

(continues on next page)

(continued from previous page)

```
      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 returns 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 *Callback procedure*, 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 *Building an AWS server*.

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 *Status page*.

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 *Configuration options*.

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 *Authentication*.

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 *AWS.Response*.

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 exit 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 *AWS.Resources.Streams.Memory.ZLib*. 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 *Split pages*.

3.3.14 Response a from pipe stream

The response sent to the server is read from the output of an external application. This kind of stream can be used to avoid writing a temporary file into the hard disk. For example it is possible to return an archive created with the *tar* tool without writing the intermediate tar achive on the disk.

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. `program_name.ini` is looked first in the application's directory and then in the current working directory. This is only supported on platforms where *Ada.Command_Line* is implemented. So, on **VxWorks** only `aws.ini` is parsed.

Furthermore, it is possible to read a specific configuration file using the *AWS.Config.Ini.Read* routine. See *AWS.Config.Ini*.

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_Password (string)

This is the password used to call the administrative page. The password can be generated with `aws_password` (the module name must be *admin*):

```
$ aws_password admin <password>
```

Admin_URI (string)

This is the URI to call the administrative page. This can be used when calling *AWS.Server.Start*. The default is **<not-defined>**.

Case_Sensitive_Parameters (boolean)

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

Client_Certificate (string)

Set the certificate file to be used with the secure slients. The default is **cert.pem**. The certificates must be in *PEM* format.

Server_Certificate (string)

Set the certificate file to be used with the secure servers. The default is **aws-server.crt**. 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.

Server_Key (string)

Set the RSA key file to be used with the secure servers. The default file is **aws-server.key**.

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**.

Cipher_Priorities

Values are dependent on the actual secure layer (OpenSSL or GNUTLS). It is used to specify the session's handshake algorithms and options.

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 **7.0** 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.

Close_On_Exec (boolean)

Set the socket close on exec policy. If set to True the child process spawned from the server won't get access to the server's sockets (listening and connection with clients). That is, all inherited sockets are closed when a child process is executed. platforms. The default is **|CLOSE_ONE_EXEC|**.

Config_Directory (string)

The directory in which AWS keeps some configuration parameters. The default is **.config/ada-web-srv**.

CRL_File (string)

This configuration option must point to a filename containing a CRL (Certificate Revocation List). This will make it possible to control client connecting to the server. The default values is **<not-defined>**.

Directory_Browser_Page (string)

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

Disable_Program_Ini (boolean)

Specify whether the configuration file `program_name.ini` should be parsed or not. If this option is set to FALSE the program specific configuration file won't be parsed. This may be useful if another application is using such a file and cannot be shared. This setting is expected to be set in `aws.ini` before the `program_name.ini` file is parsed. The default value is **FALSE**.

Down_Image (string)

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

Error_Log_Activated (boolean)

A boolean to enable or disable the error log. By default the error log activation is set to **FALSE**.

Error_Log_Filename_Prefix (string)

This is to set the filename prefix for the error log file. By default the error 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 **TRUE**.

Check_Certificate (boolean)

If set to True the server or client will reject all SSL connections if the peer did not provide a valid certificate. The default value is **TRUE**.

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 **2.0** 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**.

HTTP2_Activated (boolean)

Whether the HTTP2 protocol is to be activated for the server. The default value is **TRUE**.

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_Activated (boolean)

A boolean to enable or disable the standard log. By default the standard log activation is set to **FALSE**.

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 compartible format, otherwise log file would be in Extended format. For more information see *Server Log*.

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_Image (string).

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

Max_Concurrent_Download (positive)

Control the maximum number of parallel downloads accepted by the download manager. The default value is **25**.

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 = <main server thread> + <max connections> [+ <session>]
```

Note: [...] means optional value Add 1 only if the session feature is activated. This is due to the session cleaner task.

Max_POST_Parameters (positive)

The maximum number of POST parameters supported by AWS. The default value is **100**.

Max_WebSocket (positive)

The maximum number of WebSocket that can be opened simultaneously in AWS. Above this value AWS will try to close timed-out WebSockets (see *WebSocket_Timeout*). The default value is **512**.

Max_WebSocket_Handler (positive)

The maximum number of message to handle simultaenously. The default value is **2**.

MIME_Types (string)

The name of the file containing the MIME types associations. The default file name is **aws.mime**.

A note from the Fedora package maintainers

An up-to-date list of MIME types (`/etc/mime.types`) is available from the mailcap package.

Receive_Timeout (duration)

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

Reuse_Address (boolean)

Set the socket reuse address policy. If set to True the server will be able to bind to a socket that has just been released without the need of waiting. Enabling this feature may introduce security risks on some platforms. The default is **FALSE**.

Security_Mode (string)

Set the security mode to use for the secure connections. The default mode is **TLS**. See *AWS.Net.SSL*.

Send_Buffer_Size (positive)

This is the socket internal buffer used for sending data to the clients. The default is **0**.

Send_Timeout (duration)

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

Server_Header (string)

The value to be used for the HTTP Server header. The default is **AWS (Ada Web Server) v25.1.0**. If the value is set to the empty string, the server header is not sent.

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_Priority (natural)

Priority of the task handling the HTTP protocol. The default is Default_Priority.

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**.

Service_Priority (natural)

Priority of the tasks used by optional services like SMTP Server, Server Push, Jabber and the Transient Page cleaner. The default is **Default_Priority**.

Session (boolean)

Whether the session support must be activated or not. The default is **FALSE**.

Session_Name (string)

The name of the cookie session. This can be used to support sessions for multiple servers embedded into the same executable. The default is **AWS**.

Session_Id_Length (positive)

The length of the session id in characters. The default is **11** characters.

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.

Session_Cleaner_Priority (natural)

Priority of the task cleaning the session data. The default is **Default_Priority**.

Status_Page (string)

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

TCP_No_Delay (boolean)

This control the server's socket delay/no-delay option. This option should be used for applications that require lower latency on every packet sent. The default is **FALSE**.

TLS_Ticket_Support (boolean)

Specify whether the TLS ticket support is activated or not. The default value is **FALSE**.

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 (duration)

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.

Trusted_CA (string)

This must point to the file containing the list of trusted Certificate Authorities. The CA in this file will be used to verify the client certificate validity. The default values is **<not-defined>**.

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 **<not-defined>**.

User_Agent (string)

The value to be used for the HTTP User_Agent header. The default value is **AWS (Ada Web Server) v25.1.0**. If the value is set to the empty string, the User_Agent header is not sent.

WebSocket_Message_Queue_Size (positive)

This is the size of the queue containing incoming messages waiting to be handled by one of the task, see Max_WebSocket_Handler above. The default value is **10**.

WebSocket_Origin (string)

This is a regular expression which can be used to handle WebSockets originating from a specific domain. By default AWS handles WebSockets from any origins.

WebSocket_Priority (natural)

Priority of the task handling the WebSockets. The default is **Default_Priority**.

WebSocket_Timeout (duration)

A number of seconds after which a WebSocket without activity is considered timed-out and can be elected to be closed if the maximum number of sockets opened has been reached. (see *Max_WebSocket*). The default is **28800.0**.

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. It works by creating transparently a session ID where it is possible to insert, delete and retrieve session data using a standard Ada API (see *AWS.Session*.). Session data are key/value pair each of them being strings. These sessions data are kept on the server, for client side state management see *HTTP state management*.

- 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 HTTP state management

AWS provides a full implementation of RFC 2109 via the *AWS.Cookie* package. Using this package you set, get and expire client-side HTTP cookies.

First we set a cookie:

```
declare
   Content : AWS.Response.Data;
begin
   AWS.Cookie.Set (Content,
                   Key      => "hello",
                   Value    => "world",
                   Max_Age  => 86400.0);
end;
```

Here we set the cookie *hello* with the value *world*, and we tell the client to expire the cookie 86400 seconds into the future.

Getting the cookie value back is equally simple:

```
declare
   Request : AWS.Status.Data
   --  Assume that this object contain an actual HTTP request.
begin
   Put_Line (AWS.Cookie.Get (Request, "hello"));
   --  Output 'world'
end;
```

Had the cookie *hello* not existed, an empty *String* would've been returned.

In some cases it might be of value to know if a given cookie exists, and for that we have the *Exists* function available:

```
declare
   Request : AWS.Status.Data
   --  Assume that this object contain an actual HTTP request
begin
   if AWS.Cookie.Exists ("hello") then
      Put_Line ("The 'hello' cookie exists!");
   end if;
end;
```

Note that *Exists* doesn't care if the cookie contains an actual value or not. If a cookie with no value exists, *Exists* will return *True*.

And finally we might wish to tell the client to expire a cookie:

```
declare
   Content : AWS.Response.Data;
begin
```

(continues on next page)

(continued from previous page)

```
   AWS.Cookie.Expire (Content,
                      Key  => "hello");
end;
```

The Cookie package provide *Get* functions and *Set* procedures for *String*, *Integer*, *Float* and *Boolean* types, but since cookies are inherently strings, it's important to understand what happens when the cookie *String* value can't be converted properly to either *Integer*, *Float* or *Boolean*.

So if either conversion fails or the cookie simply doesn't exist, the following happens:

- For *Integer*, the value 0 is returned
- For *Float*, the value 0.0 is returned.
- For *Boolean*, the value *False* is returned. Note that only the string 'True' is *True*. Everything else is *False*.

For more information see *AWS.Cookie*.

3.7 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:

- 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.

- Check authentication

  When an authentication as 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.

3.8 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 *Configuration options*.

The uploaded files are removed after the user's callback. This is done for security reasons, if files were not removed it would be possible to fill the server hard disk by uploading large files to the server. This means that uploaded files must be specifically handled by the users by either copying or renaming them.

AWS will also setup the form parameters as usual. In the above example there is two parameters (see *Form parameters*).

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 simple filename (no path information) of the file on the client side.

go

The value is "Send File"

3.9 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 *SOAP*.

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.

3.9.1 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.

3.9.2 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 *Hotplug module*.

3.10 Hotplug module

An **Hotplug module** is a module that can by 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 *Building an AWS server*. 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.

3.10.1 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 *aws_password* is provided with *AWS* to generate such password. Usage is simple:

```
$ aws_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.

3.10.2 Hotplug module - creation

Here is how to create an Hotplug module:

- First you create a standard Web server, see *Building an AWS server*:

  ```
  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.

- Then you register the Hotplug module to the main server, see *AWS.Client.Hotplug*:

  ```
  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 *Hotplug module - server activation*. 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*.

- 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 *Hotplug module - server activation*.

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

3.11 Server Push

This protocol is obsolescent, it is hightly recommended to use the WebSockets now. See *WebSockets*.

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:

- 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.

- 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.

- 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.

- 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.

- Build the Push Server

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

- 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.

- 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.

3.12 Working with Server sockets

With *AWS* is 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 interractives 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 form the callback function. A user instead of replying immediatly decides to take away the socket from the server. The first step is to record the connection socket 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 *Server Push*.

- Give back the socket to the server

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

3.13 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 *AWS.Log* 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 compartible 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:

date
: Date at which transaction completed

time
: Time at which transaction completed

time-taken
: Time taken for transaction to complete in seconds

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-<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 compartible format. See *AWS.Server.Log.Start_Error* and *AWS.Server.Log.Stop_Error*.

For the full set of routines supporting the log facility see *AWS.Server.Log* .

3.14 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 *Building*. You must also have installed the *OpenSSL* or *GNUTLS* libraries on your system. If this is done, you can continue reading this section.

3.14.1 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 **aws-server.crt**. This certificate has been created with the *OpenSSL* toolset. It is valid until year 2034 and is meant only for the demos. Furthermore, 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*:

With a key and certificate files:

```
AWS.Server.Set_Security
  (WS,
   Key_Filename         => "aws-server.key",
   Certificate_Filename => "aws-server.crt");
```

Or with a self-contained certificate:

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

Or using the *server_certificate* configuration parameter, see *Configuration options*.

Alternatively it is possible to set the default SSL options before starting the server:

```
AWS.Net.SSL.Initialize_Default_Config
  (Security_Mode       => AWS.Net.SSL.TLS_Server,
   Server_Certificate  => "aws-server.crt",
   Server_Key          => "aws-server.key");
```

Or by using a configuration object:

```
HTTP   : Server.HTTP;
--  The SSL Web Server

Config : AWS.Net.SSL.Config;
--  SSL configuration object to be passed to server

AWS.Net.SSL.Initialize
  (Config,
   Security_Mode       => AWS.Net.SSL.TLS_Server,
   Server_Certificate  => "aws-server.crt",
   Server_Key          => "aws-server.key",
   Trusted_CA_Filename => "/usr/local/etc/example/additional-ca.crt");

Server.Set_SSL_Config (HTTP, Config);
```

3.14.2 Verify callback

First note that it is not necessary to use such callback to verify the certificate validity, see *Using a Certificate Authority*.

This callback will receive the client certificate as sent during SSL handshake between the server and the client. The certificate information can be checked for logging purpose or to impose some restriction. Generally this callback should return the value from *AWS.Net.SSL.Certificate.Verified*, see *AWS.Net.SSL.Certificate*.

The Verified status of the certificate is the one that has been issued by the SSL implementation during certificate verification and can generally be trusted.

3.14.3 Self-signed certificate

Creating a server certificate

The goal here is not to replace the *OpenSSL* documentation but just to present one way to create a self signed certificate for an *HTTPS* test server. Note that *GNUTLS* offers similar tools to generate certificates.

Generate a RSA key:

```
$ openssl genrsa -rand <filename> -out aws-server.key
```

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

Generate the certificate:

```
$ openssl req -new -x509 -days 730 -key aws-server.key -out aws-server.cert
```

Create a single self contained file (optional):

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

A this point you can use `aws.pem` with your server or the separate `server.key` and `server.crt` files.

It is also possible to sign the server's key. In this case the key won't be in plain text but will require to setup a password on the server code for the key to be decoded. See routine Set_Password_Callback in AWS.Net.SSL.Certificate.

Generate a crypted RSA key:

```
$ openssl genrsa -aes128 -passout pass:<PASSWORD> -out aws-server.key
```

Creating a client certificate

A certificate can also be used on a Web browser and passed to the server to have a strong client authentication. A client certificate must be *PKCS12*. The steps to generate such certificate are:

Generate a RSA key:

```
$ openssl genrsa -des3 -out aws-client.key
```

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

Generate the certificate:

```
$ openssl req -new -x509 -days 730 -key aws-client.key -out aws-client.cert
```

Create the corresponding PKCS12 certificate:

```
$ openssl pkcs12 -export -clcerts -in aws-client.cert -inkey aws-client.key -out client.
↪p12
```

3.14.4 Using a Certificate Authority

In this section we will use a Certificate Authority to signed the server certificates and the client certificates. Using this method is required if the server must ensure that only clients with a valid certificate will be able to connect to the server. The server will verify that the client certificate received has been signed by a known Certificate Authority.

Note that these checks are happening during the SSL handshake, so before the user's callback.

For this to work the following configuration options must be used:

Exchange_Certificate
To request that the client certificate be sent.

Trusted_CA
The file containing the certificate of the Certificate Authority we trust. The CA which has signed the client's certificate.

Check_Certificate
If the certificate received from the client is not valid the server will reject the connection. If this is not set, we can still validate the client's certificate in the verify callback, see *Verify callback* and for example log the connecting users.

Initializing the Certificate Authority

First the Certificate Authority must be initialized on the computer. This is heavily dependent on the actual Operating System used, describing this part is out of scope of this document.

On GNU/Debian the default setup (see default_ca in `/etc/ssl/openssl.cnf`) can be used to create a **demo** Certificate Authority locally to test this feature:

```
$ mkdir demoCA
$ mkdir demoCA/newcerts
$ touch demoCA/index.txt
$ echo ABCC > demoCA/serial
$ echo 01 > demoCA/crlnumber
```

Creating the Certificate Authority

Generate a RSA key:

```
$ openssl genrsa -out private-ca.key 1024
```

Generate the certificate signing request:

```
$ openssl req -new -key private-ca.key -out private-ca.csr
```

During this step you'll be asked for information about the CA (Country, State or Province, Organization Name...).

Create the CA certificate:

```
$ openssl x509 -req -days 365 -in private-ca.csr -signkey private-ca.key -out private-ca.
↪crt
```

This certificate will be used by AWS as the trusted CA, see *Configuration options*.

Creating a CA signed server certificate

Generate a RSA key:

```
$ openssl genrsa -out aws-server.key 1024
```

Generate the certificate signing request:

```
$ openssl req -new -key aws-server.key -out aws-server.csr
```

During this step you'll be asked for information about the server (Country, State or Province, Common Name...). Note that the Organization Name here must match the one from the CA and the Common Name should be the server fully qualified domain name.

Create the server certificate, signed it with our CA:

```
$ openssl ca -in aws-server.csr -cert private-ca.crt -keyfile private-ca.key -out aws-
↪server.crt
```

Create a single self contained file (optional):

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

Creating a CA signed client certificate

Generate a RSA key:

```
$ openssl genrsa -des3 -out aws-client.key 1024
```

Generate the certificate signing request:

```
$ openssl req -new -key aws-client.key -out aws-client.csr
```

During this step you'll be asked for information about the client (Country, State or Province, Common Name...). Note that the Organization Name here must match the one from the CA and the Common Name should be the client's one.

Create the client certificate, signed it with our CA:

```
$ openssl ca -in aws-client.csr -cert private-ca.crt -keyfile private-ca.key -out aws-
↪client.crt
```

Create the corresponding PKCS12 certificate:

```
$ openssl pkcs12 -export -clcerts -in aws-client.crt -inkey aws-client.key -out aws-
↪client.p12
```

Creating a Certificate Revocation List (CRL)

A Certificate Revocation List is used to revoke some client's certificates. Those clients won't be able to connect to the secure server anymore. Using the CA created above the following commands can be used to create a CRL.

Revoke the certificate:

```
$ openssl ca -cert private-ca.crt -keyfile private-ca.key -revoke aws-client.crt
```

Generate the CRL:

```
$ openssl ca -cert private-ca.crt -keyfile private-ca.key -gencrl -out crl.pem -crldays␣
↪30
```

The file `crl.pem` is the one to install on the server using the CRL_File configuration option, see *Configuration options*. This file contains the list of all revoked certificates for the corresponding CA.

3.14.5 Security level

This table summarize the security level achieved with different settings of the security oriented configuration parameters.

Security	SSL	Exchange Certificate	Check Certificate	Trusted CA
Data between the client and the server are encrypted.	Yes	No	No	No
Client can be identified, it is still possible to access the server without having a certificate.	Yes	Yes	No	No
Client are identified, a certificate is required. The verification of the validity is up to the application using the verify callback.	Yes	Yes	Yes	No
Client are identified and verified, the certificate must have been signed by a Certificate Authority. It is not possible to access the server without a valid certificate.	Yes	Yes	Yes	Yes

3.14.6 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.

3.15 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 *AWS.Exceptions*.

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.tmplt` 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 AWS.Server and *AWS.Exceptions*.

3.16 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 *AWS.Net.Log* and *AWS.Net.Log.Callbacks*.

3.17 Client side

3.17.1 Client

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 *AWS.Client*.

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 a non-secure Web page which is http://www.mydomain.net. The code to do so is:

```
Data := Client.Get
          (URL => "http://www.mydomain.net");
```

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 *demos* 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.

3.17.2 Secure Client

Most page are now using HTTP/S to ensure proper security when retrieving data from the Internet.

The client API will automatically detect that the connection require SSL by checking if the URL starts with *https://*. The above example will then be:

```
Data := Client.Get
        (URL => "https://www.mydomain.net");
```

The secure layer will be initialized with AWS's default values. As for the server it is possible to configure the SSL client layer by setting either the default configuration or using an SSL configuration object.

The default configuration must be set before using AWS's client API:

```
AWS.Net.SSL.Initialize_Default_Config
  (Security_Mode        => Net.SSL.TLS_Client,
   Client_Certificate   => "cert.pem",
   Check_Certificate    => True);
```

Or using a connection object initialized with an SSL configuration object:

```
Config : AWS.Net.SSL.Config;
--  SSL configuration object to be passed to server
```

(continues on next page)

(continued from previous page)

```
Data   : AWS.Response.Data;

AWS.Net.SSL.Initialize
  (Config,
   Security_Mode       => AWS.Net.SSL.TLS_Client,
   Client_Certificate  => "my-certificate.pem");

AWS.Client.Create (Connection,
                   Host       => "https://www.mydomain.net",
                   SSL_Config => Config);

AWS.Client.Get (Connection, Data);
```

CHAPTER

FOUR

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 *AWS.Templates*. This means that you can use the default directory template or provide your own.

see *AWS.Services.Directory* for the complete specification and description of this service.

4.2 Dispatchers

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

Working with a dispatcher is quite easy:

- Create a new dispatcher by inheriting from the service you want to build.
- Register a set of action based on rules (strings or 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 instead in *AWS.Dispatchers.Callback* because this is a basic service needed for the server itself. It is referenced here for documentation purposes but an AWS server can be built without using it.

see *AWS.Dispatchers.Callback* for the complete specification.

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 *AWS.Services.Dispatchers.Method* for the complete specification.

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 *AWS.Services.Dispatchers.URI* for the complete specification.

4.2.4 Virtual host dispatcher

This is a dispatcher based on the hostname. A different callback procedure can be registered for specific hostnames. This enables support for 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 hostname in the client's request.

see *AWS.Services.Dispatchers.Virtual_Host* for the complete specification.

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 *Transient Pages*.

4.2.6 Timer dispatcher

A timer dispatcher can be used to call different callback routines depending on the current date and time. Such a dispatcher is composed of a set of activation *Period`s . 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 kinds of *Period`s 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 also provides a *SOAP* specific dispatcher. This is a way to automatically route HTTP requests or *SOAP* requests to different callback routines.

see *SOAP helpers* for more information. see *SOAP.Dispatchers.Callback* for the complete specification.

4.3 Static Page server

This service is a ready to use static page server callback. It is possible to use this service to build a simple static page server; this is 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 execute it to serve *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 *Directory browser*.

Note that this service uses two template files:

aws_directory.thtml
: The template page used for directory browsing. See see *AWS.Services.Directory* 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 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.

see *AWS.Services.Page_Server* for the complete specification.

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 may or may not be requested by the client (by clicking on the corresponding link), in either case 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 does automatically. Each transient page must be registered into the service, then a specific routine named *Get_URI* can be used to create a unique *URI* on this server. see *AWS.Services.Transient_Pages*.

A transient pages dispatcher can be used to build a server supporting transient pages. see *Transient pages dispatcher*.

4.5 Split pages

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

The *AWS*'s split page feature can automatically do this 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 into (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 *AWS.Services.Split_Pages.Alpha*.

Alpha.Bounded
: Same as the alpha splitter, but pages larger than a Max_Per_Page value are further split. A secondary index is generated that gives the various pages for a given letter. see *AWS.Services.Split_Pages.Alpha.Bounded*.

Uniform
: Split into pages of length Max_Per_Page (except the last one). This corresponds to the default service in the Split_Pages package. see *AWS.Services.Split_Pages.Uniform*.

Uniform.Alpha
: Same as the uniform splitter, but additionally builds an alphabetical secondary index from a key field. see *AWS.Services.Split_Pages.Uniform.Alpha*.

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 *AWS.Services.Split_Pages.Uniform.Overlapping*.

Using the spliter abstract interface it is possible to build a customized splitter algorithm. see *AWS.Services.Split_Pages*.

4.6 Download Manager

A server that needs to handle lot of large downloads can run out of connections to answer the standard Web pages. One 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 *AWS.Dispatchers.Callback*).

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 the comments in see *AWS.Services.Download*.

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 clients being currently served).

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.thtml

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
```

(continues on next page)

(continued from previous page)

```
<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

A note from the Fedora package maintainers

The bundled third-party Javascript files are not included in the package, because they are seemingly unmaintained and their licensing is unclear. Many Javascript libraries of varying complexity can be found on the Web.

AWS provides some components to help in creating nice looking Web interfaces. It is possible to browse these 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 that can be easily reused in other applications. The first three 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 in 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 in 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 in *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 easier. Two actions are provided, one for replacing and another for clearing part of a web page's content.

Steps to do Ajax

What are the steps to do *Ajax* ?

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

- Include the AWS/Ajax support file

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

- Create the Web widgets/forms

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

- Create the Web area

  Using some *HTML* 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.

- Name the widgets/forms/area using the id="name" attribute

  Give a different name to the widgets using id="name". This name will later be 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()"*.

- Add the proper event/action to the widgets using the AWS/Ajax templates

  This is the interresting 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*, but a simple approach that works well with the *AWS/Ajax* templates.

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 *XML based Ajax*.

Let's have a very simple example:

- The AWS/Ajax runtime support

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

  Must be included into every Web page's <*head*> tag.

- 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 happens when the button is clicked:

  - the "/onclick$clickme" HTTP request is sent to the server
  - asynchronously waits for the answer, when received places 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!");
   ...
```

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=>*mall_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.

XML based Ajax

In many cases you'll want 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 specified 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.

- refresh the current page:

```
<refresh/>
```

Refresh the current page as if the Web Browser refresh button was pressed.

- 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.

- make an entry disabled or enabled:

```
<disabled id="item_id" value="true/false"/>
```

- make an entry read-only or writable:

```
<read_only id="item_id" value="true/false"/>
```

- reset a form:

```
<reset id="form_id"/>
```

Here is an example of such XML document:

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

To register an *Ajax* action to a specific tag id a macro can be used. It is named *JS_ACTION* and defined in `ajax_api.tjs`. The usage is similar to what is described in the previous section (see *Basic Ajax support*) except that in this case we use a macro instead of an include file and 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
```

  Must be included in every Web page's *<head>* tag.

- The AWS/Ajax API:

```
@@INCLUDE@@@ ajax_api.tjs
```

  Must be included at least once during an application life-time. It gives access to the *JS_ACTION* macro.

- 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:

```
@_JS_ACTION(onclick, clickme)_@
```

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

- the "/onclick$clickme" HTTP request is sent to the server
- asynchronously waits for the XML answer, when received parses 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:

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

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

```
<?xml version="1.0" encoding="UTF-8" ?>
<response>
  <replace id="item_id">
    <![CDATA[ *HTML CODE HERE* ]]>
  </replace>
</response>
```

Advanced Ajax

Finally, if this is not enough because you need to use some specific *JavaScript* code, *AWS* provides a macro named *BIND_JS* to add an event to a specific widget, the action being the name of a *JavaScript* routine.

This macro together with the `aws_func_replace.tjs`, `aws_func_clear.tjs` templates and the `JS_ACTION` macro 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`.

Suppose you want to clear a widget, change the content of another one and call one of your specific *JavaScript* routines when clicking on a button. It is not possible to have mutiple *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:

```
@_BIND_JS(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 and to the *JS_ACTION* macro. This is another way to chain multiple actions for a single event.

Note that all *AWS/Ajax* templates and the `ajax_api.tjs` file have a set of comments at the start explaining in details the usage of each parameter.

4.8 Web Blocks

The *AWS.Services.Web_Block* hierarchy contains an API useful for keeping context on Web pages. It has been designed to be able to split a Web application into a set of independent blocks that can be put together in the same Web page. The context is then useful as it is passed and known by each individual block. Note that this is different than the session as a session is global to the current Web browser whereas the context can be different for each individual web pages opened.

Instead of parsing a whole page using the *AWS.Templates* API the web blocks are registered independently using *AWS.Services.Web_Block.Registry*. The block is registered together with its templates and the callback to use to get user's data for this specific block with the given context.

So using this API, instead of having a set of callbacks returning an *AWS.Response.Data* and where the final rendering is to be done by the client code, we have a set of callbacks that returns a *Translate_Set*. The client just has to fill the set with the data corresponding to the actual request and possibly using the context. The final rendering is done by the provided services in *Web_Block.Registry*.

Note that all Web pages must also be registered into the registry to ensure that the context identification is properly kept. The context identification is injected into the Web pages transparently for the end-user when using *Ajax*.

4.8.1 Web Block example

Let's have a simple example, a page containing a single block with a tag (@_COUNTER_@) which is incremented by one each time it is used. The code can be found in `demos/web_block`.

First create the following HTML fragment and place it into `counter.thtml`:

```
<p>@_COUNTER_@</p>
```

Then create the main page and place it into `page.thtml`. The important part is the @_CTX_WB_@ tag which is passed to the link. This tag is the context identifier, it must be passed to each request. Note that this is automatically done when using the *Ajax* framework (see *Web Block and Ajax*):

```
<html>
  <head>
    <title>Main Page</title>
  </head>
  <body>
    <p>This is the main page, bellow is a simple counter</p>
    <p>@_COUNTER_@</p>
    <a href="/?CTX_WB=@_CTX_WB_@>Next</a>
  </body>
</html>
```

The *Web_Callbacks* package contains the application callbacks:

```
with AWS.Response;
with AWS.Status;
with AWS.Templates;
with AWS.Services.Web_Block.Context;

package Web_Callbacks is

   use AWS;
   use AWS.Services;

   function Main (Request : in Status.Data) return Response.Data;
   --  Main callback which handle the home page

   procedure Counter
     (Request      : in              Status.Data;
      Context      : not null access Web_Block.Context.Object;
      Translations : in out          Templates.Translate_Set);
   --  The callback handling the counter web block

end Web_Callbacks;
```

Last part is to actually implement the *Counter* callback. Here is a possible implementation making use of the context to keep the counter state:

```
with AWS.Utils;
with AWS.Messages;
with AWS.MIME;
with AWS.Services.Web_Block.Registry;

package body Web_Callbacks is

   -------------
   -- Counter --
   -------------

   procedure Counter
     (Request      : in              Status.Data;
      Context      : not null access Web_Block.Context.Object;
      Translations : in out          Templates.Translate_Set)
   is
      N : Natural := 0;
   begin
      if Context.Exist ("N") then
         N := Natural'Value (Context.Get_Value ("N"));
      end if;

      N := N + 1;
      Context.Set_Value ("N", Utils.Image (N));

      Templates.Insert
        (Translations, AWS.Templates.Assoc ("COUNTER", N));
   end Counter;
```

(continues on next page)

(continued from previous page)

```
   ----------
   -- Main --
   ----------

   function Main (Request : in Status.Data) return Response.Data is
      URI : constant String := Status.URI (Request);
   begin
      return Web_Block.Registry.Build
        (Key          => URI,
         Request      => Request,
         Translations => Set);
   end Main;

end Web_Callbacks;
```

Finally, we write the main procedure:

```
with Ada.Text_IO;

with AWS.Server;
with AWS.Services.Web_Block.Registry;

with Web_Callbacks;

procedure Web_Block is

   use Ada;
   use AWS;
   use AWS.Services;

   HTTP : AWS.Server.HTTP;

begin
   --  First we register the main page and the counter block

   Services.Web_Block.Registry.Register ("/", "page.thtml", null);

   Services.Web_Block.Registry.Register
     ("COUNTER", "counter.thtml",
      Web_Callbacks.Counter'Access, Context_Required => True);

   --  Then we just start the server

   Server.Start (HTTP, "web_block", Web_Callbacks.Main'Access);

   Text_IO.Put_Line ("Press Q to terminate.");

   Server.Wait (Server.Q_Key_Pressed);

   Server.Shutdown (HTTP);
end Web_Block;
```

Compile and run the server. Then connect to the server and click on next. The counter will be incremented by one each

time.

4.8.2 Web Block and Ajax

The Web Block framework has really been designed to be used with *Ajax*. It is the only way to gain the full power of the Web Block framework.

For the complete code, see *demos/web_block_ajax*.

When using *Ajax* it is not needed to explicitly pass the context identification to every link. This is done automatically by the framework. So the main page will look like this:

```
@@INCLUDE@@ ../../web_elements/javascripts/ajax_api.tjs
<html>
  <head>
    <title>Main Page</title>
    @@INCLUDE@@ ../../web_elements/javascripts/aws.tjs
  </head>
  <body>
    <p>This is the main page, bellow is a simple counter</p>
    @_WIDGET_COUNTER_@
  </body>
</html>
```

The counter widget is on `widget_counter.thtml`:

```
<!-- implementation of a simple counter widget -->
<p><div id="counter">@_COUNTER_@</div></p>
<a id="next" href="/">Next</a>
@_JS_ACTION(onclick, next)_@
```

For the *Ajax* part, see *Ajax*.

We now have one more register call for registering the *next* button *Ajax* callback, and a callback named *Widget_Counter* for displaying the block:

```
Services.Web_Block.Registry.Register
  ("WIDGET_COUNTER", "widget_counter.thtml",
   Web_Callbacks.Widget_Counter'Access);

Services.Web_Block.Registry.Register
  ("/onclick$next", "r_widget_counter.txml",
   Web_Callbacks.Onclick_Next'Access,
   Content_Type     => MIME.Text_XML,
   Context_Required => True);
```

The *next Ajax* button is using an XML based response which is defined in `r_widget_counter.txml`:

```
<?xml version="1.0" encoding="UTF-8" ?>
<response>
  <replace id="counter">@_COUNTER_@</replace>
</response>
```

The *Widget_Counter* callbacks just have to set the *COUNTER* tag variable to the corresponding value. This is used to display the block. The *Ajax* callback *Onclick_Next* has to increment the counter and set the *COUNTER* tag variable, a simple implementation is:

```
procedure Onclick_Next
  (Request      : in              Status.Data;
   Context      : not null access Web_Block.Context.Object;
   Translations : in out          Templates.Translate_Set)
is
   N : Natural := 0;
begin
   if Context.Exist ("N") then
      N := Natural'Value (Context.Get_Value ("N"));
   end if;

   N := N + 1;

   Context.Set_Value ("N", Utils.Image (N));

   Templates.Insert
     (Translations, Templates.Assoc ("COUNTER", N));
end Onclick_Next;
```

The framework will then call *Onclick_Next* when pressing the *Next* button. This routine increments N by one sending back a response based on *r_widget_counter.txml*. Finally, the client browser will parse this XML response and do the corresponding actions.

4.8.3 Web Block and templates2ada

For the complete code, see *demos/web_block_ajax_templates*.

It is possible to use the *Templates_Parser's templates2ada* tool for generating the callbacks register calls. This ensures that all tags on the application Web Pages have a corresponding callback.

The code is almost identical to the standard *Ajax* example above. The main difference is that we need to use a naming convention for the blocks. This way we can generate automatically the corresponding callbacks using a template. A common convention is to add *LAZY_* as prefix for the name of the blocks. With this convention the main page template is:

```
@@INCLUDE@@ ../../web_elements/javascripts/ajax_api.tjs
<html>
  <head>
    <title>Main Page</title>
    @@INCLUDE@@ ../../web_elements/javascripts/aws.tjs
  </head>
  <body>
    <p>This is the main page, bellow is a simple counter</p>
    @_LAZY_WIDGET_COUNTER_@
  </body>
</html>
```

We need also modify the standard `templates.tads` as distributed with the *Templates_Parser*. Here is the interesting part:

```
@@SET@@ PACKAGE = WBlocks

...
```

(continues on next page)

(continued from previous page)

```
with AWS.MIME;
with AWS.Services.Web_Block.Registry;
with Web_Callbacks;

@@TABLE@@
with @_PACKAGE_@.@_CAPITALIZE:REPLACE_ALL(\\./_):BASENAME_@;
@@END_TABLE@@

package body @_PACKAGE_@ is

   use AWS;

   package body Lazy is

      --------------
      -- Register --
      --------------

      procedure Register is
         use AWS.Services;
      begin
         --  Register blocks
         @@TABLE@@
         @@IF@@ @_UPPER:SLICE(1..5):VARIABLE_LIST_@ = "LAZY_"
         Web_Block.Registry.Register
           ("@_VARIABLE_LIST_@",
            "@_LOWER:REPLACE_ALL(LAZY_/):VARIABLE_LIST_@.thtml",
            Web_Callbacks.@_CAPITALIZE:REPLACE_ALL(LAZY_/):VARIABLE_LIST_@'Access);
         @@END_IF@@
         @@END_TABLE@@

         --  Register Ajax
         @@TABLE@@
         @@TABLE@@
           @@IF@@ not @_IS_EMPTY:AJAX_EVENT_@
         Services.Web_Block.Registry.Register
           ("/@_AJAX_EVENT_@$@_AJAX_ACTION_@",
            @_PACKAGE_@.R_@_CAPITALIZE:REPLACE_ALL(\\./_):AJAX_FILE_@.Template,
            Web_Callbacks.@_CAPITALIZE:AJAX_EVENT_@@_UNDERSCORE_@@_CAPITALIZE:AJAX_
↪ACTION_@'Access,
            Content_Type     => MIME.Text_XML,
            Context_Required => True);
           @@END_IF@@
         @@END_TABLE@@
         @@END_TABLE@@
      end Register;
   end Lazy;
end @_PACKAGE_@;
```

Basically this is to write a register call for every template's tag starting with *LAZY_*. The second section is to write a register call for every *Ajax* event. All callbacks are expected to be in a package named *Web_Callbacks*. It is of course possible to change this template to reference callbacks for blocks and *Ajax* in separate packages. The use of a template here is very flexible.

Now let's parse the application HTML and XML templates and create the corresponding Ada specs and register calls:

```
$ templates2ada -d . -o code.ada -t templates.tada -e .thtml -e .txml
$ gnatchop code.ada
```

Look at the generated code below, it properly register the *Widget_Counter* callback to be used for rendering *LAZY_WIDGET_COUNTER* using the `widget_counter.thtml`. So we have a tight coupling between the code and the template file. If the tag is renamed in the template file the application will not compile anymore. The same is true for *Ajax* callbacks, every *Ajax* action put in a template file needs a corresponding callback in Ada. This greatly helps keeping the application code synchronized:

```
procedure Register is
   use AWS.Services;
begin
   Web_Block.Registry.Register
     ("LAZY_WIDGET_COUNTER",
      "widget_counter.thtml",
      Web_Callbacks.Widget_Counter'Access);
   Services.Web_Block.Registry.Register
     ("/onclick$next",
      WBlocks.R_Widget_Counter.Template,
      Web_Callbacks.Onclick_Next'Access,
      Content_Type     => MIME.Text_XML,
      Context_Required => True);
end Register;
```

In the main, it is just now required to register the Web pages and to call the generated *Register* procedure:

```
Services.Web_Block.Registry.Register ("/", "page.thtml", null);

WBlocks.Lazy.Register;
```

Moreover, an Ada spec containing reference for the tag names is generated for every HTML and XML template file. All tags can be referenced using those specs, it is not needed to use string literal in the application. Again, this ensures that a tag which is renamed or deleted is detected at compilation time. For example the *Widget_Counter* callback can be rewritten as follow:

```
procedure Widget_Counter
  (Request      : in              Status.Data;
   Context      : not null access Web_Block.Context.Object;
   Translations : in out          Templates.Translate_Set)
is
   N : Natural := 0;
begin
   if Context.Exist ("N") then
      N := Natural'Value (Context.Get_Value ("N"));
   end if;

   Templates.Insert
     (Translations, Templates.Assoc (WBlocks.Widget_Counter.COUNTER, N));
end Widget_Counter;
```

4.9 Web Cross-References

When building an *Ajax* Web applications it is required to give ids to web elements to be able to reference them. It is also quite common to use CSS to give such and such item a specific style. After some time it is quite difficult to keep track of all those ids. This services helps answer if they are all used, or we reference an id that does not exist anymore.

webxref has been designed to help finding such problems.

The files kinds handled are:

.css*, *.tcss
A CSS (or template CSS file). Ids and classes inside are recorded as CSS definitions.

.xml*, *.html*, *.thtml
A meta-language document. Ids and classes inside are recorded as referencing a CSS definition and meta-language definition.

.txml
An *Ajax* response file. Ids declared inside are recorded as referencing a meta-language definition.

The features are:

cross-references
By default *webxref* output all the references to ids and classes.

finding unused items
Output the ids/classes that are defined but not used. For example an id declared in a CSS but never referenced into an HTML document or an HTML id never referenced in an *Ajax* response file `.txml` document.

finding undeclared items
Output ids/classes that are referenced but never defined. This is for example an id inside an *Ajax* response file which is never defined into an HTML document.

enforcing a naming scheme for ids and classes
It can enforce a specific prefix for ids and classes. The id prefix can be based on the filename (using filename's first character and all character before an underscore). This make it less likely to find the same id on multiple files.

Note that all references are in a format recognized by tools like *GPS* and *Emacs*. It is then possible to navigate inside them easily.

All *webxref* options are listed using the *-h* option.

4.10 WebSockets

4.10.1 Introduction to WebSockets

WebSockets are part of HTML5, the API is being standardized by the W3C and the protocol by the IETF (see RFC-6455). It is a bidirectional and full-duplex communication channel between the client and the server. Most Web Browsers are now supporting (at least part) of the WebSocket recommendation. On the client side, the WebSockets are programmed in JavaScript as done for Ajax for example.

A WebSocket is always opened at the request of a client. This can be done on the same port as the main HTTP protocol. This is possible because the initial handshake to open a WebSocket is done in pure HTTP protocol. Past this initial handshake the socket is switching protocol from HTTP to the one called WebSocket protocol.

It is not necessary to know the protocol used is WebSockets, AWS comes with some high level services on the server side and also on the client side.

4.10.2 WebSockets on the client (javascript)

The WebSocket is created on the client side. As there is some differences between Web browsers, AWS provides a wrapper routine to create a WebSocket:

```
ws = AWS.WebSocket.open('ws://localhost:8080/echo');
```

This basically creates a WebSocket and contacts the local server using port 8080.

This method is declared in `aws.tjs` which must be included:

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

A WebSocket Javascript's object has four method's callbacks:

onopen
: Called when the WebSocket has been opened. This means that the initial handshake with the server has been accepted. At this point the WebSocket is ready to send and receive messages.

onmessage
: Called for every incoming message. This callback receives a single parameter which is the event. The actual message data can be found in **e.data**.

onclose
: Called when the WebSocket is closing. This means that the server has sent a close request. After this event it is not possible to send nor receive messages through this WebSocket.

onerror
: Called when an error has occurred. This can be a lost connection for example. This callback takes a single parameter which is the error message.

AWS comes with default implementations of these callbacks. With the two optional WebSocket constructor parameters it can be configured to fit most needs:

```
ws = AWS.WebSocket.open('ws://localhost:8080/echo', message_id, status_id);
```

message_id
: The id of the HTML element which will be used to display the incoming messages. This is most of the time the id of a *p* or *div* HTML element.

status_id
: The id of the HTML element which will be used to display the status and error messages. For example when a connection is closed.

When those default callbacks are not what is needed it is always possible to redefine them:

```
ws.onmessage = function (e) {
  code there
};
```

Likewise for the other events.

4.10.3 WebSockets on the client (Ada)

AWS also supports writing websocket clients directly in Ada. Here is an example:

```
type MySocket is new AWS.Net.WebSocket.Object with null record;
overriding procedure On_Message (Self : in out MySocket; Str : String);
--  You would likely also override On_Error and On_Close
```

(continues on next page)

(continued from previous page)

```
overriding procedure On_Message (Self : in out MySocket; Str : String) is
begin
   Ada.Text_IO.Put_Line ("++ Got message '" & Str & "'");
end On_Message;

declare
   Socket     : MySocket;
begin
   AWS.Net.WebSocket.Connect (Socket, "ws://localhost:8765");

   --  Send one message
   Socket.Send ("some message");

   --  Then wait for any number of messages from the server. Give up if
   --  no message is available for 2s. If messages become available, the
   --  procedure On_Message will be called.
   while Socket.Poll (Timeout => 2.0) loop
      null;
   end loop;

   Socket.Close ("");
end;
```

You are responsible for checking regularly whether any message has been received from the server.

4.10.4 WebSockets on the server

The first step is to setup the server to dispatch the incoming messages to the proper WebSocket object. For this one needs to inherit from *AWS.Net.WebSocket.Object* and redefine at least two methods *Create* and *On_Message*:

Create

This is the constructor that will be used by the server to handle some WebSockets. This constructor will be associated to some URI, see below:

```
function Create
  (Socket  : Socket_Access;
   Request : AWS.Status.Data) return Object'Class;
```

The default constructor creates a WebSocket of type *AWS.Net.WebSocket.Object*. It is not possible to receive events (close, open, error) using such object it is only possible to send messages to the clients.

Here is an example on a custom socket:

```
type MySocket is new Net.WebSocket.Object with null record;

function Create
  (Socket  : Socket_Access;
   Request : AWS.Status.Data) return AWS.Net.WebSocket.Object'Class
is
   --  Note the call to the other version of Create*
   return MySocket'
     (AWS.Net.WebSocket.Object
```

(continues on next page)

(continued from previous page)

```
      (AWS.Net.WebSocket.Create (Socket, Request)) with null record);
end Create;
```

It is also possible to deny the handshake by returning an object from AWS.Net.WebSocket.Handshake_Error.

On_Open

This is the callback that will be called when the WebSocket is opened:

```
procedure On_Open
  (Socket : in out Object; Message : String) is null;
```

On_Message

This is the callback that will be called for every message sent by the client on the corresponding WebSocket:

```
procedure On_Message
  (Socket : in out Object; Message : String);
```

The first parameter is the WebSocket itself, it is possible to send a message directly by using the associated *Send* method. Note that the default implementation supports the XML based Ajax actions. See see *XML based Ajax* and can be used to redirect simple message to an HTML widget given it's id.

On_Close

This is the callback that will be called when the WebSocket is closed:

```
procedure On_Close
  (Socket : in out Object; Message : String) is null;
```

On_Error

This is the callback that will be called when an error occurs on the WebSocket:

```
procedure On_Error
  (Socket : in out Object; Message : String) is null;
```

When this is done, the constructor declared above needs to be registered to handle some WebSocket designated by the URI. For example to have this WebSocket handling all URI named */echo*:

```
Net.WebSocket.Registry.Register ("/echo", CB.Create'Access);
```

Where *CB.Create* is the constructor redefined for the new WebSocket class.

The last step is to start the WebSocket server which are needed to handle the incoming messages:

```
Net.WebSocket.Registry.Control.Start;
```

At this point all is setup to have AWS supports WebSockets. Sending messages can be done to a single client or by broadcasting to all clients for a specific URI. To send a message one need to create a *Net.WebSocket.Registry.Recipient* object. For example to broadcast a message to all Web clients having opened the */echo* WebSocket:

```
Rcp : Net.WebSocket.Registry.Recipient :=
        Net.WebSocket.Registry.Create (URI => "/echo");

Net.WebSocket.Registry.Send (Rcp, "A simple message");
```

As we have seen before, this will send a message to clients which will in turn trigger the *onmessage* Javascript method.

It is also possible to send a message to clients from a specific origin by using the *Origin* information:

```
Rcp : Net.WebSocket.Registry.Recipient :=
        Net.WebSocket.Registry.Create (URI => "/echo"; Origin => ".*\\.fr");

Net.WebSocket.Registry.Send (Rcp, "A simple message");
```

The above recipent targets all WebSockets whose URI is *"/echo"* and that have been created from a Web page originating from a Web server running in the *.fr* domain. Note that *URI* and the *Origin* are regular expressions.

The *Origin* value can be used by a server to handle only WebSockets originating from it's own domain. Restricting the origin of the WebSockets can be done with the *WEBSOCKET_ORIGIN* config parameter, see WebSocket_Origin.

CHAPTER

FIVE

USING SOAP

SOAP can be used to implement Web Services. The *SOAP* implementation uses *AWS HTTP* as the transport layer. *SOAP* is platform and language independent, to ensure good inter-operability, *AWS/SOAP* implementation has been validated through http://validator.soapware.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 details 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 the *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 bound 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:

- 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");
```

- 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);
```

- 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);
```

- 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 *Callback procedure*. 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:

- 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.

- 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, *SOAP.Parameters*. 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");
```

- Implements the Web Service

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

- 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*.

- 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 are 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* retrieves the *SOAPAction* and the Payload from the *SOAP* request. This is transparent to the user.

- Using Utils.SOAP_Wrapper

  It is possible to dispatch to such a 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
```

(continues on next page)

(continued from previous page)

```
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
      ...
```

- Using a SOAP Dispatcher

  AWS also provides a *SOAP* specific dispatcher. This dispatcher will automatically call 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 the SOAP dispatcher the above code would be re-written as:

```
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);
```

  The dispacther 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.

CHAPTER

SIX

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 *LibAdaLang*.

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 *LibAdaLang*.

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"?>
<wsdl:definitions name="Simple"
   targetNamespace="http://soapaws/Simple_def/"
   xmlns:tns="http://soapaws/Simple_def/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
```

(continues on next page)

(continued from previous page)

```
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:n1="http://soapaws/Standard_pkg/"
   xmlns:n2="http://soapaws/Simple_pkg/">

   <!-- Generated by AWS/Ada2WSDL v1.3.1
        on Tuesday 25 November 2014 at 11:02:44 -->

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

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

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

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

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

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

The value of the *name* attribute in the *description* node 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 more complex specifications 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 parses Ada records, arrays, derived types, enumerations, procedures and functions and generates 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**.

Long_Float
: Mapped to **xsd:double**

Long_Long_Float
: Mapped to **xsd:double**, not supported by SOAP, mapped for convenience but precision cannot be guaranteed.

SOAP.Types.Decimal
: Mapped to **xsd:decimal**

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.

SOAP.Types.Normalized_String
: Mapped to **xsd:normalizedString**

SOAP.Types.Token
: Mapped to **xsd:token**

SOAP.Types.Any_URI
: Mapped to **xsd:anyURI**

Character
: Mapped to a Character schema definition:

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

Ada.Calendar.Time* and *SOAP.Types.Local_Date_Time
: Mapped to **xsd:dateTime**

SOAP.Types.Local_Date
Mapped to **xsd:date**

SOAP.Types.Local_Time
Mapped to **xsd:time**

Duration
Mapped to **xsd:duration**

SOAP.Utils.SOAP_Base64
Mapped to **xsd:base64Binary**. *SOAP.Utils.SOAP_Base64* is a subtype of string which is 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.

Derived types
Mapped to a type schema definition:

```
type Number is new Integer;
```

is defined as:

```
<xsd:simpleType name="Number" targetNamespace="http://soapaws/WSDL_C_pkg/">
  <xsd:restriction base="xsd:int"/>
</xsd:simpleType>
```

Derived types with constraints
Mapped to a type schema definition with minInclusive and maxInclusive attributes:

```
type Number is new Integer range 1 .. 9345;
```

is defined as:

```
<xsd:simpleType name="Number" targetNamespace="http://soapaws/WSDL_C_pkg/">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value=" 1"/>
    <xsd:maxInclusive value=" 9345"/>
  </xsd:restriction>
</xsd:simpleType>
```

Or for a string:

```
.. highlight:: ada

type Code is String (1 .. 10);
```

is defined as:

```
<xsd:simpleType name="Code" targetNamespace="http://soapaws/WSDL_C_pkg/">
  <xsd:restriction base="xsd:string">
    <xsd:Length value="10"/>
  </xsd:restriction>
</xsd:simpleType>
```

Ranges

Mapped to a type schema definition with minInclusive and maxInclusive attributes:

```
type Small is range 1 .. 10;
```

is defined as:

```
<xsd:simpleType name="Small" targetNamespace="http://soapaws/WSDL_C_pkg/">
  <xsd:restriction base="xsd:byte">
    <xsd:minInclusive value=" 1"/>
    <xsd:maxInclusive value=" 10"/>
  </xsd:restriction>
</xsd:simpleType>
```

Modular types

Mapped to an unsigned type with an optional maxInclusive attribute:

```
type Count is mod 14;
```

is defined as:

```
<xsd:simpleType name="Count" targetNamespace="http://soapaws/WSDL_C_pkg/">
  <xsd:restriction base="xsd:unsignedByte">
    <xsd:maxInclusive value=" 13"/>
  </xsd:restriction>
</xsd:simpleType>
```

Decimal types

Mapped to a decimal with possible range constraints:

```
type Price is delta 0.01 digits 12 range 0.0 .. 1234.1;
```

is defined as:

```
<xsd:simpleType name="Price">
   <xsd:restriction base="xsd:decimal">
      <xsd:minInclusive value=" 0.00000000"/>
      <xsd:maxInclusive value=" 1234.10000000"/>
   </xsd:restriction>
</xsd:simpleType>
```

Enumerations

Mapped to an enumeration schema definition. For example:

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

is defined as:

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

Records

Mapped to a struct schema definition. For example:

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

is defined as:

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

Arrays

Mapped to an array schema definition. For example:

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

is defined as:

```
<xsd:complexType name="Set_Of_Rec">
  <xsd:sequence>
     <xsd:element name="x" type="n1:Rec"
          minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

A SOAP encoded format can be generated with the -sea option:

```
<xsd:complexType name="Set_Of_Rec">
  <xsd:complexContent>
    <xsd:restriction base="soap-enc:Array">
      <xsd:attribute ref="soap-enc:arrayType" wsdl:arrayType="tns:Rec[]"/>
    </xsd:restriction>
  </xsd:complexContent>
</xsd: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 an Ada.Containers.Vectors or a safe access array component (legacy mode). Both support are described below.

Array inside a record (Ada.Containers.Vectors)

Using an Ada.Containers.Vectors is the preferred way of supporting array inside records.

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 corresponding Ada.Containers:

```
package Set_Of_Int_Type is
  new Ada.Containers.Vectors (Positive, Integer);
```

And then the vectors can be added into the record:

```
type Complex_Rec is record
   SI : Set_Of_Int_Type.Vectors;
end record;
```

These Ada definitions are fully recognized by `ada2wsdl` and will generate standard array and record *WSDL* definitions as seen above:

```
<xsd:complexType name="Integer_List_Type">
   <xsd:sequence>
      <xsd:element name="x" type="xsd:int"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Complex_Rec">
   <xsd:all>
      <xsd:element name="SI" type="tns:Integer_List_Type"/>
   </xsd:all>
</xsd:complexType>
```

Array inside a record (legacy)

Using a safe pointer array component to support array inside records. Such a type is built using a generic runtime support package named *SOAP.Utils.Safe_Pointers*. This package implements a reference counter for the array access and will automatically release the memory when no more reference exist to 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.Utils.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`. *Working with WSDL documents*.

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:

```
<xsd:complexType name="Set_Of_Int">
  <xsd:sequence>
     <xsd:element name="x" type="xsd:int"
          minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

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

Array as routine parameter

When an array is passed as parameter to a *SOAP* routine it is also required to create a corresponding Ada.Containers.Vectors or a Safe_Pointer when using a *Document/Literal* binding and using a user's type package (see *-types* and '*-spec wsdl2aws* options).

Array as routine parameter (Ada.Containers.Vectors)

This is needed for the *AWS* generated code to handle this routine. Even if required in a very specific case it is never an error to declare such a Ada.Containers.Vectors for an array.

For example:

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

procedure Call (Values : Set_Of_Int);
```

Then the following declaration is required:

```
package Set_Of_Int_Type is
  new Ada.Containers.Vectors (Positive, Integer);
```

Array as routine parameter (legacy)

This is needed for the *AWS* generated code to handle this routine. Even if required in a very specific case it is never an error to declare such a Safe_Pointer for an array.

For example:

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

procedure Call (Values : Set_Of_Int);
```

Then the following declarations are required:

```
type Set_Of_Int_Access is access Set_Of_Int;

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

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.

-doc

Generate document's style binding (default is RPC)

-lit

Generate literal's style binding (default is encoded)

-n name

Specify the schema name space root name. The default value is "soapaws".

-noenum

Do not generate *WSDL* representation for Ada enumerations, map them to standard string. *Ada mapping to WSDL*.

-sea

Generate SOAP encoded format for array definitions. This option is kept for compatibility reason, but the schema based definition for arrays is recommended for better interoperability.

-o file

Generate the *WSDL* document into file.

-P proj
The project file to use for building the spec.

-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.

-t path
Specify the path to the tree file directory to use. This is needed when using a project file the object directory is not the current directory.

-d
Do not generate date/time in WSDL.

-v
Verbose mode, display the parsed spec.

6.1.4 ada2wsdl limitations

- Constrained array field in records unsupported.
- Unbounded_String are supported with full interoperability only inside a record.
- Only unconstrained arrays are supported.
- Arrays with multiple dimensions are 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):

```
$ wsdl2aws -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 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
```

(continues on next page)

(continued from previous page)

```
   package LFIO is new Text_IO.Float_IO (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 that 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:

```
$ wsdl2aws -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 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 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 be 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 is 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), Schema => BNQuoteService.Schema);
   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 in the server specification.

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 it's *URL*.

wsdl2aws options are:

-q

Quiet mode (no output)

-d

Do not generate date/time in Ada comment.

-debug

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 exiting files with the same name.

-e URL

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*).

-p name
Specify a name prefix for all SOAPActions defined in the *WDSL*. This option can be used when multiple WSDL generated callback are to be used together and some of the *WSDL* may have the same name.

-doc
Handle document style binding as RPC ones. This is sometimes needed because some *WSDL* document specify a document style binding even though the service behave like 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 Id 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. See below for the description about the way it is generated.

-n name
Specify the schema name space root name. The default value is "soapaws".

-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.

-sp
Generate legacy Safe Pointers code for the support of array inside records.

-timeouts [timeouts | connect_timeout,send_timeout,receive_timeout]
Set the timeouts for the SOAP connection. The timeouts is either a single value used for the connect, send and receive timeouts or three values separated by a colon to set each timeout independently.

6.2.4 wsdl2aws code generator

The *wsdl2aws* tool reads a *WSDL* document and generates - based on different templates files - a set of packages. The templates are rendered with the Templates_Parser engine.

All the templates can be found in AWS installation under *share/examples/aws/wsdl2aws-templates*. They can be copied into the directory where *wsdl2aws* is started or pointed to by the environment variable *AWS_TEMPLATE_FILES*. One can then change the generated code by editing those templates.

The generated packages and the corresponding templates are described below:

<root>
Template:

```
s-root.tads
```

This is the main package, it eventually contains the full *WSDL* in comments and the description of the services as read from the *WSDL* document.

<NS>.<type>_type_pkg
Templates:

```
s-name-space-pkg.tads

s-type-record.tads     s-type-record.tadb
s-type-enum.tads       s-type-enum.tadb
s-type-derived.tads
s-type-array.tads
```

Contains all the type definitions for non standard Ada types. In these packages we find for example the definition of the records and the operations to convert them to/from SOAP objects. The types defined here have possible constraints like range attributes and/or Dynamic_Predicate aspects for Pattern and/or Length WSDL attributes.

The root package <NS> is the name-space of the actual type. This ensure that no type name clash will happen. These packages are generally not directly withed.

<root>.Types
Templates:

```
s-types.tads               s-types.tadb
s-type-record-types.tads
s-type-enum-types.tads
s-type-derived-types.tads
s-type-array-types.tads
s-stub-types.tads
```

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 aliases or renamings

of types and/or routines generated in other packages rooted with a name-space as described above. This package is the one that user's should import to gain visibility to the type definitions.

This package also contains the schema object which must be used when calling a Web service or parsing a payload.

<root>.Client
Templates:

```
s-stub.tads   s-stub.tadb
```

All specifications to call Web Services.

<root>.Server
Templates:

```
s-skel.tads   s-skel.tadb
```

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

<root>.CB
Templates:

```
s-skel-cb.tads   s-skel-cb.tadb
```

The *SOAP* dispatcher callback routine.

<main>
Template:

```
s-main.tadb
```

The template used to generate the main procedure (see option -main). 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
   ...
```

6.2.5 wsdl2aws limitations

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

- Some *SOAP* base types are not currently

supported: *xsd:hexBinary*, but all could be added in the future.

- Multi-dimensional arrays are not supported.
- Abstract types are not supported.
- SOAP MIME attachments are not supported.
- The Document/Encoded SOAP messages' style is not supported.
- complexType with xs:choice are only supported with a single occurence of each choice.

6.2.6 awsascb

The awsascb (AWS Aggregate Server Callback) tool can be used to aggregate multiple SOAP callback together. That is, after generating multiple SOAP callbacks with wsdl2aws it may be necessary to create a single server handling all the services. This tools is designed for this.

```
Usage: awsascb <root1> <root2>
```

There are no option for this tool. The *root* parameters are the *wsdl2aws* generated root service name units. This tool generates a unit named *agg_server_cb* which contains a SOAP callback and a dispatcher to be used by the server's main subprogram. Here is the specification:

```
--  DO NOT EDIT : generated by awsasc

with AWS.Response;
with AWS.Status;

with SOAP.Dispatchers.Callback;
with SOAP.Message.Payload;
with SOAP.WSDL.Schema;

package Agg_Server_CB is

   use AWS;
   use SOAP;

   pragma Style_Checks (Off);

   type Handler is new SOAP.Dispatchers.Callback.Handler with null record;

   overriding function Schema
     (Dispatcher : Handler;
      SOAPAction : String)
      return WSDL.Schema.Definition;

   function Create
     (HTTP_Callback : AWS.Response.Callback) return Handler;
   --  Returns an handler whose SOAP_Callback is the one below
```

(continues on next page)

(continued from previous page)

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

end Agg_Server_CB;
```

And following is an example on using such generated aggregate server callback from a server's main:

```
   WS   : Server.HTTP;
   Conf : Config.Object;
   Disp : Agg_Server_CB.Handler;

begin
   Config.Set.Server_Port (Conf, 0);

   Disp := Agg_Server_CB.Create (HTTP_CB'Access);

   AWS.Server.Start (WS, Disp, Conf);
```

6.3 Using ada2wsdl and wsdl2aws together

Using both tools together is an effective way to rapidly build 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 : Float;
   end record;

   function Distance (P1, P2 : in Point) return 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 -largs ...
```

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 for user's specification (reference to *Graphics.Point* for example).

CHAPTER

SEVEN

WORKING WITH MAILS

7.1 Sending e-mail

AWS provides a complete API for sending e-mail using the *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 (non secure)

```
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.

- Initialize the SMTP server secure and with authentication

```
Auth      : aliased constant SMTP.Authentication.Plain.Credential :=
              SMTP.Authentication.Plain.Initialize
                ("SMTP_Login", "SMTP_Password");
```

```
SMTP_Server : SMTP.Receiver :=
                SMTP.Client.Initialize
                  (Server_Name => "smtp.hostname",
                   Port        => 587,
                   Security    => SMTP.STARTTLS,
                   Credential  => Auth'Unchecked_Access);
```

- Send the e-mail

  To send an e-mail there are many different APIs. Let's send a simple text mail:

```
Status : SMTP.Status;

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

  Here Status will contain the SMTP returned status.

- Check that everything is ok

  Using the Status data it is possible to check whether the message was or wasn't sent by the server. The status contains a code and an error message, both of them can be retrieved using specific routines, see *AWS.SMTP*. It is also possible to check that the call was successful with the *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 also provides a way to send messages to multiple recipients at the same time and to send messages with alternative content (text and HTML for example). These features are not described here, complete documentation for the specification can be found at *AWS.SMTP* and *AWS.SMTP.Client*.

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 *AWS.POP*.

- 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 typically not supported by *ISP* at this time (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);
```

- Retreiving individual e-mails

  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 retreive 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);
```

There exist 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 and the content as a memory stream. See *AWS.POP*.

- Closing the connection

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

CHAPTER

EIGHT

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
We open a connection:

```
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 *AWS.LDAP.Client*.

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 *AWS.LDAP.Client* 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 follows the LDAP API closely.

CHAPTER

NINE

JABBER

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

- Presence

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

- Message

  To send messages to a specific JID (Jabber ID)

Note that if you want an application to check the presence or send messages 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 has 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 ?");
```

CHAPTER

TEN

RESOURCES

AWS supports 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 embbed 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 contains 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 *awsres tool* 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 *AWS.Resources.Streams*.

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 [-hopqrRuz] file1/dir1 [-uz] [file2/dir2...]
```

-a
: packages are named after the actual filenames

-h
: Display help message.

-o
: Specify the output directory, by default it is the current directory.

-p name
: Append the specified prefix to the resource names.

-q
: Quiet mode.

-R
: Activate recursive behavior. In this mode `awsres` will parse recursively all subdirectories. If a directory is specified on the command line then all files in this directory and sub-directories will be added. If a file (possibly a pattern) is specificed on the command line then only files matching in directroy and sub-directories will be added.

-r name
: Set the root unit name. Default is *res*.

-u
: Add following files as uncompressed resources.

-z
: Add following files as compressed resources.

CHAPTER

ELEVEN

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:

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.

ACCEPT_QUEUE_SIZE
: see *Configuration options*.

ACCEPTOR_LENGTH
: Number of sockets in the internal socket set.

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.

ADMIN
: URI to the administrative page.

CASE_SENSITIVE_PARAMETERS
: see *Configuration options*.

CHECK_URL_VALIDITY
: see *Configuration options*.

CLEANER_CLIENT_DATA_TIMEOUT
: see *Configuration options*.

CLEANER_CLIENT_HEADER_TIMEOUT
: see *Configuration options*.

CLEANER_SERVER_RESPONSE_TIMEOUT
: see *Configuration options*.

CLEANER_WAIT_FOR_CLIENT_TIMEOUT
: see *Configuration options*.

CURRENT_CONNECTIONS
Number of current connections to the server.

ERROR_LOG (boolean tag)
This is set to true if error logging is active.

ERROR_LOG_FILE
The error log file full pathname.

ERROR_LOG_FILENAME_PREFIX
see *Configuration options*.

ERROR_LOG_SPLIT_MODE
see *Configuration options*.

FORCE_CLIENT_DATA_TIMEOUT
see *Configuration options*.

FORCE_CLIENT_HEADER_TIMEOUT
see *Configuration options*.

FORCE_SERVER_RESPONSE_TIMEOUT
see *Configuration options*.

FORCE_WAIT_FOR_CLIENT_TIMEOUT
see *Configuration options*.

FREE_SLOTS_KEEP_ALIVE_LIMIT
see *Configuration options*.

LINE_STACK_SIZE
see *Configuration options*.

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.

LOG (boolean tag)
This is set to true if logging is active.

LOG_FILE
The log file full pathname.

LOG_FILENAME_PREFIX
see *Configuration options*.

LOG_FILE_DIRECTORY
see *Configuration options*.

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

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

MAX_CONCURRENT_DOWNLOAD
see *Configuration options*.

MAX_CONNECTION
see *Configuration options*.

PEER_NAME_V (vector tag)
A list of peer name. One for each connection. This is the name of the last peer connected to the slot.

PHASE_V (vector tag)
What is the slot currently doing, for example Server_Processing or Closed.

RECEIVE_TIMEOUT
see *Configuration options*.

REUSE_ADDRESS
see *Configuration options*.

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

SECURITY_MODE
see *Configuration options*.

CIPHER_PRIORITIES
see *Configuration options*.

SEND_TIMEOUT
see *Configuration options*.

SERVER_HOST
see *Configuration options*.

SERVER_NAME
see *Configuration options*.

SERVER_PORT
see *Configuration options*.

SERVER_SOCK
Server socket ID.

SESSION
see *Configuration options*.

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.

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

SESSION_NAME
see *Configuration options*.

SESSIONS_TERMINATE_V (vector tag)
A list of time. Each item correspond to the time when the session will be obsoleted.

SESSIONS_TS_V (vector tag)
A list of time stamp. Each item correspond to a session last access time.

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 *Configuration options*.

START_TIME
A timestamp in YYYY-MM-DD HH:MM:SS format. When the server was started.

TRANSIENT_CLEANUP_INTERVAL
see *Configuration options*.

TRANSIENT_LIFETIME
see *Configuration options*.

UPLOAD_DIRECTORY
see *Configuration options*.

UPLOAD_SIZE_LIMIT
see *Configuration options*.

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.

WWW_ROOT
see *Configuration options*.

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*.

CHAPTER

TWELVE

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                                          E. Nebel
Request For Comments: 1867                                   L. Masinter
Category: Experimental                                   Xerox Corporation
                                                             November 1995

                     Form-based File Upload in HTML
```

RFC 1939

```
Network Working Group                                            J. Myers
Request for Comments: 1939                                 Carnegie Mellon
STD: 53                                                           M. Rose
Obsoletes: 1725                                 Dover Beach Consulting, Inc.
Category: Standards Track                                        May 1996

                    Post Office Protocol - Version 3
```

RFC 1945

```
Network Working Group                                       T. Berners-Lee
Request for Comments: 1945                                         MIT/LCS
Category: Informational                                       R. Fielding
                                                                UC Irvine
                                                             H. Frystyk
                                                                   MIT/LCS
                                                                  May 1996
```

(continues on next page)

(continued from previous page)

```
                Hypertext Transfer Protocol -- HTTP/1.0
```

RFC 2049

```
Network Working Group                                          N. Freed
Request for Comments: 2049                                     Innosoft
Obsoletes: 1521, 1522, 1590                                N. Borenstein
Category: Standards Track                                  First Virtual
                                                           November 1996

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

RFC 2109

```
Network Working Group                                        D. Kristol
Request for Comments: 2109       Bell Laboratories, Lucent Technologies
Category: Standards Track                                    L. Montulli
                                              Netscape Communications
                                                        February 1997

                     HTTP State Management Mechanism
```

RFC 2195

```
Network Working Group                                        J. Klensin
Request for Comments: 2195                                     R. Catoe
Category: Standards Track                                  P. Krumviede
Obsoletes: 2095                                                    MCI
                                                         September 1997

       IMAP/POP AUTHorize Extension for Simple Challenge/Response
```

RFC 2554

```
Network Working Group                                          J. Myers
Request for Comments: 2554                       Netscape Communications
Category: Standards Track                                   March 1999

                          SMTP Service Extension
                            for Authentication
```

RFC 2616

```
Network Working Group                                        R. Fielding
Request for Comments: 2616                                     UC Irvine
Obsoletes: 2068                                                J. Gettys
Category: Standards Track                                     Compaq/W3C
                                                             J. Mogul
                                                                Compaq
                                                            H. Frystyk
```

(continues on next page)

(continued from previous page)

```
                                                          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                                       J. Franks
Request for Comments: 2617                      Northwestern University
Obsoletes: 2069                                        P. Hallam-Baker
Category: Standards Track                               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
```

draft 302

```
Transport Layer Security Working Group                  Alan O. Freier
INTERNET-DRAFT                                 Netscape Communications
Expire in six months                                    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
```

(continues on next page)

(continued from previous page)

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

CHAPTER

THIRTEEN

AWS API REFERENCE

13.1 AWS

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

package AWS with Pure is

   Version      : constant String := "25.1.0";

   HTTP_10      : constant String := "HTTP/1.0";
   HTTP_11      : constant String := "HTTP/1.1";
   HTTP_2       : constant String := "HTTP/2";
```

(continues on next page)

(continued from previous page)

```
   HTTP_Version : String renames HTTP_11;

   type HTTP_Protocol is (HTTPv1, HTTPv2);

   CRLF : constant String := String'(1 => ASCII.CR, 2 => ASCII.LF);

end AWS;
```

13.2 AWS.Attachments

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with Ada.Strings.Unbounded;

with AWS.Headers;
with AWS.MIME;
with AWS.Net;

private with Ada.Containers.Vectors;

package AWS.Attachments is

   use Ada.Strings.Unbounded;

   type Element is private;
   type List is tagged private;

   Empty_List : constant List;

   type Content is private;

   type Encoding is (None, Base64);
```

(continues on next page)

(continued from previous page)

```
   function File
     (Filename     : String;
      Encode       : Encoding := None;
      Content_Id   : String := "";
      Content_Type : String := MIME.Text_Plain) return Content;
   --  A filename as content, if Encode is set to Base64 the file content will
   --  be base64 encoded.

   function Value
     (Data         : Unbounded_String;
      Name         : String := "";
      Encode       : Encoding := None;
      Content_Id   : String := "";
      Content_Type : String := MIME.Text_Plain) return Content;
   --  An unbounded string as content

   function Value
     (Data         : String;
      Name         : String := "";
      Encode       : Encoding := None;
      Content_Id   : String := "";
      Content_Type : String := MIME.Text_Plain) return Content
   is (Value (To_Unbounded_String (Data), Name, Encode, Content_Id,
              Content_Type));
   --  A string as content

   type Attachment_Kind is (Data, Alternative);
   --  Data        : for a standard MIME attachment
   --  Alternative : for a set of alternative content

   procedure Add
     (Attachments : in out List;
      Filename    : String;
      Content_Id  : String;
      Headers     : AWS.Headers.List := AWS.Headers.Empty_List;
      Name        : String := "";
      Encode      : Encoding := None)
   with Post => Count (Attachments) = Count (Attachments'Old) + 1;
   --  Adds an Attachment to the list.
   --  Note that the encoding will overwrite the corresponding entry in
   --  headers.

   procedure Add
     (Attachments : in out List;
      Filename    : String;
      Headers     : AWS.Headers.List;
      Name        : String := "";
      Encode      : Encoding := None)
   with Post => Count (Attachments) = Count (Attachments'Old) + 1;
   --  Adds an Attachment to the list.
   --  Note that the encoding will overwrite the corresponding entry in
```

(continues on next page)

(continued from previous page)

```
   --  headers.

   procedure Add
     (Attachments : in out List;
      Name        : String;
      Data        : Content;
      Headers     : AWS.Headers.List := AWS.Headers.Empty_List)
   with Post => Count (Attachments) = Count (Attachments'Old) + 1;
   --  Adds an Attachment to the list.
   --  Note that the encoding and content type attached to Data will
   --  overwrite the corresponding entry in headers.

   --  Alternatives content

   type Alternatives is private;

   procedure Add
     (Parts : in out Alternatives;
      Data  : Content);
   --  Add an alternative content

   procedure Add
     (Attachments : in out List;
      Parts       : Alternatives);
   --  Add an alternative group to the current attachment list

   procedure Reset
     (Attachments  : in out List;
      Delete_Files : Boolean)
   with Post => Count (Attachments) = 0;
   --  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 : List) return Natural with Inline;
   --  Returns the number of Attachments in the data

   function Get
     (Attachments : List;
      Index       : Positive) return Element
   with Pre => Index <= Count (Attachments);
   --  Returns specified Attachment

   function Get
     (Attachments : List;
      Content_Id  : String) return Element
   with
     Pre =>
       (for some K in 1 .. Count (Attachments)
        => AWS.Attachments.Content_Id (Get (Attachments, K)) = Content_Id);
   --  Returns the Attachment with the Content Id

   generic
```

(continues on next page)

(continued from previous page)

```
      with procedure Data (Chunk : String);
   procedure Get_Content
     (Attachments : List;
      Boundary    : String);
   --  Create the content to be sent for all attachments, call Data for each
   --  pieve of data.

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

   procedure Iterate
     (Attachments : List;
      Process     : not null access procedure (Attachment : Element));
   --  Calls Process for every Attachment in Message

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

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

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

   function Local_Filename (Attachment : 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 Filename (Attachment : Element) return String;
   --  Original filename on the server side. This is generally encoded on the
   --  content-type or content-disposition header.

   function Kind (Attachment : Element) return Attachment_Kind with Inline;
   --  Returns the kind of the given attachment

   function Length
     (Attachments : List;
      Boundary    : String) return Positive
   with Post => Length'Result > 8;
   --  Returns the complete size of all attachments including the surrounding
   --  boundaries.

   generic
      with procedure Data (Value : String);
   procedure Get_MIME_Header
```

(continues on next page)

(continued from previous page)

```
     (Attachments : List;
      Boundary    : out Unbounded_String;
      Alternative : Boolean := False);
   --  Output MIME header, returns the boundary for the content

   procedure Send_MIME_Header
     (Socket      : Net.Socket_Type'Class;
      Attachments : List;
      Boundary    : out Unbounded_String;
      Alternative : Boolean := False);
   --  Output MIME header, returns the boundary for the content

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

   type Root_MIME_Kind is (Multipart_Mixed, Multipart_Alternative);

   function Root_MIME (Attachments : List) return Root_MIME_Kind;
   --  Returns the root MIME kind for the given attachment list

private
   -- implementation removed
end AWS.Attachments;
```

13.3 AWS.Client

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

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

with AWS.Attachments;
with AWS.Default;
with AWS.Headers;
with AWS.HTTP2;
with AWS.Net.SSL.Certificate;
with AWS.Response;

private with Ada.Exceptions;
private with Ada.Finalization;
private with Ada.Real_Time;
private with ZLib;

private with AWS.Config;
private with AWS.HTTP2.Connection;
private with AWS.HTTP2.HPACK.Table;
private with AWS.HTTP2.Frame.Settings;
private with AWS.URL;
```

(continues on next page)

(continued from previous page)

```
private with AWS.Utils;

package AWS.Client is

   use Ada.Streams;
   use Ada.Strings.Unbounded;

   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.

   HTTP_Default : HTTP_Protocol renames HTTPv1;

   --------------
   -- Timeouts --
   --------------

   type Timeouts_Values is private;
   --  Defined the duration for the connect, send, receive and complete
   --  response receive timeouts.

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

   function Timeouts
     (Connect  : Duration := Net.Forever;
      Send     : Duration := Net.Forever;
      Receive  : Duration := Net.Forever;
      Response : Duration := Net.Forever) return Timeouts_Values;
   --  Constructor for the timeouts values

   function Timeouts (Each : Duration) return Timeouts_Values;
   --  Constructor for the timeouts values, sets all timeouts values (see
   --  Contructor above) to Each.

   function Connect_Timeout (T : Timeouts_Values) return Duration with Inline;
   --  Returns the corresponding timeout value

   function Send_Timeout (T : Timeouts_Values) return Duration with Inline;
   --  Returns the corresponding timeout value

   function Receive_Timeout (T : Timeouts_Values) return Duration with Inline;
```

(continues on next page)

(continued from previous page)

```
   --  Returns the corresponding timeout value

   function Response_Timeout (T : Timeouts_Values) return Duration with Inline;
   --  Returns the corresponding timeout value

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

   type Content_Bound is new
     Stream_Element_Offset range -1 .. Stream_Element_Offset'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;

   subtype Header_List is Headers.List;
   Empty_Header_List : constant Header_List := Headers.Empty_List;

   subtype Attachment_List is Attachments.List;
   Empty_Attachment_List : constant Attachment_List := Attachments.Empty_List;

   function Get
     (URL                : String;
      User               : String          := No_Data;
      Pwd                : String          := No_Data;
      Proxy              : String          := No_Data;
      Proxy_User         : String          := No_Data;
      Proxy_Pwd          : String          := No_Data;
      Timeouts           : Timeouts_Values := No_Timeout;
      Data_Range         : Content_Range   := No_Range;
      Follow_Redirection : Boolean         := False;
      Headers            : Header_List     := Empty_Header_List;
      User_Agent         : String          := Default.User_Agent;
      HTTP_Version       : HTTP_Protocol   := HTTP_Default)
      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.
```

(continues on next page)

(continued from previous page)

```
   --  If User/Pwd are given then it uses it to access the URL.
   --
   --  Optionally it connects 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          : String;
      User         : String          := No_Data;
      Pwd          : String          := No_Data;
      Proxy        : String          := No_Data;
      Proxy_User   : String          := No_Data;
      Proxy_Pwd    : String          := No_Data;
      Timeouts     : Timeouts_Values := No_Timeout;
      Headers      : Header_List     := Empty_Header_List;
      User_Agent   : String          := Default.User_Agent;
      HTTP_Version : HTTP_Protocol   := HTTP_Default) 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          : String;
      Data         : String;
      User         : String          := No_Data;
      Pwd          : String          := No_Data;
      Proxy        : String          := No_Data;
      Proxy_User   : String          := No_Data;
      Proxy_Pwd    : String          := No_Data;
      Timeouts     : Timeouts_Values := No_Timeout;
      Headers      : Header_List     := Empty_Header_List;
      User_Agent   : String          := Default.User_Agent;
      HTTP_Version : HTTP_Protocol   := HTTP_Default) return Response.Data;
   --  Send to the server URL a PUT request with Data
   --  Put will retry one time if it fails.

   function Delete
     (URL          : String;
      Data         : String;
      User         : String          := No_Data;
      Pwd          : String          := No_Data;
      Proxy        : String          := No_Data;
      Proxy_User   : String          := No_Data;
      Proxy_Pwd    : String          := No_Data;
```

(continues on next page)

(continued from previous page)

```
      Timeouts     : Timeouts_Values := No_Timeout;
      Headers      : Header_List     := Empty_Header_List;
      User_Agent   : String          := Default.User_Agent;
      HTTP_Version : HTTP_Protocol   := HTTP_Default) return Response.Data;
   --  Send to the server URL a DELETE request with Data
   --  Delete will retry one time if it fails.

   function Delete
     (URL          : String;
      Data         : Stream_Element_Array;
      User         : String          := No_Data;
      Pwd          : String          := No_Data;
      Proxy        : String          := No_Data;
      Proxy_User   : String          := No_Data;
      Proxy_Pwd    : String          := No_Data;
      Timeouts     : Timeouts_Values := No_Timeout;
      Headers      : Header_List     := Empty_Header_List;
      User_Agent   : String          := Default.User_Agent;
      HTTP_Version : HTTP_Protocol   := HTTP_Default) return Response.Data;
   --  Send to the server URL a DELETE request with Data
   --  Delete will retry one time if it fails.

   function Post
     (URL          : String;
      Data         : String;
      Content_Type : String          := No_Data;
      User         : String          := No_Data;
      Pwd          : String          := No_Data;
      Proxy        : String          := No_Data;
      Proxy_User   : String          := No_Data;
      Proxy_Pwd    : String          := No_Data;
      Timeouts     : Timeouts_Values := No_Timeout;
      Attachments  : Attachment_List := Empty_Attachment_List;
      Headers      : Header_List     := Empty_Header_List;
      User_Agent   : String          := Default.User_Agent;
      HTTP_Version : HTTP_Protocol   := HTTP_Default)
      return Response.Data;
   --  Send to the server URL a POST request with Data
   --  Post will retry one time if it fails.

   function Post
     (URL          : String;
      Data         : Stream_Element_Array;
      Content_Type : String          := No_Data;
      User         : String          := No_Data;
      Pwd          : String          := No_Data;
      Proxy        : String          := No_Data;
      Proxy_User   : String          := No_Data;
      Proxy_Pwd    : String          := No_Data;
      Timeouts     : Timeouts_Values := No_Timeout;
      Attachments  : Attachment_List := Empty_Attachment_List;
      Headers      : Header_List     := Empty_Header_List;
```

(continues on next page)

(continued from previous page)

```
   User_Agent   : String          := Default.User_Agent;
   HTTP_Version : HTTP_Protocol   := HTTP_Default)
   return Response.Data;
--  Idem as above but with binary data

function SOAP_Post
  (URL          : String;
   Data         : String;
   SOAPAction   : String;
   User         : String          := No_Data;
   Pwd          : String          := No_Data;
   Proxy        : String          := No_Data;
   Proxy_User   : String          := No_Data;
   Proxy_Pwd    : String          := No_Data;
   Timeouts     : Timeouts_Values := No_Timeout;
   Attachments  : Attachment_List := Empty_Attachment_List;
   Headers      : Header_List     := Empty_Header_List;
   User_Agent   : String          := Default.User_Agent;
   HTTP_Version : HTTP_Protocol   := HTTP_Default)
   return Response.Data;
--  Send to the server URL a POST request with Data
--  Post will retry one time if it fails.

function Upload
  (URL          : String;
   Filename     : String;
   User         : String          := No_Data;
   Pwd          : String          := No_Data;
   Proxy        : String          := No_Data;
   Proxy_User   : String          := No_Data;
   Proxy_Pwd    : String          := No_Data;
   Timeouts     : Timeouts_Values := No_Timeout;
   Headers      : Header_List     := Empty_Header_List;
   Progress     : access procedure
                    (Total, Sent : Stream_Element_Offset) := null;
   User_Agent   : String          := Default.User_Agent;
   HTTP_Version : HTTP_Protocol   := HTTP_Default)
   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;
type HTTP_Connection_Access is access all HTTP_Connection;

function Create
  (Host         : String;
   User         : String          := No_Data;
   Pwd          : String          := No_Data;
```

(continues on next page)

(continued from previous page)

```
      Proxy        : String          := No_Data;
      Proxy_User   : String          := No_Data;
      Proxy_Pwd    : String          := No_Data;
      Retry        : Natural         := Retry_Default;
      Persistent   : Boolean         := True;
      Timeouts     : Timeouts_Values := No_Timeout;
      Server_Push  : Boolean         := False;
      User_Agent   : String          := Default.User_Agent;
      HTTP_Version : HTTP_Protocol   := HTTP_Default)
      return HTTP_Connection;

   procedure Create
     (Connection   : in out HTTP_Connection;
      Host         : String;
      User         : String          := No_Data;
      Pwd          : String          := No_Data;
      Proxy        : String          := No_Data;
      Proxy_User   : String          := No_Data;
      Proxy_Pwd    : String          := No_Data;
      Retry        : Natural         := Retry_Default;
      Persistent   : Boolean         := True;
      Timeouts     : Timeouts_Values := No_Timeout;
      Server_Push  : Boolean         := False;
      SSL_Config   : Net.SSL.Config  := Net.SSL.Null_Config;
      User_Agent   : String          := Default.User_Agent;
      HTTP_Version : HTTP_Protocol   := HTTP_Default);
   --  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.
   --  SSL_Config is to define secure connection configuration. Othewhise
   --  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 HTTP_Version  (Connection : HTTP_Connection) return HTTP_Protocol;
   --  Returns connection HTTP version

   function Get_Certificate
     (Connection : HTTP_Connection) return Net.SSL.Certificate.Object;
   --  Return the certificate used for the secure connection. If this is not a
   --  secure connection, returns Net.SSL.Certificate.Undefined.

   function Host (Connection : HTTP_Connection) return String;
   --  Returns the host as recorded into the connection

   procedure Set_Headers
```

(continues on next page)

(continued from previous page)

```
   (Connection : in out HTTP_Connection; Headers : Header_List) with Inline;
--  Set additional headers for connection

procedure Set_WWW_Authentication
   (Connection : in out HTTP_Connection;
    User       : String;
    Pwd        : String;
    Mode       : 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       : String;
    Pwd        : String;
    Mode       : Authentication_Mode);
--  Sets the username, password and authentication mode for the proxy
--  authentication.

procedure Set_Persistent
   (Connection : in out HTTP_Connection; Value : Boolean) with Inline;
--  Change Persistent flag of the connection. If persistent is True the
--  connection will remain open, otherwise it will be closed after each
--  request, next request and further would be with "Connection: Close"
--  header line for HTTP/1.x protocol.

procedure Set_Retry
   (Connection : in out HTTP_Connection; Value : Natural) with Inline;
--  Set the number of attempts to get responce from server

procedure Clear_SSL_Session (Connection : in out HTTP_Connection);
--  Avoid reuse SSL session data after reconnect

procedure Copy_Cookie
   (Source      : 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 Get_Cookie (Connection : HTTP_Connection) return String
```

(continues on next page)

(continued from previous page)

```
   with Inline;
--  Get the connection cookie

procedure Set_Cookie
  (Connection : in out HTTP_Connection; Cookie : String) with Inline;
--  Set the connection cookie

function Cipher_Description (Connection : HTTP_Connection) return String;

function SSL_Session_Id (Connection : HTTP_Connection) return String;
--  Returns base64 encoded SSL session identifier.
--  Returns empty string for plain HTTP connections and for not connected
--  SSL HTTP connections.

function Read_Until
  (Connection : HTTP_Connection;
   Delimiter  : String;
   Wait       : Boolean := True) 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 Wait is False the routine is looking for delimiter only
--  in the internal socket buffer and return empty string if no delimiter
--  found. If Wait is True and returned data is empty or does not termintate
--  with the delimiter the server push connection is closed.

procedure Read_Until
  (Connection : in out HTTP_Connection;
   Delimiter  : String;
   Result     : in out Unbounded_String;
   Wait       : Boolean := True);
--  Idem as above but returns the result as an Unbounded_String

procedure Read_Some
  (Connection : in out HTTP_Connection;
   Data       : out Stream_Element_Array;
   Last       : out 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 Stream_Element_Array;
   Last       : out Stream_Element_Offset);
--  Reads data from the client's connection until Data buffer if 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
```

(continues on next page)

(continued from previous page)

```
   (Connection : in out HTTP_Connection;
    Result     : out Response.Data;
    URI        : String          := No_Data;
    Data_Range : Content_Range   := No_Range;
    Headers    : Header_List     := Empty_Header_List);
--  Same as Get above but using a Connection

procedure Head
   (Connection : in out HTTP_Connection;
    Result     : out Response.Data;
    URI        : String          := No_Data;
    Headers    : Header_List     := Empty_Header_List);
--  Same as Head above but using a Connection

procedure Delete
   (Connection : in out HTTP_Connection;
    Result     : out Response.Data;
    Data       : String;
    URI        : String          := No_Data;
    Headers    : Header_List     := Empty_Header_List);
--  Same as Delete above but using a Connection

procedure Delete
   (Connection : in out HTTP_Connection;
    Result     : out Response.Data;
    Data       : Stream_Element_Array;
    URI        : String          := No_Data;
    Headers    : Header_List     := Empty_Header_List);
--  Same as Delete above but using a Connection

procedure Put
   (Connection : in out HTTP_Connection;
    Result     : out Response.Data;
    Data       : String;
    URI        : String      := No_Data;
    Headers    : Header_List := Empty_Header_List);
--  Same as Put above but using a Connection

procedure Put
   (Connection : in out HTTP_Connection;
    Result     : out Response.Data;
    Data       : Stream_Element_Array;
    URI        : String      := No_Data;
    Headers    : Header_List := Empty_Header_List);

procedure Post
   (Connection   : in out HTTP_Connection;
    Result       : out Response.Data;
    Data         : String;
    Content_Type : String          := No_Data;
    URI          : String          := No_Data;
    Attachments  : Attachment_List := Empty_Attachment_List;
```

(continues on next page)

(continued from previous page)

```
   Headers      : Header_List     := Empty_Header_List);
--  Same as Post above but using a Connection

procedure Post
  (Connection   : in out HTTP_Connection;
   Result       : out Response.Data;
   Data         : Stream_Element_Array;
   Content_Type : String          := No_Data;
   URI          : String          := No_Data;
   Attachments  : Attachment_List := Empty_Attachment_List;
   Headers      : Header_List     := Empty_Header_List);
--  Same as Post above but using a Connection

procedure Upload
  (Connection : in out HTTP_Connection;
   Result     : out Response.Data;
   Filename   : String;
   URI        : String          := No_Data;
   Headers    : Header_List     := Empty_Header_List;
   Progress   : access procedure
     (Total, Sent : Stream_Element_Offset) := null);
--  Same as Upload above but using a Connection

procedure SOAP_Post
  (Connection  : HTTP_Connection;
   Result      : out Response.Data;
   SOAPAction  : String;
   Data        : String;
   Streaming   : Boolean         := False;
   Attachments : Attachment_List := Empty_Attachment_List;
   Headers     : Header_List     := Empty_Header_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      : Boolean)
with Inline;
--  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 : Boolean);
--  Set debug mode on/off. If debug is activated the request header and the
--  server response header will be displayed.

function Get_Socket (Connection : HTTP_Connection) return Net.Socket_Access;
```

(continues on next page)

(continued from previous page)

```
   --  Retrieve the socket used for the connection

   function Disconnect_Counter (Connection : HTTP_Connection) return Natural;
   --  Retrieve the number of disconnections during the connection life

private
   -- implementation removed
end AWS.Client;
```

13.4 AWS.Client.Hotplug

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 paswword, 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     : String;
      Password : String;
      Server   : String;
      Regexp   : String;
      URL      : 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
```

(continues on next page)

(continued from previous page)

```
     (Name     : String;
      Password : String;
      Server   : String;
      Regexp   : String) return Response.Data;
   --  Unregister hotplug module Name responding to Regexp requests from
   --  Server. See comment above about Password.

end AWS.Client.Hotplug;
```

13.5 AWS.Communication

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 : 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 emptry
   --  string value.

private
```

(continues on next page)

(continued from previous page)

```
   -- implementation removed
end AWS.Communication;
```

13.6 AWS.Communication.Client

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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     : String;
      Port       : Positive;
      Name       : String;
      Parameters : 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;
```

13.7 AWS.Communication.Server

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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     : String; -- Host name
      Name       : String; -- Message name
      Context    : not null access T;
      Parameters : 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 : Natural; Context : T_Access; Host : String := "");
```

(continues on next page)

(continued from previous page)

```
   --  Start communication HTTP server listening at the given port
   --  If Port is zero, server started at any free port and it can be taken by
   --  the Port call.

   function Port return Positive;
   --  Get the port where the server is binded

   procedure Shutdown;
   --  Shutdown the communication HTTP server

end AWS.Communication.Server;
```

13.8 AWS.Config

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2024, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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 System;

with GNAT.Regexp;

private with Ada.Strings.Unbounded;
private with AWS.Containers.String_Vectors;
private with AWS.Default;

package AWS.Config is

   type Object is tagged private;

   Default_Config : constant Object;

   --  For the external configuration to be loaded either Get_Current or
```

(continues on next page)

(continued from previous page)

```
   --  Load_Config must be called explicitely.

   function Get_Current return Object;
   --  Returns a configuration record. This is the properties as read in files
   --  'aws.ini' and 'progname.ini'. This configuration object holds only the
   --  per-server options.

   procedure Load_Config;
   --  Load configuration and store it into an internal object. This can be
   --  called when only some server-wide configuration are to be set from
   --  .ini files for example.

   ------------------------
   -- Per Server options --
   ------------------------

   ------------
   -- Server --
   ------------

   function Server_Name (O : Object) return String with Inline;
   --  This is the name of the server as set by AWS.Server.Start

   function Protocol_Family (O : Object) return String with Inline;
   --  Server protocol family. Family_Inet for IPv4, Family_Inet6 for IPv6 and
   --  Family_Unspec for unspecified protocol family.

   function IPv6_Only (O : Object) return Boolean with Inline;
   --  IPv6 server accepts only IPv6 connections

   function Server_Host (O : Object) return String with Inline;
   --  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 : Object) return Natural with Inline;
   --  This is the server port as set by the HTTP object declaration

   function HTTP2_Activated (O : Object) return Boolean with Inline;
   --  Returns True if the HTTP/2 protocol is activated

   function Hotplug_Port (O : Object) return Positive with Inline;
   --  This is the hotplug communication port needed to register and
   --  un-register an hotplug module.

   function Session (O : Object) return Boolean with Inline;
   --  Returns True if the server session is activated

   function Case_Sensitive_Parameters (O : Object) return Boolean with Inline;
   --  HTTP parameters are case sensitive

   function Session_Name (O : Object) return String with Inline;
```

(continues on next page)

(continued from previous page)

```
   --  Name of the cookie session

   function Session_Private_Name (O : Object) return String with Inline;
   --  Name of the private cookie session

   function Server_Priority (O : Object) return System.Any_Priority
     with Inline;
   --  Returns the priority used by the HTTP and WebSockets servers

   function Server_Header (O : Object) return String with Inline;
   --  Returns the Server header value

   function HTTP2_Enable_Push (O : Object) return Boolean with Inline;
   --  Whether the server has push support

   ----------------
   -- Connection --
   ----------------

   function Max_Connection (O : Object) return Positive with Inline;
   --  This is the max simultaneous connections as set by the HTTP object
   --  declaration.

   function Send_Buffer_Size (O : Object) return Natural with Inline;
   --  This is the socket buffer size used for sending data. Increasing this
   --  value will give better performances on slow or long distances
   --  connections.

   function TCP_No_Delay (O : Object) return Boolean with Inline;
   --  Returns wether the TCP_NODELAY option is set for this server

   function Free_Slots_Keep_Alive_Limit (O : Object) return Natural
     with Inline;
   --  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 : Object) return Positive with Inline;
   --  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 Keep_Alive_Close_Limit (O : Object) return Positive with Inline;
   --  This parameter defines the limit of keep alive sockets in the internal
   --  server socket set. If the number of sockets in socket set became more
   --  than Keep_Alive_Close_Limit, most close to timeout socket would be
```

(continues on next page)

(continued from previous page)

```
   --  closed. If this parameter is not defined in the configuration,
   --  the server uses Max_Connection * 4 as value.

   function Accept_Queue_Size (O : Object) return Positive with Inline;
   --  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 : Object) return Positive with Inline;
   --  HTTP lines stack size

   function Reuse_Address (O : Object) return Boolean with Inline;
   --  Returns true if bind is allowed to reuse an address (not waiting for
   --  the delay between two bind to the same port).

   function Close_On_Exec (O : Object) return Boolean with Inline;
   --  Returns true if socket descriptor are closed on child processes

   function HTTP2_Header_Table_Size (O : Object) return Positive with Inline;
   --  HTTP2 max header table size

   function HTTP2_Max_Concurrent_Streams
     (O : Object) return Positive with Inline;
   --  HTTP2 maximum number of concurrent streams

   function HTTP2_Initial_Window_Size (O : Object) return Positive with Inline;
   --  HTTP2 initial flow control window size

   function HTTP2_Max_Frame_Size (O : Object) return Positive with Inline;
   --  HTTP2 the maximum size (in bytes) of a frame

   function HTTP2_Max_Header_List_Size
     (O : Object) return Positive with Inline;
   --  HTTP2 the maximum size (in bytes) of the header list

   ----------
   -- Data --
   ----------

   function WWW_Root (O : Object) return String with Inline;
   --  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. The returned
   --  directory ends with a directory separator.

   function Upload_Directory (O : Object) return String with Inline;
   --  This point to the directory where uploaded files will be stored. The
   --  directory returned will end with a directory separator.

   function Upload_Size_Limit (O : Object) return Positive with Inline;
   --  Size limit for the client uploading data before calling the user's
```

(continues on next page)

(continued from previous page)

```
   --  callback or dispatcher handler. User can call
   --  AWS.Status.Is_Body_Uploaded to check if client data is uploaded or not
   --  because of this limit. User can still approve the uploading data above
   --  this limit by using AWS.Server.Get_Message_Body.

   function Directory_Browser_Page (O : Object) return String with Inline;
   --  Filename for the directory browser template page

   function Max_POST_Parameters (O : Object) return Positive with Inline;
   --  Returns the maximum number of POST parameters handled. Past this limit
   --  the exception Too_Many_Parameters is raised.

   ---------
   -- Log --
   ---------

   function Log_Activated (O : Object) return Boolean with Inline;
   --  Whether the default log should be activated

   function Log_File_Directory (O : Object) return String with Inline;
   --  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 : Object) return String with Inline;
   --  This is the prefix to use for the log filename

   function Log_Split_Mode (O : Object) return String with Inline;
   --  This is split mode for the log file. Possible values are : Each_Run,
   --  Daily, Monthly and None. Any other values will raise an exception.

   function Log_Size_Limit (O : Object) return Natural with Inline;

   generic
      with procedure Field_Id (Id : String);
   procedure Log_Extended_Fields_Generic_Iterate (O : Object);
   --  Calls procedure Field_Id for each extended http log field identifier

   function Log_Extended_Fields_Length (O : Object) return Natural with Inline;
   --  Returns the number of extended http log fileds identifiers.
   --  If returned value is zero then http log is not extended.

   function Error_Log_Activated (O : Object) return Boolean with Inline;
   --  Whether the error log should be activated

   function Error_Log_Filename_Prefix (O : Object) return String with Inline;
   --  This is the prefix to use for the log filename

   function Error_Log_Split_Mode (O : Object) return String with Inline;
   --  This is split mode for the log file. Possible values are : Each_Run,
   --  Daily, Monthly and None. Any other values will raise an exception.

   ------------
```

(continues on next page)

(continued from previous page)

```
   -- Status --
   ------------

   function Admin_Password (O : Object) return String with Inline;
   --  The admin password

   function Admin_Realm (O : Object) return String with Inline;
   --  The admin password

   function Admin_URI (O : Object) return String with Inline;
   --  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 : Object) return String with Inline;
   --  Filename for the status template page

   function Up_Image (O : Object) return String with Inline;
   --  Filename for the up arrow image used in the status page

   function Down_Image (O : Object) return String with Inline;
   --  Filename for the down arrow image used in the status page

   function Logo_Image (O : Object) return String with Inline;
   --  Filename for the AWS logo image used in the status page

   --------------
   -- Timeouts --
   --------------

   function Cleaner_Wait_For_Client_Timeout (O : Object) return Duration
     with Inline;
   --  Number of seconds to timout on waiting for a client request.
   --  This is a timeout for regular cleaning task.

   function Cleaner_Client_Header_Timeout (O : Object) return Duration
     with Inline;
   --  Number of seconds to timout on waiting for client header.
   --  This is a timeout for regular cleaning task.

   function Cleaner_Client_Data_Timeout (O : Object) return Duration
     with Inline;
   --  Number of seconds to timout on waiting for client message body.
   --  This is a timeout for regular cleaning task.

   function Cleaner_Server_Response_Timeout (O : Object) return Duration
     with Inline;
   --  Number of seconds to timout on waiting for client to accept answer.
   --  This is a timeout for regular cleaning task.

   function Force_Wait_For_Client_Timeout (O : Object) return Duration
     with Inline;
   --  Number of seconds to timout on waiting for a client request.
```

(continues on next page)

(continued from previous page)

```
   --  This is a timeout for urgent request when resources are missing.

   function Force_Client_Header_Timeout (O : Object) return Duration
     with Inline;
   --  Number of seconds to timout on waiting for client header.
   --  This is a timeout for urgent request when resources are missing.

   function Force_Client_Data_Timeout (O : Object) return Duration with Inline;
   --  Number of seconds to timout on waiting for client message body.
   --  This is a timeout for urgent request when resources are missing.

   function Force_Server_Response_Timeout (O : Object) return Duration
     with Inline;
   --  Number of seconds to timout on waiting for client to accept answer.
   --  This is a timeout for urgent request when resources are missing.

   function Send_Timeout (O : Object) return Duration with Inline;
   --  Number of seconds to timeout when sending chunck of data

   function Receive_Timeout (O : Object) return Duration with Inline;
   --  Number of seconds to timeout when receiving chunck of data

   --------------
   -- Security --
   --------------

   function Check_URL_Validity (O : Object) return Boolean with Inline;
   --  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 : Object) return Boolean with Inline;
   --  Is the server working through th SSL

   function Server_Certificate (O : Object) return String with Inline;
   --  Returns the certificate to be used with the secure server. Returns the
   --  empty string if the server is not a secure one.

   function Client_Certificate (O : Object) return String with Inline;
   --  Returns the certificate to be used with the secure client. Returns
   --  the empty string if the client is not a secure one or does not use
   --  a certificate.

   function Server_Key (O : Object) return String with Inline;
   --  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 : Object) return String with Inline;
   --  Returns the security mode to be used with the secure server. Returns the
   --  empty string if the server is not a secure one.

   function Cipher_Priorities (O : Object) return String with Inline;
   --  Returns the cipher priorities for the security communication
```

(continues on next page)

(continued from previous page)

```
   function TLS_Ticket_Support (O : Object) return Boolean with Inline;
   --  Is security communication side has support stateless TLS session
   --  resumption. See RFC 5077.

   function Exchange_Certificate (O : Object) return Boolean with Inline;
   --  Returns True if the client is requested to send its certificate to the
   --  server.

   function Check_Certificate (O : Object) return Boolean with Inline;
   --  Returns True if the server or client must abort the connection if the
   --  peer did not provide trusted certificate.

   function Trusted_CA (O : Object) return String with Inline;
   --  Returns the filename containing a list of trusted CA, this is to be used
   --  with the Exchange_Certificate option. The filename is on bundle of CAs
   --  that can be trusted. A client certificate signed with one of those CA
   --  will be accetped by the server.

   function CRL_File (O : Object) return String with Inline;
   --  Returns the filename containing the Certificate Revocation List. This
   --  list is used by the server to check for revoked certificate.

   function SSL_Session_Cache_Size (O : Object) return Natural with Inline;
   --  Returns SSL session cashe size

   -------------------------
   -- Per Process options --
   -------------------------

   function Session_Cleanup_Interval return Duration with Inline;
   --  Number of seconds between each run of the cleaner task to remove
   --  obsolete session data.

   function Session_Lifetime return Duration with Inline;
   --  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 Session_Id_Length return Positive with Inline;
   --  Returns the length (number of characters) of the session id

   function Session_Cleaner_Priority return System.Any_Priority with Inline;
   --  Returns the priority used by the session cleaner task

   function Service_Priority return System.Any_Priority with Inline;
   --  Returns the priority used by the others services (SMTP server, Jabber
   --  server, Push server...).

   function Config_Directory return String with Inline;
   --  Directory where AWS parameter files are located

   function Disable_Program_Ini return Boolean with Inline;
```

(continues on next page)

(continued from previous page)

```
   --  Whether the <program_name>.ini file should be read

   function Transient_Cleanup_Interval return Duration with Inline;
   --  Number of seconds between each run of the cleaner task to remove
   --  transient pages.

   function Transient_Lifetime return Duration with Inline;
   --  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 with Inline;
   --  Number of maximum concurrent download supported by the download manager
   --  service.

   function MIME_Types return String with Inline;
   --  Returns the file name of the MIME types to use

   function Input_Line_Size_Limit return Positive with Inline;
   --  Limit of the HTTP protocol text lines length

   function Context_Lifetime return Duration with Inline;
   --  Number of seconds to keep a context if not used. After this period the
   --  context data is obsoleted and will be removed during next cleanup.

   function Max_WebSocket_Handler return Positive with Inline;
   --  This is the max simultaneous connections handling WebSocket's messages

   function WebSocket_Message_Queue_Size return Positive with Inline;
   --  This is the size of the queue containing incoming messages

   function WebSocket_Send_Message_Queue_Size return Positive with Inline;
   --  This is the size of the queue containing messages to send

   function Max_WebSocket return Positive with Inline;
   --  The maximum number of simultaneous WebSocket opened. Note that that
   --  there could be more WebSocket registered when counting the closing
   --  WebSockets.

   function WebSocket_Timeout return Duration with Inline;
   --  Returns the WebSocket activity timeout. After this number of seconds
   --  without any activity the WebSocket can be closed when needed.

   function Is_WebSocket_Origin_Set return Boolean with Inline;
   --  Returns True if the Origin has been set

   function WebSocket_Origin return GNAT.Regexp.Regexp;
   --  This is regular expression to restrict WebSocket to a specific origin

   function WebSocket_Origin return String;
   --  This is the string regular expression to restrict WebSocket to a
   --  specific origin.
```

(continues on next page)

(continued from previous page)

```
   function WebSocket_Priority return System.Any_Priority;
   --  Set the priority used by the WebSocket service

   function User_Agent return String with Inline;
   --  Returns the User_Agent header value

private
   -- implementation removed
end AWS.Config;
```

13.9 AWS.Config.Ini

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 (Full_Path : Boolean) return String;
   --  Returns initialization filename for current server (using the
   --  executable name and adding .ini).

   procedure Read
     (Config   : in out Object;
      Filename : String);
   --  Read Filename and update the configuration object with the options read
   --  from it.
   --  Raises Constraint_Error in case of wrong any parameter name or value.
   --  Raises Ada.Text_IO.Status_Error if the Filename is already open.
   --  Raises Ada.Text_IO.Name_Error if Filename does not exist.
   --  Raises Ada.Text_IO.Use_Error if the external environment does not
   --  support opening for an external file with the given name.

end AWS.Config.Ini;
```

13.10 AWS.Config.Set

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2022, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 : String);
   --  This is the name of the server as set by AWS.Server.Start

   procedure Protocol_Family (O : in out Object; Value : String);
   --  Set the server protocol family. Family_Inet for IPv4, Family_Inet6 for
   --  IPv6 and Family_Unspec for unspecified protocol family.

   procedure IPv6_Only (O : in out Object; Value : Boolean);
   --  Set the mode when IPv6 server allows connect only IPv6 clients
```

(continues on next page)

(continued from previous page)

```
   procedure Server_Host (O : in out Object; Value : String);
   --  This is the server host as set by the HTTP object declaration

   procedure Server_Port (O : in out Object; Value : Natural);
   --  This is the server port as set by the HTTP object declaration

   procedure HTTP2_Activated (O : in out Object; Value : Boolean);
   --  Set to True if the HTTP/2 protocol is to be activated

   procedure Hotplug_Port (O : in out Object; Value : Positive);
   --  This is the hotplug communication port needed to register and
   --  un-register an hotplug module.

   procedure Session (O : in out Object; Value : Boolean);
   --  Enable session handling is Value is True

   procedure Case_Sensitive_Parameters (O : in out Object; Value : Boolean);
   --  Parameters are handled with the case if Value is True

   procedure Line_Stack_Size (O : in out Object; Value : Positive);
   --  HTTP lines stack size

   procedure Reuse_Address (O : in out Object; Value : Boolean);
   --  Set the reuse address policy allowing a bind without a dealy to the same
   --  address and port.

   procedure Close_On_Exec (O : in out Object; Value : Boolean);
   --  Set the close on exec policy forcing the socket descriptors to
   --  be closed in child processes.

   procedure Session_Name (O : in out Object; Value : String);
   --  Name of the cookie session

   procedure Server_Priority (O : in out Object; Value : System.Any_Priority);
   --  Set the priority used by the HTTP and WebSockets servers

   procedure Server_Header (O : in out Object; Value : String);
   --  Set the server header (value used by the Server: request header)

   procedure HTTP2_Enable_Push (O : in out Object; Value : Boolean);
   --  Whether the server has push support

   ----------------
   -- Connection --
   ----------------

   procedure Max_Connection (O : in out Object; Value : Positive);
   --  This is the max simultaneous connections as set by the HTTP object
   --  declaration.

   procedure Send_Buffer_Size (O : in out Object; Value : Positive);
   --  This is the socket buffer size used for sending data. Increasing this
```

(continues on next page)

(continued from previous page)

```
   --  value will give better performances on slow or long distances
   --  connections.

   procedure TCP_No_Delay (O : in out Object; Value : Boolean);
   --  Set the TCP_NODELAY option for this server

   procedure Free_Slots_Keep_Alive_Limit
     (O : in out Object; Value : 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 : 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 : 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.

   procedure HTTP2_Header_Table_Size (O : in out Object; Value : Positive);
   --  HTTP2 max header table size

   procedure HTTP2_Max_Concurrent_Streams
     (O : in out Object; Value : Positive);
   --  HTTP2 maximum number of concurrent streams

   procedure HTTP2_Initial_Window_Size (O : in out Object; Value : Positive);
   --  HTTP2 initial flow control window size

   procedure HTTP2_Max_Frame_Size (O : in out Object; Value : Positive);
   --  HTTP2 the maximum size (in bytes) of a frame

   procedure HTTP2_Max_Header_List_Size (O : in out Object; Value : Positive);
   --  HTTP2 the maximum size (in bytes) of the header list

   ----------
   -- Data --
   ----------

   procedure WWW_Root (O : in out Object; Value : 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.
```

(continues on next page)

(continued from previous page)

```
   procedure Upload_Directory (O : in out Object; Value : String);
   --  This point to the directory where uploaded files will be stored. The
   --  directory returned will end with a directory separator.

   procedure Upload_Size_Limit (O : in out Object; Value : Positive);
   --  Set the maximum size accepted for uploaded files

   procedure Directory_Browser_Page (O : in out Object; Value : String);
   --  Filename for the directory browser template page

   procedure Max_POST_Parameters (O : in out Object; Value : Positive);
   --  Set the maximum number of POST parameters handled. Past this limit
   --  the exception Too_Many_Parameters is raised.

   ---------
   -- Log --
   ---------

   procedure Log_Activated (O : in out Object; Value : Boolean);
   --  Whether the default log should be activated

   procedure Log_File_Directory (O : in out Object; Value : 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 : String);
   --  This is the prefix to use for the log filename

   procedure Log_Size_Limit (O : in out Object; Value : Natural);
   --  If Log_Size_Limit is more than zero and size of log file
   --  become more than Log_Size_Limit, log file is be split.

   procedure Log_Split_Mode (O : in out Object; Value : 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 : String);
   --  Comma separated list of the extended log field names. If this parameter
   --  is empty, the HTTP log would have fixed apache compartible 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:
   --
```

(continues on next page)

(continued from previous page)

```
   --  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-<appfield> Any application defined field name

   procedure Error_Log_Activated (O : in out Object; Value : Boolean);
   --  Whether the error log should be activated

   procedure Error_Log_Filename_Prefix (O : in out Object; Value : String);
   --  This is the prefix to use for the log filename

   procedure Error_Log_Split_Mode (O : in out Object; Value : 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_Password (O : in out Object; Value : String);
   --  This is the password for the admin server page as set by
   --  AWS.Server.Start. The password must be created with the aws_password
   --  tool.

   procedure Admin_URI (O : in out Object; Value : String);
   --  This is the name of the admin server page as set by AWS.Server.Start

   procedure Status_Page (O : in out Object; Value : String);
   --  Filename for the status template page

   procedure Up_Image (O : in out Object; Value : String);
   --  Filename for the up arrow image used in the status page

   procedure Down_Image (O : in out Object; Value : String);
   --  Filename for the down arrow image used in the status page

   procedure Logo_Image (O : in out Object; Value : String);
   --  Filename for the AWS logo image used in the status page
```

(continues on next page)

(continued from previous page)

```
   --------------
   -- Timeouts --
   --------------

   procedure Cleaner_Wait_For_Client_Timeout
     (O     : in out Object;
      Value : Duration);
   --  Number of seconds to timout on waiting for a client request.
   --  This is a timeout for regular cleaning task.

   procedure Cleaner_Client_Header_Timeout
     (O     : in out Object;
      Value : Duration);
   --  Number of seconds to timout on waiting for client header.
   --  This is a timeout for regular cleaning task.

   procedure Cleaner_Client_Data_Timeout
     (O     : in out Object;
      Value : Duration);
   --  Number of seconds to timout on waiting for client message body.
   --  This is a timeout for regular cleaning task.

   procedure Cleaner_Server_Response_Timeout
     (O     : in out Object;
      Value : Duration);
   --  Number of seconds to timout 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 : Duration);
   --  Number of seconds to timout 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 : Duration);
   --  Number of seconds to timout 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 : Duration);
   --  Number of seconds to timout 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 : Duration);
   --  Number of seconds to timout on waiting for client to accept answer.
   --  This is a timeout for urgent request when resources are missing.
```

(continues on next page)

(continued from previous page)

```
   procedure Send_Timeout (O  : in out Object; Value : Duration);
   --  Number of seconds to timeout when sending chunck of data

   procedure Receive_Timeout (O : in out Object; Value : Duration);
   --  Number of seconds to timeout when receiving chunck of data

   --------------
   -- Security --
   --------------

   procedure Check_URL_Validity (O : in out Object; Value : 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 : Boolean);
   --  Enable security (HTTPS/SSL) if Value is True

   procedure Server_Certificate (O : in out Object; Filename : String);
   --  Set the certificate filename in PEM format to be used with the secure
   --  server.

   procedure Client_Certificate (O : in out Object; Filename : String);
   --  Set the certificate filename in PEM format to be used with the secure
   --  client.

   procedure Server_Key (O : in out Object; Filename : String);
   --  Set the key to be used with the secure server

   procedure Security_Mode (O : in out Object; Mode : String);
   --  Set the security mode to be used with the secure server. Only values
   --  from AWS.Net.SSL.Method can be used.

   procedure Cipher_Priorities (O : in out Object; Value : String);
   --  Sets priorities for the cipher suites supported by SSL implementation.
   --  GNUTLS and OpenSSL implementations has different sintax for this
   --  parameter.

   procedure TLS_Ticket_Support (O : in out Object; Value : Boolean);
   --  Set to True for security communication side support stateless TLS
   --  session resumption. See RFC 5077.

   procedure Exchange_Certificate (O : in out Object; Value : Boolean);
   --  Set to True to request the client to send its certificate to the server

   procedure Check_Certificate (O : in out Object; Value : Boolean);
   --  Set to True if the server or client must abort the connection if the
   --  peer did not provide trusted certificate.

   procedure Trusted_CA (O : in out Object; Filename : String);
   --  Set the trusted filename containing a list of trusted CA, this
   --  is to be used with the Exchange_Certificate option. The
   --  filename is a bundle of CAs that can be trusted. A client
```

(continues on next page)

(continued from previous page)

```
   --  certificate signed with one of those CA will be accepted by the
   --  server.

   procedure CRL_File (O : in out Object; Filename : String);
   --  Returns the filename containing the Certificate Revocation List. This
   --  list is used by the server to check for revoked certificate.

   procedure SSL_Session_Cache_Size (O : in out Object; Value : Natural);

   ------------------------
   -- Per Process Options --
   ------------------------

   procedure Session_Cleanup_Interval (Value : Duration);
   --  Number of seconds between each run of the cleaner task to remove
   --  obsolete session data.

   procedure Session_Lifetime (Value : 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 Session_Id_Length (Value : Positive);
   --  Returns the length (number of characters) of the session id

   procedure Session_Cleaner_Priority (Value : System.Any_Priority);
   --  Set the priority used by the session cleaner task

   procedure Service_Priority (Value : System.Any_Priority);
   --  Set the priority used by the others services (SMTP server, Jabber
   --  server, Push server...).

   procedure Config_Directory (Value : String);
   --  Directory where AWS parameter files are located

   procedure Transient_Cleanup_Interval (Value : Duration);
   --  Number of seconds between each run of the cleaner task to remove
   --  transient pages.

   procedure Transient_Lifetime (Value : 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 Context_Lifetime (Value : Duration);
   --  Number of seconds to keep a context if not used. After this period the
   --  context data is obsoleted and will be removed during next cleanup.

   procedure Max_Concurrent_Download (Value : Positive);
   --  Control the maximum number of parallel downloads accepted by the
   --  download manager.

   procedure Max_WebSocket (Value : Positive);
   --  The maximum number of simultaneous WebSocket opened. Note that that
```

(continues on next page)

(continued from previous page)

```
   --  there could be more WebSocket registered when counting the closing
   --  WebSockets.

   procedure Max_WebSocket_Handler (Value : Positive);
   --  This is the max simultaneous connections handling WebSocket's messages

   procedure MIME_Types (Value : String);
   --  The name of the file containing the MIME types associations

   procedure WebSocket_Message_Queue_Size (Value : Positive);
   --  This is the size of the queue containing incoming messages

   procedure WebSocket_Send_Message_Queue_Size (Value : Positive);
   --  This is the size of the queue containing messages to send

   procedure WebSocket_Origin (Value : String);
   --  This is regular expression to restrict WebSocket to a specific origin

   procedure WebSocket_Priority (Value : System.Any_Priority);
   --  Set the priority used by the WebSocket service

   procedure WebSocket_Timeout (Value : Duration);
   --  Returns the WebSocket activity timeout. After this number of seconds
   --  without any activity the WebSocket can be closed when needed.

   procedure Input_Line_Size_Limit (Value : Positive);
   --  Maximum length of an HTTP parameter

   procedure User_Agent (Value : String);
   --  Set the user agent for client request heaser

   procedure Parameter
     (Config        : in out Object;
      Name          : String;
      Value         : String;
      Error_Context : 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 form.

   procedure Parameter
     (Name          : String;
      Value         : String;
      Error_Context : String := "");
   --  Set one of the AWS HTTP per process parameters. See description above

end AWS.Config.Set;
```

13.11 AWS.Containers.Tables

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2017, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with Ada.Strings.Unbounded;

private with Ada.Containers.Indefinite_Ordered_Maps;
private with Ada.Containers.Vectors;

package AWS.Containers.Tables is

   use Ada.Strings.Unbounded;

   type Table_Type is tagged private;

   Empty_Table : constant Table_Type;

   type Element is record
      Name  : Unbounded_String;
      Value : Unbounded_String;
   end record;
   --  Data type to store name/value pair retrieved from a Table_Type

   Null_Element : constant Element;
```

(continues on next page)

(continued from previous page)

```
type VString_Array is array (Positive range <>) of Unbounded_String;

function Count (Table : Table_Type) return Natural;
--  Returns the number of items in Table

function Is_Empty (Table : Table_Type) return Boolean;
--  Returns true if table is empty

function Name_Count (Table : Table_Type) return Natural;
--  Returns the number of unique key name in Table

function Case_Sensitive (Table : Table_Type) return Boolean with Inline;
--  Returns case sensitivity flag of the Table

function Names_Lowercased (Table : Table_Type) return Boolean with Inline;
--  Returns True if the names was lowercased on put into the table

function Count (Table : Table_Type; Name : String) return Natural;
--  Returns the number of value for Key Name in Table. It returns
--  0 if Key does not exist.

function Exist (Table : Table_Type; Name : String) return Boolean;
--  Returns True if Key exist in Table

function Get
  (Table : Table_Type;
   Name  : String;
   N     : Positive := 1) return String
with Post => (if N > Count (Table, Name) then Get'Result'Length = 0);
--  Returns the Nth value associated with Key into Table. Returns
--  the emptry string if key does not exist.

function Get_Name
  (Table : Table_Type; N : Positive := 1) return String
with Post => (if N > Count (Table) then Get_Name'Result'Length = 0);
--  Returns the Nth Name in Table or the empty string if there is
--  no parameter with this number.

function Get_Value
  (Table : Table_Type; N : Positive := 1) return String
with Post => (if N > Count (Table) then Get_Value'Result'Length = 0);
--  Returns the Nth Value in Table or the empty string if there is
--  no parameter with this number.

function Get (Table : Table_Type; N : Positive) return Element with
  Post => (if N > Count (Table) then Get'Result = Null_Element);
--  Returns N'th name/value pair. Returns Null_Element if there is no
--  such item in the table.

function Get_Names (Table : Table_Type) return VString_Array
with Post => Get_Names'Result'Length = Name_Count (Table);
```

(continues on next page)

(continued from previous page)

```
   --  Returns sorted array of unique key names

   function Get_Values
     (Table : Table_Type; Name : String) return VString_Array
   with Post => Get_Values'Result'Length = Count (Table, Name);
   --  Returns all values for the specified parameter key name

   generic
      with procedure Process (Name, Value : String);
   procedure Generic_Iterate_Names
     (Table : Table_Type; Separator : String);
   --  Iterates over all names in the table.
   --  All Values of the same name are separated by Separator string.

   procedure Iterate_Names
     (Table     : Table_Type;
      Separator : String;
      Process   : not null access procedure (Name, Value : String));

   function Union
     (Left   : Table_Type;
      Right  : Table_Type;
      Unique : Boolean) return Table_Type;
   --  Concatenates two tables, If Unique is True do not add Right container
   --  element into result when element with the same name already exists in
   --  the Left container.

   procedure Union
     (Left   : in out Table_Type;
      Right  : Table_Type;
      Unique : Boolean);
   --  Concatenates two tables and put result to Left, If Unique is True do not
   --  add Right container element into result when element with the same name
   --  already exists in the Left container.

   procedure Add (Table : in out Table_Type; Name, Value : String);

   procedure Add
     (Table       : in out Table_Type;
      Name, Value : Unbounded_String)
   with Post => Count (Table) = Count (Table'Old) + 1
                or else
                 Count (Table, To_String (Name))
                 = Count (Table'Old, To_String (Name)) + 1;
   --  Add a new Key/Value pair into Table. A new value is always added,
   --  even if there is already an entry with the same name.

   procedure Update
     (Table : in out Table_Type;
      Name  : String;
      Value : String;
      N     : Positive := 1);
```

(continues on next page)

(continued from previous page)

```
   procedure Update
     (Table : in out Table_Type;
      Name  : Unbounded_String;
      Value : Unbounded_String;
      N     : Positive := 1)
   with
     Pre  =>
       --  Count + 1 means it is added at the end of the table
       N <= Count (Table, To_String (Name)) + 1,
     Post =>
       --  Value already exists, it is updated
       (N <= Count (Table'Old, To_String (Name))
        and then Count (Table, To_String (Name))
                 = Count  (Table'Old, To_String (Name)))
       --  New value appended
       or else
         (N = Count (Table'Old, To_String (Name)) + 1
          and then N = Count (Table, To_String (Name)));
   --  Update the N-th Value with the given Name into the Table.
   --  The container could already have more than one value associated with
   --  this name.

   procedure Case_Sensitive (Table : in out Table_Type; Mode : Boolean);
   --  If Mode is True it will use all parameters with case sensitivity

   procedure Names_Lowercased (Table : in out Table_Type; Mode : Boolean);
   --  If Mode is True all names will be lowercased on put into the table

   procedure Reset (Table : in out Table_Type) with
     Post => Count (Table) = 0;
   --  Removes all object from Table. Table will be reinitialized and will be
   --  ready for new use.

private
   -- implementation removed
end AWS.Containers.Tables;
```

13.12 AWS.Cookie

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2010-2018, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

--  A package for basic HTTP state management, ie. cookies. Tokens and
--  attributes adhere to RFC-2109: http://tools.ietf.org/html/rfc2109

with AWS.Default;
with AWS.Response;
with AWS.Status;

package AWS.Cookie is

   use type AWS.Response.Data_Mode;

   Response_Data_Not_Initialized : exception;
   --  The Response_Data_Not_Initialized exception is raised when trying to add
   --  headers to an un-initialized AWS.Response.Data object.
   --  The AWS.Response.Data object is initialized using the
   --  AWS.Response.Build function.

   No_Max_Age : constant Duration;
   --  When no Max-Age is required, this value can be passed to the Set
   --  routines below.

   function Exists
```

(continues on next page)

(continued from previous page)

```
   (Request        : Status.Data;
    Key            : String;
    Case_Sensitive : Boolean := True) return Boolean;
--  Check if the 'Key' cookie exists in AWS.Headers.List. Return Boolean
--  True of the cookie exists, else Boolean False.

procedure Expire
   (Content : in out Response.Data;
    Key     : String;
    Path    : String := "/");
--  Expire the 'Key' cookie. This is done by setting the Max-Age attribute
--  to 0. The Value of the cookie is also set to "", in case a browser does
--  not honor the Max-Age attribute.

function Get
   (Request        : Status.Data;
    Key            : String;
    Case_Sensitive : Boolean := True) return String;
--  Return the 'Key' cookie from AWS.Headers.List. If the cookie does not
--  exist, return an empty string, ie. ""

function Get
   (Request        : Status.Data;
    Key            : String;
    Case_Sensitive : Boolean := True) return Integer;
--  Return the 'Key' cookie from AWS.Headers.List. If the cookie does not
--  exist or can't be converted from String to Integer then return 0.

function Get
   (Request        : Status.Data;
    Key            : String;
    Case_Sensitive : Boolean := True) return Float;
--  Return the 'Key' cookie from AWS.Headers.List. If the cookie does not
--  exist or can't be converted from String to Float then return 0.0.

function Get
   (Request        : Status.Data;
    Key            : String;
    Case_Sensitive : Boolean := True) return Boolean;
--  Return the 'Key' cookie from AWS.Headers.List. Only if the cookie value
--  equals "True" is Boolean True returned, else Boolean False is returned.

procedure Set
   (Content   : in out Response.Data;
    Key       : String;
    Value     : String;
    Comment   : String := "";
    Domain    : String := "";
    Max_Age   : Duration := Default.Ten_Years;
    Path      : String := "/";
    Secure    : Boolean := False;
    HTTP_Only : Boolean := False)
```

(continues on next page)

(continued from previous page)

```
   with Pre => Response.Mode (Content) /= Response.No_Data;
   --  Set a new cookie named 'Key' with value 'Value'. See RFC 2109 for more
   --  information about the individual cookie attributes:
   --    http://tools.ietf.org/html/rfc2109
   --
   --  Exceptions:
   --    Response_Data_Not_Initialized
   --      Is raised if AWS.Cookie.Set is called before the Content object has
   --      been initialized by a call to AWS.Response.Build

   procedure Set
     (Content   : in out Response.Data;
      Key       : String;
      Value     : Integer;
      Comment   : String := "";
      Domain    : String := "";
      Max_Age   : Duration := Default.Ten_Years;
      Path      : String := "/";
      Secure    : Boolean := False;
      HTTP_Only : Boolean := False)
   with Pre => Response.Mode (Content) /= Response.No_Data;
   --  Set a new cookie named 'Key' with Integer value 'Value'. The Integer is
   --  converted to a String, as both cookie keys and values are inherently
   --  strings.
   --
   --  Exceptions:
   --    Response_Data_Not_Initialized
   --      Is raised if AWS.Cookie.Set is called before the Content object has
   --      been initialized by a call to AWS.Response.Build

   procedure Set
     (Content   : in out Response.Data;
      Key       : String;
      Value     : Float;
      Comment   : String := "";
      Domain    : String := "";
      Max_Age   : Duration := Default.Ten_Years;
      Path      : String := "/";
      Secure    : Boolean := False;
      HTTP_Only : Boolean := False)
   with Pre => Response.Mode (Content) /= Response.No_Data;
   --  Set a new cookie named 'Key' with Float value 'Value'. The Float is
   --  converted to a String, as both cookie keys and values are inherently
   --  strings.
   --
   --  Exceptions:
   --    Response_Data_Not_Initialized
   --      Is raised if AWS.Cookie.Set is called before the Content object has
   --      been initialized by a call to AWS.Response.Build

   procedure Set
     (Content   : in out Response.Data;
```

(continues on next page)

(continued from previous page)

```
      Key       : String;
      Value     : Boolean;
      Comment   : String := "";
      Domain    : String := "";
      Max_Age   : Duration := Default.Ten_Years;
      Path      : String := "/";
      Secure    : Boolean := False;
      HTTP_Only : Boolean := False)
   with Pre => Response.Mode (Content) /= Response.No_Data;
   --  Set a new cookie named 'Key' with Boolean value 'Value'. The Boolean is
   --  converted to a String ("False" or "True"), as both cookie keys and
   --  values are inherently strings.
   --
   --  Exceptions:
   --    Response_Data_Not_Initialized
   --      Is raised if AWS.Cookie.Set is called before the Content object has
   --      been initialized by a call to AWS.Response.Build

private
   -- implementation removed
end AWS.Cookie;
```

13.13 AWS.Default

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2024, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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.

with System;

package AWS.Default with Pure is

   use System;

   --  All times are in seconds

   Ten_Years     : constant := 86_400.0 * 365 * 10;

   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;
```

(continues on next page)

(continued from previous page)

```
   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   := "";
   Admin_Password                  : constant String   := "";
   Admin_Realm                     : constant String   := "AWS Admin Page";
   Protocol_Family                 : constant String   := "FAMILY_UNSPEC";
   IPv6_Only                       : constant Boolean  := False;
   Server_Port                     : constant          := 8080;
   Hotplug_Port                    : constant          := 8888;
   Max_Connection                  : constant          := 5;
   Max_WebSocket_Handler           : constant          := 2;
   Max_WebSocket                   : constant          := 512;
   WebSocket_Message_Queue_Size    : constant          := 10;
   WebSocket_Send_Message_Queue_Size : constant        := 30;
   WebSocket_Timeout               : constant Duration := Eight_Hours;
   Send_Buffer_Size                : constant          := 0;
   TCP_No_Delay                    : constant Boolean  := False;
   Free_Slots_Keep_Alive_Limit     : constant          := 1;
   Keep_Alive_Force_Limit          : constant          := 0;
   Keep_Alive_Close_Limit          : constant          := 0;
   Accept_Queue_Size               : constant          := 64;
   Upload_Directory                : constant String   := "";
   Upload_Size_Limit               : constant          := 16#500_000#;
   Line_Stack_Size                 : constant          := 16#150_000#;
   Case_Sensitive_Parameters       : constant Boolean  := True;
   Input_Line_Size_Limit           : constant          := 16#4000#;
   Max_POST_Parameters             : constant          := 100;
   Max_Concurrent_Download         : constant          := 25;
   Reuse_Address                   : constant Boolean  := False;
   Close_On_Exec                   : constant Boolean  := False;
   MIME_Types                      : constant String   := "aws.mime";

   HTTP2_Activated                 : constant Boolean  := True;
   HTTP2_Header_Table_Size         : constant          := 4_096;
   HTTP2_Enable_Push               : constant Boolean  := False;
   HTTP2_Max_Concurrent_Streams    : constant          := 250;
   HTTP2_Initial_Window_Size       : constant          := 65_535;
   HTTP2_Max_Frame_Size            : constant          := 16_384;
   HTTP2_Max_Header_List_Size      : constant          := 1_048_576;

   --  Client configuration

   User_Agent                    : constant String :=
                                     "AWS (Ada Web Server) v" & Version;
   Server_Header                 : constant String :=
                                     User_Agent;
```

(continues on next page)

(continued from previous page)

```
   --  Log values. The character '@' in the error log filename prefix is
   --  replaced by the running program name.

   Log_Activated                   : constant Boolean := False;
   Log_File_Directory              : constant String := "./";

   Log_Split_Mode                  : constant String := "NONE";
   Log_Filename_Prefix             : constant String := "@";

   Error_Log_Activated             : constant Boolean := False;
   Error_Log_Split_Mode            : constant String := "NONE";
   Error_Log_Filename_Prefix       : constant String := "@_error";

   Log_Size_Limit                  : constant Natural := 0;

   --  Session

   Session                         : constant Boolean  := False;
   Session_Name                    : constant String   := "AWS";
   Session_Private_Name            : constant String   := "AWS_Private";
   Session_Cleanup_Interval        : constant Duration := Five_Minutes;
   Session_Lifetime                : constant Duration := Ten_Minutes;
   Session_Id_Length               : constant Positive := 11;

   --  Context

   Context_Lifetime                : constant Duration := Eight_Hours;

   --  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.thtml";
```

(continues on next page)

(continued from previous page)

```
   --  Status page

   Status_Page                    : constant String := "aws_status.thtml";
   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  := "TLS";
   Config_Directory               : constant String  := ".config/ada-web-srv";
   Disable_Program_Ini            : constant Boolean := False;
   Cipher_Priorities              : constant String  := "";
   TLS_Ticket_Support             : constant Boolean := False;
   Server_Certificate             : constant String  := "aws-server.crt";
   Server_Key                     : constant String  := "aws-server.key";
   Client_Certificate             : constant String  := "cert.pem";
   Exchange_Certificate           : constant Boolean := True;
   Check_Certificate              : constant Boolean := True;
   Trusted_CA                     : constant String  := "";
   CRL_File                       : constant String  := "";
   Check_URL_Validity             : constant Boolean := True;
   SSL_Session_Cache_Size         : constant         := 16#4000#;

   --  Priorities

   Server_Priority                : constant Any_Priority := Default_Priority;
   WebSocket_Priority             : constant Any_Priority := Default_Priority;
   Session_Cleaner_Priority       : constant Any_Priority := Default_Priority;
   Service_Priority               : constant Any_Priority := Default_Priority;

end AWS.Default;
```

13.14 AWS.Dispatchers

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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
     and AWS.Utils.Clonable with private;

   function Dispatch
     (Dispatcher : Handler;
      Request    : Status.Data) return Response.Data is abstract;
   --  Call the appropriate inherited dispatcher
```

(continues on next page)

(continued from previous page)

```
   function Ref_Counter (Dispatcher : 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) with Inline;
   --  Release memory associated with the dispatcher

private
   -- implementation removed
end AWS.Dispatchers;
```

13.15 AWS.Dispatchers.Callback

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  Dispatch on a Callback procedure

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 : Response.Callback) return Handler
     with Inline;
   --  Build a dispatcher for the specified callback

private
   -- implementation removed
end AWS.Dispatchers.Callback;
```

13.16 AWS.Exceptions

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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

   use Ada.Exceptions;

   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.
```

(continues on next page)

(continued from previous page)

```
   end record;

   type Unexpected_Exception_Handler is not null access
     procedure (E      : Exception_Occurrence;
                Log    : in out AWS.Log.Object;
                Error  : 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;
```

13.17 AWS.Headers

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

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.

   Empty_List : constant List;

   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
```

(continues on next page)

(continued from previous page)

```
   procedure Send_Header
     (Socket    : Net.Socket_Type'Class;
      Headers   : List;
      End_Block : Boolean := False);
   --  Send all header lines in Headers list to the socket

   generic
      with procedure Data (Value : String);
   procedure Get_Content
     (Headers   : List;
      End_Block : Boolean := False);

   function Get_Line (Headers : List; N : Positive) return String with
     Post =>
       (N > Count (Headers) and then Get_Line'Result'Length = 0)
       or else N <= Count (Headers);
   --  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 : List; Name : 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 : AWS.Headers.List) return Natural;
   --  Returns the length (in bytes) of the header, including the ending
   --  empty line.

   generic
      with function Get_Line return String;
   procedure Read_G (Headers : in out List);

   procedure Read (Headers : in out List; Socket : Net.Socket_Type'Class);
   --  Read and parse HTTP header from the socket

   overriding procedure Reset (Headers : in out List)
     with Post => Headers.Count = 0;
   --  Removes all object from Headers. Headers will be reinitialized and will
   --  be ready for new use.

   procedure Debug (Activate : Boolean);
```

(continues on next page)

(continued from previous page)

```
   --  Turn on Debug output

   procedure Debug_Print (Headers : List);
   --  Print headers to output if debug flag set

   --  See AWS.Containers.Tables for inherited routines

private
   -- implementation removed
end AWS.Headers;
```

13.18 AWS.Headers.Values

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2002-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;

   Format_Error : exception renames Headers.Format_Error;

   --  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 =>
```

(continues on next page)

(continued from previous page)

```
         null;
   end case;
end record;

type Set is array (Positive range <>) of Data;

-----------
-- Parse --
-----------

generic

   with procedure Value (Item : String; Quit : in out Boolean);
   --  Called for every un-named value read from the header value

   with procedure Named_Value
     (Name  : String;
      Value : String;
      Quit  : in out Boolean);
   --  Called for every named value read from the header value

procedure Parse (Header_Value : 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 : String) return Set;
--  Returns a Set with each named and un-named values splited from Data

function Index
  (Set            : Values.Set;
   Name           : String;
   Case_Sensitive : 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   : String;
   Name           : String;
   Case_Sensitive : 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.
```

(continues on next page)

(continued from previous page)

```
   function Get_Unnamed_Value
     (Header_Value : String; N : Positive := 1) return String;
   --  Returns N-th un-named value from Header_Value

   function Unnamed_Value_Exists
     (Header_Value   : String;
      Value          : String;
      Case_Sensitive : Boolean := True) return Boolean;
   --  Returns True if the unnamed value specified has been found in
   --  Header_Value.

end AWS.Headers.Values;
```

13.19 AWS.Jabber

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2002-2013, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

package AWS.Jabber with Pure is

end AWS.Jabber;
```

13.20 AWS.LDAP.Client

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 add, read, modify and delete information from a LDAP
--  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.Containers.Indefinite_Vectors;
with Ada.Exceptions;
with Ada.Strings.Unbounded;

with AWS.LDAP.Thin;

package AWS.LDAP.Client is

   use Ada.Exceptions;
   use Ada.Strings.Unbounded;

   LDAP_Error : exception renames LDAP.LDAP_Error;

   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
```

(continues on next page)

(continued from previous page)

```
--  and terminated with Unbind.

subtype LDAP_Message is Thin.LDAPMessage;
--  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_LDAPMessage;

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 : Exception_Occurrence) return Thin.Return_Code;
--  Returns the error code in the LDAP_Error exception occurence 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 : 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 : String := "") return String;
--  Returns the uid attribute, if Val is specified "=<Val>" is
--  added after the attribute name.

function givenName (Val : String := "") return String;
--  Returns the given name (firstname) attribute. if Val is specified
--  "=<Val>" is added after the attribute name.
```

(continues on next page)

(continued from previous page)

```
   function cn (Val : String := "") return String;
   function commonName (Val : String := "") return String renames cn;
   --  Returns the common Name attribute, if Val is specified "=<Val>" is
   --  added after the attribute name.

   function sn (Val : String := "") return String;
   function surname (Val : String := "") return String renames sn;
   --  Returns the surname attribute, if Val is specified "=<Val>" is
   --  added after the attribute name.

   function telephoneNumber (Val : 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 : String := "") return String;
   --  Returns the mail attribute. if Val is specified "=<Val>" is added after
   --  the attribute name.

   function l (Val : String := "") return String;
   function localityName (Val : String := "") return String renames l;
   --  Returns the locality attribute, if Val is specified "=<Val>" is
   --  added after the attribute name.

   function o (Val : String := "") return String;
   function organizationName (Val : String := "") return String renames o;
   --  Returns the organization attribute, if Val is specified "=<Val>" is
   --  added after the attribute name.

   function ou (Val : String := "") return String;
   function organizationalUnitName (Val : String := "") return String
     renames ou;
   --  Returns the organizational unit attribute, if Val is specified "=<Val>"
   --  is added after the attribute name.

   function st (Val : String := "") return String;
   function stateOrProvinceName (Val : String := "") return String
     renames st;
   --  Returns the state name attribute, if Val is specified "=<Val>" is
   --  added after the attribute name.

   function c (Val : String := "") return String;
   function countryName (Val : 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 : String := "") return String;
   function domainComponent (Val : String := "") return String renames dc;
   --  Returns a domain component attribute, if Val is specified "=<Val>" is
   --  added after the attribute name.
```

(continues on next page)

(continued from previous page)

```
   function Cat
     (S1, S2, S3, S4, S5, S6, S7, S8, S9, S10 : 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 : String;
      Port : 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      : Directory;
      Login    : String;
      Password : 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 : Directory) return Boolean;
   --  Returns True if the directory has correctly been initialized and binded
   --  with the server.

   ------------
   -- Search --
   ------------

   function Search
     (Dir        : Directory;
      Base       : String;
      Filter     : String;
      Scope      : Scope_Type    := LDAP_Scope_Default;
      Attrs      : Attribute_Set := Null_Set;
      Attrs_Only : 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.

   ----------------------
   -- Add/Modify/Delete --
   ----------------------

   type Mod_Type is (LDAP_Mod_Add, LDAP_Mod_Replace, LDAP_Mod_BValues);
```

(continues on next page)

(continued from previous page)

```
--  Modification types: Add, Replace and BER flag

type Mod_Element (Values_Size : Natural) is record
   Mod_Op     : Mod_Type;
   Mod_Type   : Unbounded_String;
   Mod_Values : Attribute_Set (1 .. Values_Size);
end record;
--  Holds modification elements. 'Abstraction' of the LDAPMod_Element type
--  used in the thin-binding. Mod_Values is static to make it less complex.

package LDAP_Mods is
  new Ada.Containers.Indefinite_Vectors (Positive, Mod_Element);
--  Vector-based Storage for all modification elements. Will be
--  mapped to C LDAPMod **.

procedure Add
  (Dir  : Directory;
   DN   : String;
   Mods : LDAP_Mods.Vector);
--  Add an entry specified by 'DN' to the LDAP server. The Mods-Vector
--  contains the attributes for the entry.

procedure Modify
  (Dir  : Directory;
   DN   : String;
   Mods : LDAP_Mods.Vector);
--  Modify an attribute of entry specified by 'DN'. The Mods-Vector
--  contains the attributes to add/replace/delete for the entry.

procedure Delete (Dir : Directory; DN : String);
--  Delete an entry specified by 'DN' from the LDAP server

---------------
-- Iterators --
---------------

function First_Entry
  (Dir   : Directory;
   Chain : LDAP_Message) return LDAP_Message;
--  Returns the first entry (or Node) for the search result (Chain)

function Next_Entry
  (Dir     : Directory;
   Entries : LDAP_Message) return LDAP_Message;
--  Returns next entry (or Node) for Entries

function Count_Entries
  (Dir   : Directory;
   Chain : LDAP_Message) return Natural;
--  Returns the number of entries in the search result (Chain)

procedure Free (Chain : LDAP_Message);
```

(continues on next page)

(continued from previous page)

```
   --  Release memory associated with the search result Chain

   generic
      with procedure Action
        (Node : LDAP_Message;
         Quit : in out Boolean);
   procedure For_Every_Entry (Dir : Directory; Chain : 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  : Directory;
      Node : LDAP_Message;
      BER  : not null access BER_Element) return String;
   --  Returns the first attribute for the entry. It initialize an iteraror
   --  (the BER structure). The BER structure must be released after used by
   --  using the Free routine below.

   function Next_Attribute
     (Dir  : Directory;
      Node : LDAP_Message;
      BER  : BER_Element) return String;
   --  Returns next attribute for iterator BER. First_Attribute must have been
   --  called to initialize this iterator.

   procedure Free (BER : BER_Element);
   --  Releases memory associated with the BER structure which has been
   --  allocated by the First_Attribute routine.

   generic
      with procedure Action
        (Attribute : String;
         Quit      : in out Boolean);
   procedure For_Every_Attribute
     (Dir  : Directory;
      Node : 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  : Directory;
      Node : LDAP_Message) return String;
   --  Returns the distinguished name for the given entry Node

   function DN2UFN (DN : String) return String;
   --  Returns a distinguished name converted to a user-friendly format
```

(continues on next page)

(continued from previous page)

```
   function Get_Values
     (Dir    : Directory;
      Node   : LDAP_Message;
      Target : String) return String_Set;
   --  Returns the list of values of a given attribute (Target) found in entry
   --  Node.

   function Explode_DN
     (DN       : String;
      No_Types : 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;
```

13.21 AWS.Log

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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
--  '<progname>-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 AWS.Containers.String_Vectors;
with AWS.Headers;
with AWS.Messages;
with AWS.Response;
with AWS.Status;

private with Ada.Containers.Indefinite_Ordered_Maps;
private with Ada.Finalization;
private with Ada.Strings.Unbounded;
private with Ada.Text_IO;
private with AWS.Utils;

package AWS.Log is
```

(continues on next page)

(continued from previous page)

```
   type Object is limited private;
   --  A log object. It must be activated by calling Start below

   type Callback is access procedure (Message : String);
   --  Access to a procedure that handles AWS access and/or error log data.
   --  If the access and/or error logs are started with a Callback procedure
   --  set, then AWS will no longer handle writing the log data to file, nor
   --  will it rotate or split the data. In short : If you set a Callback, it's
   --  up to you to handle these things.
   --  The raw log data generated by AWS is simply handed verbatim to the
   --  Callback procedure.

   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           : Split_Mode := None;
      Size_Limit      : Natural    := 0;
      File_Directory  : String     := Not_Specified;
      Filename_Prefix : String     := Not_Specified;
      Auto_Flush      : Boolean    := False);
   --  Activate server's activity logging. Split indicate the way the log file
   --  should be created. If Size_Limit more than zero and size of log file
   --  become more than Size_Limit, log file would be splitted. Filename_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 Start
     (Log    : in out Object;
      Writer : Callback;
      Name   : String);
   --  Activate server's activity logging and send all log data to Callback.
   --  When the logging object is started with a Callback no splitting or size
   --  limits are imposed on the logging data. This will all have to be handled
   --  in the Callback.
   --  When a log is started with a Callback, all log data is passed verbatim
```

(continues on next page)

(continued from previous page)

```
   --  to the Callback.
   --  The Name String is returned when the Filename function is called. This
   --  serves no other function than to label the Callback procedure.

   procedure Register_Field (Log : in out Object; Id : String);
   --  Register field to be written into extended log format

   procedure Set_Field
     (Log : Object; Data : in out Fields_Table; Id, Value : 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    : Object;
      Data   : in out Fields_Table;
      Prefix : String;
      Header : 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 : Status.Data;
      Answer       : Response.Data);
   --  Write log info if activated (i.e. Start routine above has been called)

   procedure Write
     (Log            : in out Object;
      Connect_Stat   : Status.Data;
      Status_Code    : Messages.Status_Code;
      Content_Length : 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 : Status.Data;
      Data         : String);
   --  Write user's log info if activated.  (i.e. Start routine above has been
   --  called).

   procedure Write (Log : in out Object; Data : String);
   --  Write Data into the log file. This Data is unstructured, only a time
```

(continues on next page)

(continued from previous page)

```
   --  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.
   --  If a Callback procedure is used to handle the log data, then calling
   --  Flush does nothing.

   procedure Stop (Log : in out Object);
   --  Stop logging activity

   function Is_Active (Log : Object) return Boolean;
   --  Returns True if Log is activated

   function Filename (Log : Object) return String;
   --  Returns current log filename or the empty string if the log is not
   --  activated.
   --  If a Callback is used to handle the log, then the name given in the
   --  Start procedure is returned. See the Start procedure for starting logs
   --  with a Callback.

   function Mode (Log : Object) return Split_Mode;
   --  Returns the split mode. None will be returned if log is not activated or
   --  a Callback procedure is used to handle the log data.

private
   -- implementation removed
end AWS.Log;
```

13.22 AWS.Messages

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

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

package AWS.Messages is

   use Ada;
   use Ada.Streams;
   use Ada.Strings.Unbounded;

   -----------------
   -- HTTP tokens --
   -----------------

   HTTP_Token    : constant String := "HTTP/";
   Options_Token : constant String := "OPTIONS";
   Get_Token     : constant String := "GET";
   Head_Token    : constant String := "HEAD";
   Post_Token    : constant String := "POST";
   Put_Token     : constant String := "PUT";
```

(continues on next page)

(continued from previous page)

```
   Delete_Token  : constant String := "DELETE";
   Trace_Token   : constant String := "TRACE";
   Connect_Token : constant String := "CONNECT";
   --  Sorted like in RFC 2616 Method definition

   ------------------------
   -- HTTP header tokens --
   ------------------------

   --  HTTP/2 header tokens RFC 7540

   Status_Token              : constant String := ":status";
   Method_Token              : constant String := ":method";
   Path2_Token               : constant String := ":path";
   Scheme_Token              : constant String := ":scheme";

   H2_Token                  : constant String := "h2";
   H2C_Token                 : constant String := "h2c";

   --  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";

   --  Cross-Origin Resource Sharing request header tokens
```

(continues on next page)

(continued from previous page)

```
   Access_Control_Request_Headers_Token : constant String :=
                                            "Access-Control-Request-Headers";
   Access_Control_Request_Method_Token  : constant String :=
                                            "Access-Control-Request-Method";
   Origin_Token                         : constant String := "Origin";

   --  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";

   --  Cross-Origin Resource Sharing response header tokens
   Access_Control_Allow_Credentials_Token : constant String :=
     "Access-Control-Allow-Credentials";
   Access_Control_Allow_Headers_Token     : constant String :=
                                              "Access-Control-Allow-Headers";
   Access_Control_Allow_Methods_Token     : constant String :=
                                              "Access-Control-Allow-Methods";
   Access_Control_Allow_Origin_Token      : constant String :=
                                              "Access-Control-Allow-Origin";
   Access_Control_Expose_Headers_Token    : constant String :=
                                              "Access-Control-Expose-Headers";
   Access_Control_Max_Age_Token           : constant String :=
                                              "Access-Control-Max-Age";

   --  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";

   --  Cookie token RFC 2109
   Cookie_Token              : constant String := "Cookie";
   Set_Cookie_Token          : constant String := "Set-Cookie";
   Comment_Token             : constant String := "Comment";
   Domain_Token              : constant String := "Domain";
   Max_Age_Token             : constant String := "Max-Age";
   Path_Token                : constant String := "Path";
   Secure_Token              : constant String := "Secure";
   HTTP_Only_Token           : constant String := "HttpOnly";
```

(continues on next page)

(continued from previous page)

```
   --  Other tokens
   Proxy_Connection_Token          : constant String := "Proxy-Connection";
   Content_Disposition_Token       : constant String := "Content-Disposition";
   SOAPAction_Token                : constant String := "SOAPAction";
   Content_Id_Token                : constant String := "Content-ID";
   Content_Transfer_Encoding_Token : constant String :=
                                       "Content-Transfer-Encoding";

   --  WebSockets tokens
   Websocket_Token              : constant String := "WebSocket";
   Sec_WebSocket_Accept_Token   : constant String := "Sec-WebSocket-Accept";
   Sec_WebSocket_Protocol_Token : constant String := "Sec-WebSocket-Protocol";
   Sec_WebSocket_Key_Token      : constant String := "Sec-WebSocket-Key";
   Sec_WebSocket_Key1_Token     : constant String := "Sec-WebSocket-Key1";
   Sec_WebSocket_Key2_Token     : constant String := "Sec-WebSocket-Key2";
   Sec_WebSocket_Version_Token  : constant String := "Sec-WebSocket-Version";
   Sec_WebSocket_Origin_Token   : constant String := "Sec-WebSocket-Origin";
   Sec_WebSocket_Location_Token : constant String := "Sec-WebSocket-Location";
   Chat_Token                   : constant String := "chat";

   S100_Continue : constant String := "100-continue";
   --  Supported expect header value

   --  HTTP2 specific

   HTTP2_Settings : constant String := "HTTP2-Settings";

   -----------------
   -- Status Code --
   -----------------

   type Status_Code is
     (S100, S101, S102,
      --  1xx : Informational - Request received, continuing process

      S200, S201, S202, S203, S204, S205, S206, S207, S208, S226,
      --  2xx : Success - The action was successfully received, understood and
      --  accepted

      S300, S301, S302, S303, S304, S305, S306, S307, S308,
      --  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, S418, S421, S422, S423,
      S424, S425, S426, S428, S429, S431, S451,
      --  4xx : Client Error - The request contains bad syntax or cannot be
      --  fulfilled

      S500, S501, S502, S503, S504, S505, S506, S507, S508, S510, S511, S520,
      S521, S522, S523, S524, S525, S526
```

(continues on next page)

(continued from previous page)

```
   --  5xx : Server Error - The server failed to fulfill an apparently
   --  valid request
   );

subtype Informational is Status_Code range S100 .. S102;
subtype Success       is Status_Code range S200 .. S226;
subtype Redirection   is Status_Code range S300 .. S308;
subtype Client_Error  is Status_Code range S400 .. S451;
subtype Server_Error  is Status_Code range S500 .. S526;

function Image (S : Status_Code) return String;
--  Returns Status_Code image. This value does not contain the leading S

function Reason_Phrase (S : Status_Code) return String;
--  Returns the reason phrase for the status code S, see [RFC 2616 - 6.1.1]

function With_Body (S : Status_Code) return Boolean;
--  Returns True if message with status can have a body

----------------------
-- 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 four 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;

type Cache_Kind is (Request, Response);

type Delta_Seconds is new Integer range -1 .. Integer'Last;
--  Represents a delta-seconds parameter for some Cache_Data fields like
--  max-age, max-stale (value -1 is used for Unset).
```

(continues on next page)

(continued from previous page)

```
   Unset         : constant Delta_Seconds;
   No_Max_Stale  : constant Delta_Seconds;
   Any_Max_Stale : constant Delta_Seconds;

   type Private_Option is new Unbounded_String;

   All_Private   : constant Private_Option;
   Private_Unset : constant Private_Option;

   --  Cache_Data is a record that represents cache control information

   type Cache_Data (CKind : Cache_Kind) is record
      No_Cache       : Boolean       := False;
      No_Store       : Boolean       := False;
      No_Transform   : Boolean       := False;
      Max_Age        : Delta_Seconds := Unset;

      case CKind is
         when Request =>
            Max_Stale      : Delta_Seconds := Unset;
            Min_Fresh      : Delta_Seconds := Unset;
            Only_If_Cached : Boolean       := False;

         when Response =>
            S_Max_Age        : Delta_Seconds  := Unset;
            Public           : Boolean        := False;
            Private_Field    : Private_Option := Private_Unset;
            Must_Revalidate  : Boolean        := False;
            Proxy_Revalidate : Boolean        := False;
      end case;
   end record;

   function To_Cache_Option (Data : Cache_Data) return Cache_Option;
   --  Returns a cache control value for an HTTP request/response, fields are
   --  described into RFC 2616 [14.9 Cache-Control].

   function To_Cache_Data
     (Kind : Cache_Kind; Value : Cache_Option) return Cache_Data;
   --  Returns a Cache_Data record parsed out of Cache_Option

   ----------
   -- ETag --
   ----------

   type ETag_Value is new String;

   function Create_ETag
     (Name : String; Weak : Boolean := False) return ETag_Value;
   --  Returns an ETag value (strong by default and Weak if specified). For a
   --  discussion about ETag see RFC 2616 [3.11 Entity Tags] and [14.19 ETag].
```

(continues on next page)

(continued from previous page)

```
   ------------------------------
   -- HTTP message constructors --
   ------------------------------

   function Accept_Encoding (Encoding : String) return String with Inline;

   function Accept_Type (Mode : String) return String with Inline;

   function Accept_Language (Mode : String) return String with Inline;

   function Authorization (Mode, Password : String) return String with Inline;

   function Connection (Mode : String) return String with Inline;

   function Content_Length (Size : Stream_Element_Offset) return String
     with Inline;

   function Cookie (Value : String) return String with Inline;

   function Content_Type (Format : String) return String with Inline;

   function Content_Type
     (Format : String; Boundary : String) return String with Inline;

   function Cache_Control (Option : Cache_Option) return String with Inline;

   function Cache_Control (Data : Cache_Data) return String with Inline;

   function Content_Disposition
     (Format, Name, Filename : String) return String with Inline;
   --  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 Date (Date : Calendar.Time) return String with Inline;
   --  The date header

   function ETag (Value : ETag_Value) return String with Inline;

   function Expires (Date : Calendar.Time) return String with Inline;
   --  The date should not be more than a year in the future, see RFC 2616
   --  [14.21 Expires].

   function Host (Name : String) return String with Inline;

   function Last_Modified (Date : Calendar.Time) return String with Inline;

   function Location (URL : String) return String with Inline;

   function Proxy_Authorization (Mode, Password : String) return String
     with Inline;
```

(continues on next page)

(continued from previous page)

```
   function Proxy_Connection (Mode : String) return String with Inline;

   function Data_Range (Value : String) return String with Inline;

   function SOAPAction (URI : String) return String with Inline;

   function Status_Line
     (Code          : Status_Code;
      Reason_Phrase : String := "") return String with Inline;
   --  The HTTP status line on the form: HTTP/1.1 <code> <reason>

   function Status_Value
     (Code          : Status_Code;
      Reason_Phrase : String := "") return String with Inline;
   --  As above but only with the values : <code> <reason>

   function Transfer_Encoding (Encoding : String) return String with Inline;

   function User_Agent (Name : String) return String with Inline;

   function WWW_Authenticate (Realm : String) return String with Inline;
   --  Basic authentication request

   function WWW_Authenticate
     (Realm, Nonce : String; Stale : Boolean) return String with Inline;
   --  Digest authentication request

   function Sec_WebSocket_Accept (Key : String) return String with Inline;

   ----------------------
   --  helper functions --
   ----------------------

   function To_HTTP_Date (Time : 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 : 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;
```

13.23 AWS.MIME

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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_Javascript             : constant String := "text/javascript";
   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 --
```

(continues on next page)

(continued from previous page)

```
   -----------

   Image_Gif                   : constant String := "image/gif";
   Image_Jpeg                  : constant String := "image/jpeg";
   Image_Png                   : constant String := "image/png";
   Image_SVG                   : constant String := "image/svg+xml";
   Image_Tiff                  : constant String := "image/tiff";
   Image_Icon                  : constant String := "image/x-icon";
   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_XML             : constant String := "application/xml";
   Application_JSON            : constant String := "application/json";
   Application_SOAP            : constant String := "application/soap";
   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";
```

(continues on next page)

(continued from previous page)

```
   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 : String; MIME_Type : 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 : String; MIME_Type : 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 : String;
       Default  : 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 : String) return String;
```

(continues on next page)

(continued from previous page)

```
   --  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 : String) return Boolean;
   --  Returns True if the MIME_Type is a text data

   function Is_Audio (MIME_Type : String) return Boolean;
   --  Returns True if the MIME_Type is an audio data

   function Is_Image (MIME_Type : String) return Boolean;
   --  Returns True if the MIME_Type is an image data

   function Is_Video (MIME_Type : String) return Boolean;
   --  Returns True if the MIME_Type is a video data

   function Is_Application (MIME_Type : String) return Boolean;
   --  Returns True if the MIME_Type is an application data

   procedure Load (MIME_File : String);
   --  Load MIME_File, record every MIME type. Note that the format of this
   --  file follows the common standard format used by Apache mime.types.

end AWS.MIME;
```

13.24 AWS.Net

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2022, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  There 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 rountine
--  below. The corresponding implementation will be selected at build time.

with Ada.Containers.Doubly_Linked_Lists;
with Ada.Exceptions;
with Ada.Finalization;
with Ada.Streams;

private with AWS.Containers.Key_Value;
private with AWS.Utils;
private with Interfaces.C;

package AWS.Net is

   use Ada;
   use Ada.Exceptions;
   use Ada.Streams;
```

(continues on next page)

(continued from previous page)

```
   Socket_Error : exception;
   --  Raised by all routines below, a message will indicate the nature of
   --  the error.

   type Socket_Type is abstract new Finalization.Controlled with private;
   type Socket_Access is access all Socket_Type'Class;

   type Socket_Set is array (Positive range <>) of Socket_Access;

   package Socket_Lists is new Containers.Doubly_Linked_Lists
     (Socket_Access);

   subtype Socket_List is Socket_Lists.List;

   subtype FD_Type is Integer;
   --  Represents an external socket file descriptor

   No_Socket : constant := -1;
   --  Represents closed socket file descriptor

   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.

   type Family_Type is (Family_Inet, Family_Inet6, Family_Unspec);

   type Shutmode_Type is (Shut_Read, Shut_Write, Shut_Read_Write);

   Forever : constant Duration;
   --  The longest delay possible on the implementation

   ----------------
   -- Initialize --
   ----------------

   function Socket (Security : Boolean) return Socket_Type'Class;
   --  Create an uninitialized socket

   function Socket
     (Security : Boolean) return not null access Socket_Type'Class;
   --  Create a dynamically allocated uninitialized socket

   procedure Bind
```

(continues on next page)

(continued from previous page)

```
   (Socket        : in out Socket_Type;
    Port          : Natural;
    Host          : String      := "";
    Reuse_Address : Boolean     := False;
    IPv6_Only     : Boolean     := False;
    Family        : Family_Type := Family_Unspec) 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.
  --  IPv6_Only has meaning only for Family = Family_Inet6 and mean that only
  --  IPv6 clients allowed to connect.

  procedure Listen
    (Socket : Socket_Type; Queue_Size : Positive := 5) is abstract;
  --  Set the queue size of the socket

  procedure Accept_Socket
    (Socket : 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.

  procedure Set_Close_On_Exec (Socket : Socket_Type) is abstract;
  --  Set the close on exec socket flags

  type Socket_Constructor is not null access
    function (Security : Boolean) return Socket_Type'Class;

  procedure Connect
    (Socket : in out Socket_Type;
     Host   : String;
     Port   : Positive;
     Wait   : Boolean     := True;
     Family : Family_Type := Family_Unspec) is abstract
  with Pre'Class => Host'Length > 0;
  --  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 estabilised. 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 : Socket_Type;
     How    : Shutmode_Type := Shut_Read_Write) is abstract;
  --  Shutdown the read, write or both side of the socket.
  --  If How is Both, close it. Does not raise Socket_Error if the socket is
  --  not connected or already shutdown.
```

(continues on next page)

(continued from previous page)

```
   procedure Free (Socket : in out Socket_Access);
   --  Release memory associated with the socket

   --------
   -- IO --
   --------

   procedure Send
     (Socket : Socket_Type'Class; Data : Stream_Element_Array);
   --  Send Data chunk to the socket

   procedure Send
     (Sockets : Socket_Set; Data : Stream_Element_Array);
   --  Send Data to all sockets from the socket set. This call will ensure that
   --  the data are sent in priority to client waiting for reading. That is,
   --  slow connection for one sokcet should not delay the fast connections.
   --  Yet, this routine will return only when the data is sent to all sockets.

   procedure Send
     (Socket : Socket_Type;
      Data   : 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
   --  Data'First - 1. If Data'First is equal to Stream_Element_Offset'First
   --  then constraint error is raised to follow advice in AI95-227.

   procedure Receive
     (Socket : 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 : Socket_Type'Class;
      Max    : 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 : Socket_Type) return Stream_Element_Count
      is abstract;
   --  Returns the number of bytes which are available inside socket
   --  for immediate read.

   function Output_Space (Socket : Socket_Type) return Stream_Element_Offset;
   --  Returns the free space in output buffer in bytes. If OS could not
   --  provide such information, routine returns -1.
```

(continues on next page)

(continued from previous page)

```
   function Output_Busy (Socket : Socket_Type) return Stream_Element_Offset;
   --  How many bytes in the send queue. If OS could not provide such
   --  information, routine returns -1.

   ------------
   -- Others --
   ------------

   function Get_FD (Socket : Socket_Type) return FD_Type is abstract;
   --  Returns the file descriptor associated with the socket

   function Peer_Addr (Socket : Socket_Type) return String is abstract;
   --  Returns the peer name/address

   function Peer_Port (Socket : Socket_Type) return Positive is abstract;
   --  Returns the port of the peer socket

   function Get_Addr (Socket : Socket_Type) return String is abstract;
   --  Returns the name/address of the socket

   function Get_Port (Socket : Socket_Type) return Positive is abstract;
   --  Returns the port of the socket

   function Is_Any_Address (Socket : Socket_Type) return Boolean;
   --  Return true if the socket accepts connections on any of the hosts's
   --  network addresses.

   function Is_IPv6 (Socket : Socket_Type) return Boolean;

   function Is_Listening (Socket : Socket_Type) return Boolean;
   --  Returns true if the socket has been marked to accept connections with
   --  listen.

   function Is_Secure (Socket : Socket_Type) return Boolean is abstract;
   --  Returns True if socket is secure

   function IPv6_Available return Boolean;
   --  Returns True if IPv6 available in OS and in AWS socket implementation

   function Host_Name return String;
   --  Returns the running host name

   procedure Set_Send_Buffer_Size
     (Socket : Socket_Type; Size : 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 : Socket_Type; Size : Natural) is abstract;
   --  Set the internal socket receive buffer size.
   --  Do not confuse with buffers for the AWS.Net.Buffered operations.
```

(continues on next page)

(continued from previous page)

```
   function Get_Send_Buffer_Size (Socket : 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 : Socket_Type) return Natural
      is abstract;
   --  Returns the internal socket receive buffer size.
   --  Do not confuse with buffers for the AWS.Net.Buffered operations.

   function Cipher_Description (Socket : Socket_Type) return String;
   --  Returns cipher description on SSL implementation or empty string on
   --  plain socket.

   procedure Set_Blocking_Mode
     (Socket : in out Socket_Type; Blocking : Boolean);
   pragma Obsolescent ("Use Set_Timeout instead");
   --  Set the blocking mode for the socket

   procedure Set_Timeout (Socket : in out Socket_Type; Timeout : Duration)
     with Inline;
   --  Sets the timeout for the socket read/write operations

   procedure Set_No_Delay
     (Socket : Socket_Type; Value : Boolean := True) is null;
   --  Set/clear TCP_NODELAY option on socket

   function Wait
     (Socket : Socket_Type'Class;
      Events : 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 occured.

   function Check
     (Socket : Socket_Type'Class;
      Events : Wait_Event_Set) return Event_Set;
   --  Check for Input/Output/Error events availability.
   --  No wait for socket timeout.

   function Poll
     (Socket  : Socket_Type'Class;
      Events  : Wait_Event_Set;
      Timeout : Duration) return Event_Set;
   --  Wait events on socket descriptor for specified Timeout

   function Errno (Socket : Socket_Type) return Integer is abstract;
   --  Returns and clears error state in socket

   function Is_Timeout
     (Socket : Socket_Type;
      E      : Exception_Occurrence) return Boolean;
```

(continues on next page)

(continued from previous page)

```
   --  Returns True if the message associated with the Exception_Occurence for
   --  a Socket_Error is a timeout.

   function Is_Timeout (E : Exception_Occurrence) return Boolean;
   --  As above but without Socket parameter

   function Is_Peer_Closed
     (Socket : Socket_Type;
      E      : Exception_Occurrence) return Boolean;
   --  Returns True if the message associated with the Exception_Occurence for
   --  a Socket_Error is a "socket closed by peer".

   --------------------
   -- Socket FD sets --
   --------------------

   type FD_Set (Size : Natural) is abstract tagged private;
   --  Abstract type for waiting of network events on group of sockets FD

   type FD_Set_Access is access all FD_Set'Class;

   function To_FD_Set
     (Socket : Socket_Type;
      Events : Wait_Event_Set;
      Size   : Positive := 1) return FD_Set'Class;
   --  Create appropriate socket FD set and put Socket fd there

   procedure Add
     (FD_Set : in out FD_Set_Access;
      FD     : FD_Type;
      Event  : Wait_Event_Set);
   --  Add FD to the end of FD_Set

   procedure Free (FD_Set : in out FD_Set_Access) with Inline;
   --  Deallocate the socket FD set

   procedure Add
     (FD_Set : in out Net.FD_Set;
      FD     : FD_Type;
      Event  : Wait_Event_Set) is abstract;
   --  Add FD to the end of FD_Set

   procedure Replace
     (FD_Set : in out Net.FD_Set;
      Index  : Positive;
      FD     : FD_Type) is abstract
   with Pre'Class => Index <= Length (FD_Set);
   --  Replaces the socket FD in FD_Set

   procedure Set_Mode
     (FD_Set : in out Net.FD_Set;
      Index  : Positive;
```

(continues on next page)

(continued from previous page)

```
    Mode   : Wait_Event_Set) is abstract
  with Pre'Class => Index <= Length (FD_Set);
  --  Sets the kind of network events to wait for

  procedure Set_Event
    (FD_Set : in out Net.FD_Set;
     Index  : Positive;
     Event  : Wait_Event_Type;
     Value  : Boolean) is abstract
  with Pre'Class => Index <= Length (FD_Set);

  function Copy
    (FD_Set : not null access Net.FD_Set;
     Size   : 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 : Positive) is abstract
  with Pre'Class => Index <= Length (FD_Set);
  --  Removes socket FD from Index position.
  --  Last socket FD in FD_Set is placed at position Index.

  function Length (FD_Set : 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 : Duration;
     Count   : out Natural) is abstract
  with Post'Class => Count <= Length (FD_Set);
  --  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 : Net.FD_Set; Index : in out Positive) is abstract
  with
    Pre'Class  => Index <= Length (FD_Set) + 1,
    Post'Class => Index <= Length (FD_Set) + 1;
  --  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 retreive the kind of network
  --  events for this socket.

  function Status
    (FD_Set : Net.FD_Set;
     Index  : Positive) return Event_Set is abstract
  with Pre'Class => Index <= Length (FD_Set);
  --  Returns events for the socket FD at position Index

  procedure Free (Socket : in out Socket_Type) is null;
  --  Release memory associated with the socket object. This default version
  --  can be overriden to properly release the memory for the derived
```

(continues on next page)

(continued from previous page)

```
   --  implementation. The controlled Finalize routine is in charge of calling
   --  Free. We could not have it in the private part because we could not make
   --  AWS.Net.SSL.Free overriding this way.

   function Localhost (IPv6 : Boolean) return String;
   --  Returns "::1" if IPv6 is true or "127.0.0.1" otherwise

   procedure Set_Host_Alias (Alias, Host : String);
   --  Set alias for host. When Connect call will be to Alias then the real
   --  plain socket connection will be performed to Host. But the servername
   --  information into the SSL socket will be set to Alias.
   --  This routine can be called one or few times from main task before first
   --  call to Connect. Note that the Alias is case sensitive, i.e. if you set
   --  alias www.google.com for localhost and call for www.Google.com you are
   --  going to connect to original address.

private
   -- implementation removed
end AWS.Net;
```

13.25 AWS.Net.Buffered

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2002-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  All routines below are buffered both ways (input and output) for better
--  performances.

package AWS.Net.Buffered is

   ------------
   -- Output --
   ------------

   procedure Put (Socket : Socket_Type'Class; Item : String);
   --  Write Item into Socket's buffer. Send the buffer to the socket if full

   procedure Put_Line (Socket : Socket_Type'Class; Item : String);
   --  Write Item & CRLF into Socket's buffer. Send the buffer to the socket
   --  if full.

   procedure New_Line (Socket : Socket_Type'Class) with Inline;
   --  Write CRLF into Socket's buffer. Send the buffer to the socket if full

   procedure Write
```

(continues on next page)

(continued from previous page)

```
   (Socket : Socket_Type'Class; Item : Stream_Element_Array);
--  Write Item into Socket's buffer. Send the buffer to the socket if full

procedure Flush (Socket : Socket_Type'Class);
--  Send the buffer to the socket

-----------
-- Input --
-----------

Data_Overflow : exception;
--  Raised from Get_Line and Read_Until routines when size of receiving data
--  exceeds the limit defined by Set_Input_Limit. It avoid unlimited dynamic
--  memory allocation inside of Get_Line and Read_Until when client trying
--  to attack the server by the very long lines in request. Moreover it
--  avoid stack overflow on very long data returned from Get_Line and
--  Read_Until.

procedure Set_Input_Limit (Limit : Positive) with Inline;
--  Set the input size limit for Get_Line and Read_Until routines

function Get_Input_Limit return Stream_Element_Offset with Inline;
--  Get the input size limit for Get_Line and Read_Until routines

procedure Read
   (Socket : Socket_Type'Class; Data : out Stream_Element_Array) with Inline;
--  Returns Data array read from the socket

function Read
   (Socket : Socket_Type'Class;
    Max    : Stream_Element_Count := 4096) return Stream_Element_Array
   with Inline;
--  Returns an array of bytes read from the socket

procedure Read
   (Socket : 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 : 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 : Socket_Type'Class) return Character with Inline;
--  Returns a single character read from socket

function Get_Byte
   (Socket : Socket_Type'Class) return Stream_Element with Inline;
--  Returns a single byte read from socket
```

(continues on next page)

(continued from previous page)

```
   function Peek_Char (Socket : Socket_Type'Class) return Character
     with Inline;
   --  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.

   function Pending (Socket : Socket_Type'Class) return Stream_Element_Count;
   --  Returns number of bytes to read in sockets and cache

   procedure Read_Buffer
     (Socket : 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.

   function Read_Until
     (Socket    : Socket_Type'Class;
      Delimiter : Stream_Element_Array;
      Wait      : Boolean := True) return Stream_Element_Array;
   --  Read data on the Socket until the delimiter (including the delimiter).
   --  If Wait is False the routine looking for the delimiter only in the
   --  cache buffer, if delimiter not found in the cache buffer, empty array
   --  is be returned.
   --  If returned data is without delimiter at the end, it means that socket
   --  is closed from peer or socket error occured and rest of data returned.
   --  This routine could loose some data on timeout if does not meet delimiter
   --  longer then Read buffer size.

   function Read_Until
     (Socket    : Socket_Type'Class;
      Delimiter : String;
      Wait      : Boolean := True) return String;
   --  Same as above but returning a standard string

   -------------
   -- Control --
   -------------

   procedure Shutdown (Socket : Socket_Type'Class);
   --  Shutdown and close the socket. Release all memory and resources
   --  associated with it.

private
   -- implementation removed
end AWS.Net.Buffered;
```

13.26 AWS.Net.Log

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2013, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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 : Data_Direction;
      Socket    : Socket_Type'Class;
      Data      : Stream_Element_Array;
```

(continues on next page)

(continued from previous page)

```
      Last      : Stream_Element_Offset);
   --  The callback procedure which is called for each incoming/outgoing data

   type Event_Callback is access procedure
     (Action : Event_Type; Socket : Socket_Type'Class);
   --  The callback procedure which is called for every socket creation,
   --  connect and accept.

   type Error_Callback is access procedure
     (Socket : Socket_Type'Class; Message : String);
   --  The callback procedure which is called for every socket error

   procedure Start
     (Write : Write_Callback;
      Event : Event_Callback := null;
      Error : Error_Callback := null);
   --  Activate the logging

   function Is_Active return Boolean with Inline;
   --  Returns True if Log is activated and False otherwise

   function Is_Write_Active return Boolean with Inline;
   --  Returns True if Write Log is activated and False otherwise

   function Is_Event_Active return Boolean with Inline;
   --  Returns True if Event Log is activated and False otherwise

   procedure Write
     (Direction : Data_Direction;
      Socket    : Socket_Type'Class;
      Data      : Stream_Element_Array;
      Last      : 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 : Event_Type; Socket : Socket_Type'Class);
   --  Call Event callback if activated (i.e. Start routine above has been
   --  called). Otherwise this call does nothing.

   procedure Error (Socket : Socket_Type'Class; Message : 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;
```

13.27 AWS.Net.Log.Callbacks

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 : String;
      Callback : Write_Callback);
   --  Initialize the logging, must be called before using the callbacks below

   procedure Finalize;
   --  Stop logging, close log file

   procedure Text
     (Direction : Data_Direction;
      Socket    : Socket_Type'Class;
      Data      : Stream_Element_Array;
      Last      : 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>
```

(continues on next page)

(continued from previous page)

```
   procedure Binary
     (Direction : Data_Direction;
      Socket    : Socket_Type'Class;
      Data      : Stream_Element_Array;
      Last      : 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   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;
```

13.28 AWS.Net.SSL

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2002-2024, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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 Ada.Calendar;

with System;

with AWS.Containers.String_Vectors;
with AWS.Default;
with AWS.Net.Std;
with SSL.Thin;

private with Ada.Strings.Unbounded;

package AWS.Net.SSL is

   package SV renames AWS.Containers.String_Vectors;

   Socket_Error : exception renames Net.Socket_Error;
```

(continues on next page)

(continued from previous page)

```
   type Socket_Type is new Net.Std.Socket_Type with private;

   type Session_Type is private;
   --  To keep session data over plain socket reconnect

   Null_Session : constant Session_Type;

   Is_Supported : constant Boolean;
   --  True if SSL supported in the current runtime

   type Debug_Output_Procedure is access procedure (Text : String);

   ----------------
   -- Initialize --
   ----------------

   overriding procedure Accept_Socket
     (Socket : Net.Socket_Type'Class; New_Socket : in out Socket_Type);
   --  Accept a connection on a socket

   overriding procedure Connect
     (Socket : in out Socket_Type;
      Host   : String;
      Port   : Positive;
      Wait   : Boolean     := True;
      Family : Family_Type := Family_Unspec);
   --  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 establised 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.

   overriding procedure Socket_Pair (S1, S2 : out Socket_Type);
   --  Create 2 sockets and connect them together

   overriding procedure Shutdown
     (Socket : Socket_Type; How : Shutmode_Type := Shut_Read_Write);
   --  Shutdown the read, write or both side of the socket.
   --  If How is Both, close it. Does not raise Socket_Error if the socket is
   --  not connected or already shutdown.

   --------
   -- IO --
   --------

   overriding procedure Send
     (Socket : Socket_Type;
      Data   : Stream_Element_Array;
      Last   : out Stream_Element_Offset);
```

(continues on next page)

(continued from previous page)

```
   overriding procedure Receive
     (Socket : Socket_Type;
      Data   : out Stream_Element_Array;
      Last   : out Stream_Element_Offset)
     with Inline;

   overriding function Pending
     (Socket : Socket_Type) return Stream_Element_Count;
   --  Returns the number of bytes which are available inside socket
   --  for immediate read.

   --------------------
   -- Initialization --
   --------------------

   type Method is
     (TLS,     TLS_Server,     TLS_Client,      -- Highest available TLS
      TLSv1,   TLSv1_Server,   TLSv1_Client,    -- TLS 1.0
      TLSv1_1, TLSv1_1_Server, TLSv1_1_Client,  -- TLS 1.1
      TLSv1_2, TLSv1_2_Server, TLSv1_2_Client); -- TLS 1.2

   SSLv23        : constant Method := TLS
     with Obsolescent => "use TLS instead";
   SSLv23_Server : constant Method := TLS_Server
     with Obsolescent => "use TLS_Server instead";
   SSLv23_Client : constant Method := TLS_Client
     with Obsolescent => "use TLS_Client instead";
   SSLv3         : constant Method := TLS
     with Obsolescent => "use TLS instead";
   SSLv3_Server  : constant Method := TLS_Server
     with Obsolescent => "use TLS_Server instead";
   SSLv3_Client  : constant Method := TLS_Client
     with Obsolescent => "use TLS_Client instead";

   type Config is private;

   Null_Config : constant Config;

   procedure Initialize
     (Config               : in out SSL.Config;
      Security_Mode        : Method    := TLS;
      Server_Certificate   : String    := "";
      Server_Key           : String    := "";
      Client_Certificate   : String    := "";
      Priorities           : String    := "";
      Ticket_Support       : Boolean   := False;
      Exchange_Certificate : Boolean   := False;
      Check_Certificate    : Boolean   := True;
      Trusted_CA_Filename  : String    := "";
      CRL_Filename         : String    := "";
      Session_Cache_Size   : Natural   := 16#4000#;
      ALPN                 : SV.Vector := SV.Empty_Vector);
```

(continues on next page)

(continued from previous page)

```
   --  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 its certificate to
   --  the server, if False only the server will send its certificate.
   --  ALPN is abbreviation of Application Layer Protocol Negotiation.

   procedure Add_Host_Certificate
     (Config               : SSL.Config;
      Host                 : String;
      Certificate_Filename : String;
      Key_Filename         : String := "");
   --  Support for Server name indication (SNI). Client can ask for different
   --  host names on the same IP address. This routines provide a way to have
   --  different certificates for different server host names.

   procedure Initialize_Default_Config
     (Security_Mode        : Method    := TLS;
      Server_Certificate   : String    := Default.Server_Certificate;
      Server_Key           : String    := Default.Server_Key;
      Client_Certificate   : String    := Default.Client_Certificate;
      Priorities           : String    := "";
      Ticket_Support       : Boolean   := False;
      Exchange_Certificate : Boolean   := False;
      Check_Certificate    : Boolean   := True;
      Trusted_CA_Filename  : String    := Default.Trusted_CA;
      CRL_Filename         : String    := "";
      Session_Cache_Size   : Natural   := 16#4000#;
      ALPN                 : SV.Vector := SV.Empty_Vector);
   --  As above but for the default SSL configuration which will be used
   --  for any socket not setting explicitly an SSL config object. Not that
   --  this routine can only be called once. Subsequent calls are no-op. To
   --  be effective it must be called before any SSL socket is created.

   --  function Get_Default_Config return SSL.Config;
   --  Returns the default configuration object as defined above

   procedure ALPN_Set (Config : SSL.Config; Protocols : SV.Vector);
   --  This function is to be used by both clients and servers, to declare the
   --  supported ALPN protocols (Application Layer Protocol Negotiation), which
   --  are used during negotiation with peer.

   procedure ALPN_Include (Config : SSL.Config; Protocol : String);
   --  Append protocol into ALPN if it was not there

   procedure Release (Config : in out SSL.Config);
   --  Release memory associated with the Config object

   procedure Set_Config
     (Socket : in out Socket_Type; Config : SSL.Config);
```

(continues on next page)

(continued from previous page)

```
   --  Set the SSL configuration object for the secure socket

   function Get_Config (Socket : Socket_Type) return SSL.Config with Inline;
   --  Get the SSL configuration object of the secure socket

   function Secure_Client
     (Socket : Net.Socket_Type'Class;
      Config : SSL.Config := Null_Config;
      Host   : String     := "") 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.
   --  Host parameter is hostname to connect and used to send over SSL
   --  connection to server if defined.

   function Secure_Server
     (Socket : Net.Socket_Type'Class;
      Config : 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.

   function ALPN_Get (Socket : Socket_Type) return String;
   --  This function allows you to get the negotiated protocol name. The
   --  returned protocol should be treated as opaque, constant value and only
   --  valid during the session life. The selected protocol is the first
   --  supported by the list sent by the client.
   --  Empty if no supported protocol found.

   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.

   function Version (Build_Info : Boolean := False) return String;
   --  Returns version information

   procedure Clear_Session_Cache (Config : SSL.Config := Null_Config);
   --  Remove all sessions from SSL session cache from the SSL context.
   --  Null_Config mean default context.

   procedure Set_Session_Cache_Size
     (Size : Natural; Config : SSL.Config := Null_Config);
   --  Set session cache size in the SSL context.
   --  Null_Config mean default context.

   function Session_Cache_Number
     (Config : SSL.Config := Null_Config) return Natural;
   --  Returns number of sessions currently in the cache.
   --  Null_Config mean default context.
```

(continues on next page)

(continued from previous page)

```
   overriding function Cipher_Description (Socket : Socket_Type) return String;

   procedure Ciphers (Cipher : not null access procedure (Name : String));
   --  Calls callback Cipher for all available ciphers

   procedure Generate_DH;
   --  Regenerates Diffie-Hellman parameters.
   --  The call could take a quite long time.
   --  Diffie-Hellman parameters should be discarded and regenerated once a
   --  week or once a month. Depends on the security requirements.
   --  (gnutls/src/serv.c).

   procedure Generate_RSA;
   --  Regenerates RSA parameters.
   --  The call could take some time.
   --  RSA parameters should be discarded and regenerated once a day, once
   --  every 500 transactions etc. Depends on the security requirements
   --  (gnutls/src/serv.c).

   procedure Abort_DH_Generation with Inline;
   --  DH generation could be for a few minutes. If it is really necessary to
   --  terminate process faster, this call should be used.
   --  GNUTLS generates DH parameters much faster than OpenSSL, at least in
   --  Linux x86_64 and does not support DH generation abort at least in
   --  version 3.2.12.

   procedure Start_Parameters_Generation
     (DH : Boolean; Logging : access procedure (Text : String) := null)
     with Inline;
   --  Start SSL parameters regeneration in background.
   --  DH is False mean only RSA parameters generated.
   --  DH is True mean RSA and DH both parameters generated.

   function Generated_Time_DH return Ada.Calendar.Time with Inline;
   --  Returns date and time when the DH parameters was generated last time.
   --  Need to decide when new regeneration would start.

   function Generated_Time_RSA return Ada.Calendar.Time with Inline;
   --  Returns date and time when the RSA parameters was generated last time.
   --  Need to decide when new regeneration would start.

   procedure Set_Debug
     (Level : Natural; Output : Debug_Output_Procedure := null);
   --  Set debug information printed level and output callback.
   --  Null output callback mean output to Ada.Text_IO.Current_Error.

   function Session_Id_Image (Session : Session_Type) return String;
   --  Returns base64 encoded session id. Could be used to recognize resumed
   --  session when it has the same Id.

   function Session_Id_Image (Socket : Socket_Type) return String;
```

(continues on next page)

(continued from previous page)

```
   --  Returns base64 encoded session id of the socket

   function Session_Data (Socket : Socket_Type) return Session_Type;
   --  For the client side SSL socket returns session data to be used to
   --  resume session after socket disconnected.

   procedure Free (Session : in out Session_Type);
   --  Free session data

   procedure Set_Session_Data
     (Socket : in out Socket_Type; Data : Session_Type);
   --  For the client side SSL socket try to resume session from data taken
   --  from previosly connected socket by Session_Data routine.

   function Session_Reused (Socket : Socket_Type) return Boolean;
   --  Returns True in case session was successfully reused after
   --  Set_Session_Data and handshake.

   type Private_Key is private;

   Null_Private_Key : constant Private_Key;

   type Hash_Method is (MD5, SHA1, SHA224, SHA256, SHA384, SHA512);

   function Load (Filename : String) return Private_Key;

   procedure Free (Key : in out Private_Key) with Inline;

   function Signature
     (Data : String;
      Key  : Private_Key;
      Hash : Hash_Method) return Stream_Element_Array with Inline;

   function Signature
     (Data : Stream_Element_Array;
      Key  : Private_Key;
      Hash : Hash_Method) return Stream_Element_Array with Inline;

   overriding function Is_Secure (Socket : Socket_Type) return Boolean;

   procedure Show_Session_Statistic
     (Config : SSL.Config;
      Report : not null access procedure (Line : String));
   --  Show session statistic for Config. Report will be called for each line
   --  of the statistic.

   function Get_Default_Client_Config return SSL.Config;

   function Get_Default_Server_Config return SSL.Config;

private
   -- implementation removed
```

(continues on next page)

(continued from previous page)

```
end AWS.Net.SSL;
```

13.29 AWS.Net.SSL.Certificate

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2024, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with Ada.Calendar;

private with Ada.Containers.Indefinite_Holders;
private with AWS.Utils;

package AWS.Net.SSL.Certificate is

   type Object is private;

   Undefined : constant Object;

   function Get (Socket : Socket_Type) return Object;
   --  Returns the certificate used by the SSL

   function Common_Name (Certificate : Object) return String with Inline;
   --  Returns the certificate's common name

   function Subject (Certificate : Object) return String with Inline;
   --  Returns the certificate's subject
```

(continues on next page)

(continued from previous page)

```
   function Issuer (Certificate : Object) return String with Inline;
   --  Returns the certificate's issuer

   function Serial_Number (Certificate : Object) return String with Inline;
   --  Returns the certificate's serial number

   function DER (Certificate : Object) return Stream_Element_Array with Inline;
   --  Returns all certificate's data in DER format

   overriding function "=" (Left, Right : Object) return Boolean with Inline;
   --  Compare 2 certificates

   function Load (Filename : String) return Object;
   --  Load certificate from file in PEM format

   function Activation_Time (Certificate : Object) return Calendar.Time
     with Inline;
   --  Certificate validity starting date

   function Expiration_Time (Certificate : Object) return Calendar.Time
     with Inline;
   --  Certificate validity ending date

   function Verified (Certificate : Object) return Boolean with Inline;
   --  Returns True if the certificate has already been verified, this is
   --  mostly interresting when used from the Verify_Callback below. If this
   --  routine returns True it means that the certificate has already been
   --  properly checked. If checked the certificate can be trusted and the
   --  Verify_Callback should return True also. If it is False it is up to
   --  the application to check the certificate into the Verify_Callback and
   --  returns the appropriate status.

   function Status (Certificate : Object) return Long_Integer with Inline;
   --  Returns the status for the certificate. This is to be used inside the
   --  verify callback to know why the certificate has been rejected.

   function Status_Message (Certificate : Object) return String;
   --  Returns the error message for the current certificate status (as
   --  returned by Status above).

   --
   --  Client verification support
   --

   type Verify_Callback is
     access function (Cert : SSL.Certificate.Object) return Boolean;
   --  Client certificate verification callback, must return True if Cert can
   --  be accepted or False otherwise. Such callback should generally return
   --  the value returned by Verified above.

   procedure Set_Verify_Callback
     (Config : in out SSL.Config; Callback : Verify_Callback);
```

(continues on next page)

(continued from previous page)

```
   --  Register the callback to use to verify client's certificates

   type Password_Callback is
     access function (Certificate_Filename : String) return String;
   --  Callback to get password for signed server's keys. An empty string
   --  must be returned if the password is unknown or the certificate isn't
   --  signed.

   procedure Set_Password_Callback (Callback : Password_Callback);
   --  Set the password callback

   function Get_Password (Certificate_Filename : String) return String;
   --  Request a password for the giver certificate. The default
   --  implementation just returns an empty string.

private
   -- implementation removed
end AWS.Net.SSL.Certificate;
```

13.30 AWS.Net.WebSocket

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2012-2022, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  This implements the WebSocket protocol as defined in RFC-6455

with Ada.Strings.Unbounded;
with AWS.Status;

private with Ada.Calendar;
private with Ada.Containers.Doubly_Linked_Lists;
private with AWS.Client;
private with Interfaces;

with GNATCOLL.Refcount;

package AWS.Net.WebSocket is

   use Ada.Strings.Unbounded;

   type Object is new Net.Socket_Type with private;

   type Object_Class is private;
```

(continues on next page)

(continued from previous page)

```
   No_Object : constant Object_Class;

   type Kind_Type
     is (Unknown, Connection_Open, Text, Binary, Ping, Pong, Connection_Close);
   --  Data Frame Kind

   type Error_Type is
     (Normal_Closure,
      Going_Away,
      Protocol_Error,
      Unsupported_Data,
      No_Status_Received,
      Abnormal_Closure,
      Invalid_Frame_Payload_Data,
      Policy_Violation,
      Message_Too_Big,
      Mandatory_Extension,
      Internal_Server_Error,
      TLS_Handshake,
      Cannot_Resolve_Error,
      User_01,              -- User's defined error code
      User_02,
      User_03,
      User_04,
      User_05);

   --
   --  The following three methods are the one to override or redefine. In fact
   --  the default Send implementation should be ok for most usages.
   --

   function Create
     (Socket  : Socket_Access;
      Request : AWS.Status.Data) return Object'Class
   with Pre => Socket /= null;
   --  Create a new instance of the WebSocket, this is used by AWS internal
   --  server to create a default WebSocket if no other constructor are
   --  provided. It is also needed when deriving from WebSocket.
   --
   --  This function must be registered via AWS.Net.WebSocket.Registry.Register

   procedure On_Message (Socket : in out Object; Message : String) is null;
   --  Default implementation does nothing, it needs to be overriden by the
   --  end-user. This is the callback that will get activated for every server
   --  incoming data. It is also important to keep in mind that the thread
   --  handling this WebSocket won't be released until the procedure returns.
   --  So the code inside this routine should be small and most importantly not
   --  wait for an event to occur otherwise other requests won't be served.

   procedure On_Message (Socket : in out Object; Message : Unbounded_String);
   --  Same a above but takes an Unbounded_String. This is supposed to be
   --  overriden when handling large messages otherwise a stack-overflow could
```

(continues on next page)

(continued from previous page)

```
   --  be raised. The default implementation of this procedure to to call the
   --  On_Message above with a string.
   --
   --  So either this version is overriden to handle the incoming messages or
   --  the one above if the messages are known to be small.

   procedure On_Open (Socket : in out Object; Message : String) is null;
   --  As above but activated when a WebSocket is opened

   procedure On_Close (Socket : in out Object; Message : String) is null;
   --  As above but activated when a WebSocket is closed. This may be
   --  called from a protected object, so should not do any
   --  potentially blocking operation.

   procedure On_Error (Socket : in out Object; Message : String) is null;
   --  As above but activated when a WebSocket error is detected

   procedure Send
     (Socket    : in out Object;
      Message   : String;
      Is_Binary : Boolean := False);
   --  This default implementation just send a message to the client. The
   --  message is sent in a single chunk (not fragmented).

   procedure Send
     (Socket    : in out Object;
      Message   : Unbounded_String;
      Is_Binary : Boolean := False);
   --  Same as above but can be used for large messages. The message is
   --  possibly sent fragmented.

   procedure Send
     (Socket    : in out Object;
      Message   : Stream_Element_Array;
      Is_Binary : Boolean := True);
   --  As above but default is a binary message

   procedure Close
     (Socket  : in out Object;
      Message : String;
      Error   : Error_Type := Normal_Closure);
   --  Send a close frame to the WebSocket

   --
   --  Client side
   --

   procedure Connect
     (Socket : in out Object'Class;
      URI    : String);
   --  Connect to a remote server using websockets.
   --  Socket can then be used to Send messages to the server. It will
```

(continues on next page)

(continued from previous page)

```
   --  also receive data from the server, via the On_Message, when you call
   --  Poll

   function Poll
     (Socket  : in out Object'Class;
      Timeout : Duration) return Boolean;
   --  Wait for up to Timeout seconds for some message.
   --
   --  In the websockets protocol, a message can be split (by the server)
   --  onto several frames, so that for instance the server doesn't have to
   --  store the whole message in its memory.
   --  The size of those frames, however, is not limited, and they will
   --  therefore possibly be split into several chunks by the transport
   --  layer.
   --
   --  These function waits until it either receives a close or an error, or
   --  the beginning of a message frame. In the latter case, the function
   --  will then block until it has receives all chunks of that frame, which
   --  might take longer than Timeout.
   --
   --  The function will return early if it doesn't receive the beginning
   --  of a frame within Timeout seconds.
   --
   --  When a full frame has been received, it will be sent to the
   --  Socket.On_Message primitive operation. Remember this might not be the
   --  whole message however, and you should check Socket.End_Of_Message to
   --  check.
   --
   --  Return True if a message was processed, False if nothing happened during
   --  Timeout.

   --
   --  Simple accessors to WebSocket state
   --

   function Kind (Socket : Object) return Kind_Type;
   --  Returns the message kind of the current read data

   function Protocol_Version (Socket : Object) return Natural;
   --  Returns the version of the protocol for this WebSocket

   function URI (Socket : Object) return String;
   --  Returns the URI for the WebSocket

   function Origin (Socket : Object) return String;
   --  Returns the Origin of the WebSocket. That is the value of the Origin
   --  header of the client which has opened the socket.

   function Request (Socket : Object) return AWS.Status.Data;
   --  Returns Request of the WebSocket. That is the HTTP-request
   --  of the client which has opened the socket.
```

(continues on next page)

(continued from previous page)

```
   function Error (Socket : Object) return Error_Type;
   --  Returns the current error type

   function End_Of_Message (Socket : Object) return Boolean;
   --  Returns True if we have read a whole message

   --
   --  Socket's methods that must be overriden
   --

   overriding procedure Shutdown
     (Socket : Object;
      How    : Shutmode_Type := Shut_Read_Write);
   --  Shutdown the socket

   overriding function Get_FD (Socket : Object) return FD_Type;
   --  Returns the file descriptor associated with the socket

   overriding function Peer_Addr (Socket : Object) return String;
   --  Returns the peer name/address

   overriding function Peer_Port (Socket : Object) return Positive;
   --  Returns the port of the peer socket

   overriding function Get_Addr (Socket : Object) return String;
   --  Returns the name/address of the socket

   overriding function Get_Port (Socket : Object) return Positive;
   --  Returns the port of the socket

   overriding function Errno (Socket : Object) return Integer;
   --  Returns and clears error state in socket

   overriding function Get_Send_Buffer_Size (Socket : Object) return Natural;
   --  Returns the internal socket send buffer size.
   --  Do not confuse with buffers for the AWS.Net.Buffered operations.

   overriding function Get_Receive_Buffer_Size
     (Socket : Object) return Natural;
   --  Returns the internal socket receive buffer size.
   --  Do not confuse with buffers for the AWS.Net.Buffered operations.

   --
   --  Socket reference
   --

   type UID is range 0 .. 2**31;

   No_UID : constant UID;
   --  Not an UID, this is a WebSocket not yet initialized

   function Get_UID (Socket : Object) return UID;
```

(continues on next page)

(continued from previous page)

```
   --  Returns a unique id for the given socket. The uniqueness for this socket
   --  is guaranteed during the lifetime of the application.

   overriding function Is_Secure (Socket : Object) return Boolean;

private
   -- implementation removed
end AWS.Net.WebSocket;
```

13.31 AWS.Net.WebSocket.Registry

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2012-2019, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  This package is used to build and register the active WebSockets. Some
--  services to send or broadcast messages are also provided.

with AWS.Status;

private with GNAT.Regexp;

package AWS.Net.WebSocket.Registry is

   type Factory is not null access function
     (Socket  : Socket_Access;
      Request : AWS.Status.Data) return Object'Class;

   --  Creating and Registering WebSockets

   function Constructor (URI : String) return Registry.Factory
     with Pre => URI'Length > 0;
   --  Get the WebObject's constructor for a specific URI

   procedure Register (URI : String; Factory : Registry.Factory)
```

(continues on next page)

(continued from previous page)

```
   with Pre => URI'Length > 0;
--  Register a WebObject's constructor for a specific URI

procedure Register_Pattern
   (Pattern : String;
    Factory : Registry.Factory)
   with Pre => Pattern'Length > 0;
--  Register a WebObject's constructor for a specific URI and pattern

--  Sending messages

type Recipient is private;

No_Recipient : constant Recipient;

function Create (URI : String; Origin : String := "") return Recipient
   with Pre  => URI'Length >  0,
        Post => Create'Result /= No_Recipient;
--  A recipient with only an URI is called a broadcast as it designate all
--  registered WebSocket for this specific URI. If Origin is specified then
--  it designates a single client.
--
--  Note that both URI and Origin can be regular expressions.

function Create (Id : UID) return Recipient
   with Pre  => Id /= No_UID,
        Post => Create'Result /= No_Recipient;
--  A recipient for a specific WebSocket

type Action_Kind is (None, Close);

procedure Send
   (To           : Recipient;
    Message      : String;
    Except_Peer  : String := "";
    Timeout      : Duration := Forever;
    Asynchronous : Boolean := False;
    Error        : access procedure (Socket : Object'Class;
                                     Action : out Action_Kind) := null)
   with Pre => To /= No_Recipient
               and then (if Asynchronous then Error = null);
--  Send a message to the WebSocket designated by Origin and URI. Do not
--  send this message to the peer whose address is given by Except_Peer.
--  Except_Peer must be the address as reported by AWS.Net.Peer_Addr. It is
--  often needed to send a message to all registered sockets except the one
--  which has sent the message triggering a response.

procedure Send
   (To           : Recipient;
    Message      : Unbounded_String;
    Except_Peer  : String := "";
    Timeout      : Duration := Forever;
```

(continues on next page)

(continued from previous page)

```
    Asynchronous : Boolean := False;
    Error        : access procedure (Socket : Object'Class;
                                     Action : out Action_Kind) := null)
   with Pre => To /= No_Recipient
               and then (if Asynchronous then Error = null);
--  As above but with an Unbounded_String

procedure Send
   (To           : Recipient;
    Message      : String;
    Request      : AWS.Status.Data;
    Timeout      : Duration := Forever;
    Asynchronous : Boolean := False;
    Error        : access procedure (Socket : Object'Class;
                                     Action : out Action_Kind) := null)
   with Pre => To /= No_Recipient
               and then (if Asynchronous then Error = null);
--  As above but filter out the client having set the given request

procedure Send
   (To           : Recipient;
    Message      : Unbounded_String;
    Request      : AWS.Status.Data;
    Timeout      : Duration := Forever;
    Asynchronous : Boolean := False;
    Error        : access procedure (Socket : Object'Class;
                                     Action : out Action_Kind) := null)
   with Pre => To /= No_Recipient
               and then (if Asynchronous then Error = null);
--  As above but with an Unbounded_String

procedure Close
   (To          : Recipient;
    Message     : String;
    Except_Peer : String := "";
    Timeout     : Duration := Forever;
    Error       : Error_Type := Normal_Closure)
   with Pre => To /= No_Recipient;
--  Close connections

--  Targeting a single WebSocket, these routines are equivalent to the
--  Net.WebSocket ones but are thread-safe. That is, they can be mixed
--  with other WebSocket activity to and from the clients.

procedure Send
   (Socket       : in out Object'Class;
    Message      : String;
    Is_Binary    : Boolean := False;
    Timeout      : Duration := Forever;
    Asynchronous : Boolean := False);
--  This default implementation just send a message to the client. The
--  message is sent in a single chunk (not fragmented).
```

(continues on next page)

(continued from previous page)

```
   procedure Send
     (Socket       : in out Object'Class;
      Message      : Unbounded_String;
      Is_Binary    : Boolean := False;
      Timeout      : Duration := Forever;
      Asynchronous : Boolean := False);
   --  Same as above but can be used for large messages. The message is
   --  possibly sent fragmented.

   procedure Send
     (Socket       : in out Object'Class;
      Message      : Stream_Element_Array;
      Is_Binary    : Boolean := True;
      Timeout      : Duration := Forever;
      Asynchronous : Boolean := False);
   --  As above but for a Stream_Element_Array

   procedure Close
     (Socket  : in out Object'Class;
      Message : String;
      Timeout : Duration := Forever;
      Error   : Error_Type := Normal_Closure);

   function Is_Registered (Id : UID) return Boolean;
   --  Returns True if the WebSocket Id is registered and False otherwise

private
   -- implementation removed
end AWS.Net.WebSocket.Registry;
```

13.32 AWS.Net.WebSocket.Registry.Control

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2012, AdaCore                          --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 start/stop the WebSockets services

package AWS.Net.WebSocket.Registry.Control is

   procedure Start;
   --  Start the WebSockets servers

   procedure Shutdown;
   --  Shutdown the WebSockets servers

end AWS.Net.WebSocket.Registry.Control;
```

13.33 AWS.Parameters

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2017, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with Ada.Strings.Unbounded; use Ada.Strings.Unbounded;
with AWS.Containers.Tables;
with AWS.Resources.Streams.Memory;

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 : List; Limit : Positive := Positive'Last) return String;
   --  Returns the list of parameters in the URI format. This can be added
   --  after the resource to form the complete URI. The format is:
   --  "?name1=value1&name2=value2..."
   --  If there is no parameter in the list, the empty string is returned.
   --  Limit is maximum size of the output line, parameters name=value will be
   --  returned unbroken in case of limit applied.

   procedure Add
```

(continues on next page)

(continued from previous page)

```
      (Parameter_List : in out List; Name, Value : String; Decode : Boolean);

   procedure Add
     (Parameter_List : in out List;
      Name, Value    : Unbounded_String;
      Decode         : Boolean);
   --  URL decode and add Name=Value pair into parameters

   procedure Add (Parameter_List : in out List; Parameters : String);
   --  Set parameters for the current request. The Parameters string has the
   --  form "name1=value1&name2=value2...". The paramaters are added to the
   --  list. The parameters can start with a '?' (standard Web character
   --  separator) which is just ignored.

   procedure Add
     (Parameter_List : in out List;
      Parameters     : in out Resources.Streams.Memory.Stream_Type'Class);
   --  Same as above, but use different parameters source. Used to reduce
   --  stack usage on big POST requests. This is the routine used by AWS for
   --  parsing the POST parameters. This routine also control the maximum
   --  number of parameter parsed as set by the corresponding configuration
   --  option.

   procedure Update
     (Parameter_List : in out List; Name, Value : String; Decode : Boolean);

   procedure Update
     (Parameter_List : in out List;
      Name, Value    : Unbounded_String;
      Decode         : Boolean);

   Too_Long_Parameter : exception;
   --  Raised if the Add routine detects a too long parameter line when reading
   --  parameters from Memory_Stream.

   Too_Many_Parameters : exception;
   --  Raised when the maximum number of parameters has been reached

   --  See AWS.Containers.Tables for inherited routines

private
   -- implementation removed
end AWS.Parameters;
```

13.34 AWS.POP

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;
```

(continues on next page)

(continued from previous page)

```
   type Authenticate_Mode is (Clear_Text, APOP);

   function Initialize
     (Server_Name  : String;
      User         : String;
      Password     : String;
      Authenticate : Authenticate_Mode := Clear_Text;
      Port         : 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 : POP.Mailbox);
   --  Close mailbox

   function User_Name (Mailbox : POP.Mailbox) return String;
   --  Returns User's name for this mailbox

   function Message_Count (Mailbox : POP.Mailbox) return Natural;
   --  Returns the number of messages in the user's mailbox

   function Size (Mailbox : POP.Mailbox) return Natural;
   --  Returns the total size in bytes of the user's mailbox

   -------------
   -- Message --
   -------------

   type Message is tagged private;

   function Get
     (Mailbox : POP.Mailbox;
      N       : Positive;
      Remove  : Boolean     := False) return Message;
   --  Retrieve Nth message from the mailbox, let the message on the mailbox
   --  if Remove is False.

   procedure Delete
     (Mailbox : POP.Mailbox;
      N       : Positive);
   --  Detele message number N from the mailbox

   function Get_Header
     (Mailbox : POP.Mailbox;
      N       : 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 : POP.Message;
         Index   : Positive;
```

(continues on next page)

(continued from previous page)

```
      Quit    : in out Boolean);
procedure For_Every_Message
  (Mailbox : POP.Mailbox;
   Remove  : 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 : POP.Message;
      Index   : Positive;
      Quit    : in out Boolean);
procedure For_Every_Message_Header (Mailbox : 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 : POP.Message) return Natural;
--  Returns the message size in bytes

function Content (Message : POP.Message) return Unbounded_String;
--  Returns message's content as an Unbounded_String. Each line are
--  separated by CR+LF characters.

function From (Message : POP.Message) return String;
--  Returns From header value

function To (Message : 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 : 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 : POP.Message; N : Natural := 0) return String;
--  Retruns 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 : 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 : POP.Message) return String;
--  Returns Subject header value

function Date (Message : POP.Message) return String;
--  Returns Date header value

function Header
  (Message : POP.Message;
```

(continues on next page)

(continued from previous page)

```
      Header  : 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 : POP.Message) return Natural;
   --  Returns the number of Attachments into Message

   function Get
     (Message : POP.Message'Class;
      Index   : Positive) return Attachment;
   --  Returns the Nth Attachment for Message, Raises Constraint_Error if
   --  there is not such attachment.

   generic
      with procedure Action
        (Attachment : POP.Attachment;
         Index      : Positive;
         Quit       : in out Boolean);
   procedure For_Every_Attachment (Message : 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 : 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 : 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 : 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 : POP.Attachment) return Boolean;
   --  Returns True if Attachment is a file

   procedure Write (Attachment : POP.Attachment; Directory : String);
   --  Writes Attachment's file content into Directory. This must only be used
   --  for a file attachment.

private
   -- implementation removed
```

(continues on next page)

(continued from previous page)

```
end AWS.POP;
```

13.35 AWS.Resources

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2002-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 AWS.Utils;

private 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.

   function "or" (I1, I2 : File_Instance) return File_Instance;
   --  Returns the union of I1 and I2
```

(continues on next page)

(continued from previous page)

```
   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 : String;
      Form : 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 : String;
      Form : 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       : 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
```

(continues on next page)

(continued from previous page)

```
     (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 EOL) or ASCII.CR+ASCII.LF (DOS style EOL).

   function End_Of_File (Resource : File_Type) return Boolean;
   --  Returns true if there is no more data to read

   function LF_Terminated (Resource : 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 : 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 : 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 : 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 : String) return Utils.File_Size_Type;
   --  Returns Filename's size in bytes. Checks first for in memory file
   --  then for disk file.

   function File_Timestamp (Name : 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;
```

13.36 AWS.Resources.Embedded

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2002-2013, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with AWS.Resources.Streams.Memory;

package AWS.Resources.Embedded is

   use Ada;

   Resource_Error : exception renames Resources.Resource_Error;

   subtype Buffer_Access is Streams.Memory.Buffer_Access;

   procedure Open
     (File : out File_Type;
      Name : String;
      Form : String    := "";
      GZip : in out Boolean);
   --  Open resource from registered data

   procedure Create
     (File   : out File_Type;
      Buffer : Buffer_Access);
```

(continues on next page)

(continued from previous page)

```
   --  Create the resource directly from memory data

   function Exist (Name : 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 : String) return Boolean with Inline;
   --  Returns True if file named Name has been registered (i.e. it is an
   --  in-memory file).

   function File_Size (Name : String) return Utils.File_Size_Type;

   function File_Timestamp (Name : String) return Ada.Calendar.Time;

   procedure Register
     (Name      : String;
      Content   : Buffer_Access;
      File_Time : 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.

   function Get_Buffer
     (Name : String; GZip : in out Boolean) return Buffer_Access;
   --  Get registered embedded resource buffer access by Name. Returns null if
   --  not found. GZip "in" value defines what resource version is preferred,
   --  compressed or plain. GZip "out" value defines what resource version was
   --  found, compressed or plain. In the spetial case when Name has .gz"
   --  suffix already, GZip "in" value is ignored, routine looks only for Name
   --  without duplicated additional suffix, and GZip "out" value became False
   --  if resource was found.

end AWS.Resources.Embedded;
```

13.37 AWS.Resources.Files

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2002-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.Utils;

package AWS.Resources.Files is

   Resource_Error : exception renames Resources.Resource_Error;

   procedure Open
     (File : out File_Type;
      Name : String;
      Form : String    := "";
      GZip : in out Boolean);

   procedure Open
     (File : out File_Type;
      Name : String;
      Form : String    := "");

   function Exist (Name : 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.
```

(continues on next page)

(continued from previous page)

```
   function Is_Regular_File (Name : String) return Boolean;

   function File_Size (Name : String) return Utils.File_Size_Type;

   function File_Timestamp (Name : String) return Ada.Calendar.Time;

end AWS.Resources.Files;
```

13.38 AWS.Resources.Streams

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2002-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

package AWS.Resources.Streams is

   type Stream_Type is abstract tagged limited private;

   type Stream_Access is access all Stream_Type'Class;

   function End_Of_File (Resource : 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;

   function Get_Line (Resource : in out Stream_Type'Class) return String;
   --  Returns a line (set of bytes ending with CR and/or LF) read
   --  from Resource.

   procedure Reset (Resource : in out Stream_Type) is abstract;

   procedure Set_Index
```

(continues on next page)

(continued from previous page)

```
     (Resource : in out Stream_Type;
      To       : 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 : 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 : 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   : Stream_Access) with Inline;
   --  Create a resource file from stream

   function Open
     (Name : String;
      Form : String         := "";
      GZip : in out Boolean;
      Once : Boolean        := False) return Stream_Access;
   --  Create stream by name either from embedded resource or from file.
   --  Returns null if neither embedded resource nore file can be found with
   --  such Name. GZip parameter has the same meaning like in
   --  AWS.Resources.Open routine.
   --  If Once is True than remove file on Close the stream.

private
   -- implementation removed
end AWS.Resources.Streams;
```

13.39 AWS.Resources.Streams.Disk

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.

private with Ada.Strings.Unbounded;
private 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 : String;
      Form : String    := "shared=no");

   overriding function End_Of_File (Resource : Stream_Type) return Boolean;

   overriding procedure Read
     (Resource : in out Stream_Type;
      Buffer   : out Stream_Element_Array;
      Last     : out Stream_Element_Offset);
```

(continues on next page)

(continued from previous page)

```
   overriding function Size
     (Resource : Stream_Type) return Stream_Element_Offset;

   overriding function Name (Resource : Stream_Type) return String;

   overriding procedure Reset (Resource : in out Stream_Type);

   overriding procedure Set_Index
     (Resource : in out Stream_Type;
      To       : Stream_Element_Offset);

   overriding procedure Close (Resource : in out Stream_Type);

private
   -- implementation removed
end AWS.Resources.Streams.Disk;
```

13.40 AWS.Resources.Streams.Disk.Once

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;

   overriding 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;
```

13.41 AWS.Resources.Streams.Memory

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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;

private with AWS.Containers.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;
   subtype Buffer_Access is Utils.Stream_Element_Array_Constant_Access;

   procedure Append
     (Resource : in out Stream_Type;
      Buffer   : Stream_Element_Array;
      Trim     : Boolean := False);
   --  Append Buffer into the memory stream
```

(continues on next page)

(continued from previous page)

```
   procedure Append
     (Resource : in out Stream_Type;
      Buffer   : 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   : 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.

   overriding procedure Read
     (Resource : in out Stream_Type;
      Buffer   : out Stream_Element_Array;
      Last     : out Stream_Element_Offset);
   --  Returns a chunck of data in Buffer, Last point to the last element
   --  returned in Buffer.

   overriding function End_Of_File (Resource : Stream_Type) return Boolean;
   --  Returns True if the end of the memory stream has been reached

   procedure Clear (Resource : in out Stream_Type) with Inline;
   --  Delete all data from memory stream

   overriding procedure Reset (Resource : in out Stream_Type);
   --  Reset the streaming data to the first position

   overriding procedure Set_Index
     (Resource : in out Stream_Type;
      To       : Stream_Element_Offset);
   --  Set the position in the stream, next Read will start at the position
   --  whose index is To.

   overriding function Size
     (Resource : Stream_Type) return Stream_Element_Offset;
   --  Returns the number of bytes in the memory stream

   overriding 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;
```

13.42 AWS.Resources.Streams.Memory.ZLib

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2013, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                      --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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;
```

(continues on next page)

(continued from previous page)

```
   procedure Deflate_Initialize
     (Resource     : in out Stream_Type;
      Level        : Compression_Level  := ZL.Default_Compression;
      Strategy     : Strategy_Type      := ZL.Default_Strategy;
      Method       : Compression_Method := ZL.Deflated;
      Window_Bits  : Window_Bits_Type   := ZL.Default_Window_Bits;
      Memory_Level : Memory_Level_Type  := ZL.Default_Memory_Level;
      Header       : Header_Type        := ZL.Default)
     with Inline;
   --  Initialize the compression

   procedure Inflate_Initialize
     (Resource    : in out Stream_Type;
      Window_Bits : Window_Bits_Type := ZL.Default_Window_Bits;
      Header      : Header_Type      := ZL.Default)
     with Inline;
   --  Initialize the decompression

   overriding procedure Append
     (Resource : in out Stream_Type;
      Buffer   : Stream_Element_Array;
      Trim     : Boolean := False);
   --  Compress/decompress and Append Buffer into the memory stream

   overriding procedure Read
     (Resource : in out Stream_Type;
      Buffer   : out Stream_Element_Array;
      Last     : out Stream_Element_Offset);
   --  Returns a chunck of data in Buffer, Last point to the last element
   --  returned in Buffer.

   overriding function Size
     (Resource : Stream_Type) return Stream_Element_Offset;
   --  Returns the number of bytes in the memory stream

   overriding 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;
```

13.43 AWS.Resources.Streams.Pipe

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2007-2016, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 a pipe.

with GNAT.OS_Lib;

private with Ada.Strings.Unbounded;
private with GNAT.Expect;

package AWS.Resources.Streams.Pipe is

   use GNAT;

   type Stream_Type is new Streams.Stream_Type with private;

   type On_Error_Callback is
     access procedure (Status : Integer; Error : String);

   procedure Open
     (Pipe     : out Stream_Type;
      Command  : String;
      Args     : OS_Lib.Argument_List;
      Timeout  : Integer := 10_000;
```

(continues on next page)

(continued from previous page)

```
      On_Error : On_Error_Callback := null);
   --  Open the pipe and connect it to the given command's output. Args are
   --  passed to the command. Timeout is given in milliseconds and corresponds
   --  to the time waiting for output data before timeout. This timeout must be
   --  adjusted to be compatible to the output activity of the Command process.

   overriding function End_Of_File (Resource : Stream_Type) return Boolean;

   overriding procedure Read
     (Resource : in out Stream_Type;
      Buffer   : out Stream_Element_Array;
      Last     : out Stream_Element_Offset);

   overriding procedure Close (Resource : in out Stream_Type);

   overriding procedure Reset (Resource : in out Stream_Type) is null;
   --  Does nothing as not supported on pipe streams

   overriding procedure Set_Index
     (Resource : in out Stream_Type;
      To       : Stream_Element_Offset) is null;
   --  Does nothing as not supported on pipe streams

private
   -- implementation removed
end AWS.Resources.Streams.Pipe;
```

13.44 AWS.Response

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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.Calendar;
with Ada.Streams;
with Ada.Strings.Unbounded;

with AWS.Headers;
with AWS.Messages;
with AWS.MIME;
with AWS.Net;
with AWS.Resources.Streams;
with AWS.Status;

private with AWS.Utils;
private with Ada.Finalization;
private with Ada.Unchecked_Deallocation;
```

(continues on next page)

(continued from previous page)

```
package AWS.Response is

   use Ada;
   use Ada.Streams;
   use Ada.Strings.Unbounded;

   use type AWS.Messages.Status_Code;

   type Data is private;
   --  Note that this type use a reference counter which is not thread safe

   type Callback is access function (Request : Status.Data) return Data;
   --  This is the Web Server Callback procedure. A client must declare and
   --  pass such procedure to the HTTP server.

   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
      WebSocket,       -- Protocol switched to WebSocket
      No_Data);        -- No data, this is not a response

   type Authentication_Mode is (Unknown, 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.
   --  Unknown, means that an unsupported mode has been found.
   --  Note the order here should not be changed as it is used in AWS.Client.

   subtype Content_Length_Type
     is Stream_Element_Offset range -1 .. Stream_Element_Offset'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 --
```

(continues on next page)

(continued from previous page)

```
   ----------------------

   function Build
     (Content_Type  : String;
      Message_Body  : String;
      Status_Code   : Messages.Status_Code      := Messages.S200;
      Cache_Control : Messages.Cache_Option     := Messages.Unspecified;
      Encoding      : Messages.Content_Encoding := Messages.Identity)
      return Data
   with Post => not Is_Empty (Build'Result)
                and then Response.Status_Code (Build'Result) = Status_Code;

   function Build
     (Content_Type    : String;
      UString_Message : Unbounded_String;
      Status_Code     : Messages.Status_Code      := Messages.S200;
      Cache_Control   : Messages.Cache_Option     := Messages.Unspecified;
      Encoding        : Messages.Content_Encoding := Messages.Identity)
      return Data
   with Post => not Is_Empty (Build'Result)
                and then Response.Status_Code (Build'Result) = Status_Code;
   --  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  : String;
      Message_Body  : Stream_Element_Array;
      Status_Code   : Messages.Status_Code         := Messages.S200;
      Cache_Control : Messages.Cache_Option        := Messages.Unspecified;
      Encoding      : Messages.Content_Encoding    := Messages.Identity)
      return Data
   with Post => not Is_Empty (Build'Result)
                and then Response.Status_Code (Build'Result) = Status_Code;
   --  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.
```

(continues on next page)

(continued from previous page)

```
   function File
     (Content_Type  : String;
      Filename      : String;
      Status_Code   : Messages.Status_Code      := Messages.S200;
      Cache_Control : Messages.Cache_Option     := Messages.Unspecified;
      Encoding      : Messages.Content_Encoding := Messages.Identity;
      Once          : Boolean                   := False;
      Disposition   : Disposition_Mode          := None;
      User_Filename : String                    := "")
      return Data
   with Post => not Is_Empty (File'Result)
                and then Response.Status_Code (File'Result) = Status_Code
                and then (if Once
                          then Mode (File'Result) = File_Once
                          else Mode (File'Result) = File);
   --  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  : String;
      Handle        : not null access Resources.Streams.Stream_Type'Class;
      Status_Code   : Messages.Status_Code      := Messages.S200;
      Cache_Control : Messages.Cache_Option     := Messages.No_Cache;
      Encoding      : Messages.Content_Encoding := Messages.Identity;
      Server_Close  : Boolean                   := True;
      Disposition   : Disposition_Mode          := None;
      User_Filename : String                    := "")
      return Data
   with Post => not Is_Empty (Stream'Result)
                and then Response.Status_Code (Stream'Result) = Status_Code;
   --  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
   --  responsability 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. Encoding mean additional encoding would be applied on top
   --  of given Handler stream.

   -----------------------------
   -- Redirection Constructors --
   -----------------------------

   function URL
     (Location      : String;
```

(continues on next page)

(continued from previous page)

```
   Cache_Control : Messages.Cache_Option := Messages.Unspecified)
   return Data
with Post => not Is_Empty (URL'Result)
             and then Status_Code (URL'Result) = Messages.S302
             and then Mode (URL'Result) = Header;
--  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      : String;
   Message       : String                := Default_Moved_Message;
   Content_Type  : String                := AWS.MIME.Text_HTML;
   Cache_Control : Messages.Cache_Option := Messages.Unspecified)
   return Data
with Post => not Is_Empty (Moved'Result)
             and then Status_Code (Moved'Result) = Messages.S301;
--  This send back a moved message (Messages.S301) with the specified
--  message body and content type.
--  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  : Messages.Status_Code;
   Message_Body : String := "";
   Content_Type : String := MIME.Text_HTML) return Data
with Post =>
    not Is_Empty (Acknowledge'Result)
    and then Response.Status_Code (Acknowledge'Result) = Status_Code
    and then (if Message_Body = ""
              then Mode (Acknowledge'Result) = Header);
--  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   : String;
   Mode    : Authentication_Mode := Basic;
   Stale   : Boolean             := False;
   Message : String              := Default_Authenticate_Message)
   return Data
with Post => not Is_Empty (Authenticate'Result)
             and then Status_Code (Authenticate'Result) = Messages.S401;
--  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.
```

(continues on next page)

(continued from previous page)

```
   function Socket_Taken return Data with
     Post => not Is_Empty (Socket_Taken'Result)
             and then Mode (Socket_Taken'Result) = Socket_Taken;
   --  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 with
     Post => Status_Code (Empty'Result) = Messages.S204
             and then Mode (Empty'Result) = No_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.

   function Continue return Data with
     Post => Status_Code (Continue'Result) = Messages.S100
             and then Mode (Continue'Result) = No_Data;
   --  Returns an empty message (Data_Mode = No_Data and Status_Code is 100).
   --  It is to control the client data upload.
   --
   --  If upload data size is known from Content-Length header and less than
   --  Upload_Size_Limit configuration parameter then the client message body
   --  arrived at once to the dispatcher handler. User can check this by
   --  calling AWS.Status.Is_Body_Uploaded.
   --
   --  If upload data size is unknown or more than Upload_Size_Limit then
   --  Is_Body_Uploaded returns False and user is able to allow or disable the
   --  client data upload.
   --
   --  If user returns Continue response from dispatcher handler, then next
   --  time the dispatcher handler will be called with uploaded body from
   --  client. If user returns some other responses then client body upload
   --  will be terminated and ignored.

   --
   --  API to retrieve response data
   --

   function Is_Continue (D : Data) return Boolean;
   --  Return True if the message (Data_Mode = No_Data and Status_Code is 100)

   ------------
   -- Header --
   ------------

   function Header (D : Data; Name : String; N : Positive) return String
     with Inline;
   --  Return the N-th value for header Name

   function Header (D : Data; Name : String) return String with Inline;
```

(continues on next page)

(continued from previous page)

```
   --  Return all values as a comma-separated string for header Name.
   --  See [RFC 2616 - 4.2] last paragraph.

   function Header (D : Data) return AWS.Headers.List;

   function Has_Header (D : Data; Name : String) return Boolean with Inline;
   --  Returns True if D headers contains Name

   procedure Send_Header
     (Socket    : Net.Socket_Type'Class;
      D         : Data;
      End_Block : Boolean := False) with Inline;
   --  Send all header lines to the socket

   function Status_Code (D : Data) return Messages.Status_Code with Inline;
   --  Returns the status code

   function Content_Length (D : Data) return Content_Length_Type with Inline;
   --  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 : Data) return String with Inline;
   --  Returns the MIME type for the message body

   function Cache_Control (D : Data) return Messages.Cache_Option with Inline;
   --  Returns the cache control specified for the response

   function Cache_Control (D : Data) return Messages.Cache_Data;
   --  As above but returns a structured record of type "Cache_Data (Request)"
   --  representing the cache options.

   function Expires (D : Data) return Calendar.Time with Inline;
   --  Returns the Expires date as a time value

   function Location (D : Data) return String with Inline;
   --  Returns the location for the new page in the case of a moved
   --  message. See Moved constructor above.

   ----------
   -- Data --
   ----------

   function Mode (D : Data) return Data_Mode with Inline;
   --  Returns the data mode, either Header, Message or File

   function Is_Empty (D : Data) return Boolean with Inline;
   --  Returns True if D.Mode is No_Data

   function Message_Body (D : Data) return String with Inline;
   --  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
```

(continues on next page)

(continued from previous page)

```
   --  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 : Data) return Unbounded_String;
   --  Returns message body content as an unbounded_string

   function Message_Body (D : Data) return Stream_Element_Array;
   --  Returns message body as a binary content

   procedure Message_Body
     (D    : Data;
      File : out AWS.Resources.File_Type);
   --  Returns the message body as a stream

   function Filename (D : Data) return String with Inline;
   --  Returns the filename which should be sent back or the filename which
   --  was containing the response for a server response.

   --------------------
   -- Authentication --
   --------------------

   function Realm (D : Data) return String with Inline;
   --  Returns the Realm for the current authentication request

   function Authentication (D : Data) return Authentication_Mode with Inline;
   --  Returns the authentication mode requested by server

   function Authentication_Stale (D : Data) return Boolean with Inline;
   --  Returns the stale parameter for authentication

   ---------------
   -- Resources --
   ---------------

   procedure Create_Resource
     (D    : in out Data;
      File : out AWS.Resources.File_Type;
      GZip : Boolean)
   with Inline;
   --  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 Create_Stream
     (D    : in out Data;
      GZip : Boolean) return AWS.Resources.Streams.Stream_Access;
   --  Creates the stream access for the data to be sent to the client.
```

(continues on next page)

(continued from previous page)

```
   --  The resource should be closed and freed 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 : Data) return Boolean;
   --  Returns True if the resource stream must be closed

   function Keep_Alive (D : Data) return Boolean with Inline;
   --  Returns True if the user want to keep connection alive

   ----------------
   -- WebSockets --
   ----------------

   function WebSocket return Data with
     Post => not Is_Empty (WebSocket'Result)
             and then Status_Code (WebSocket'Result) = Messages.S101
             and then Mode (WebSocket'Result) = WebSocket;
   --  WebSocket handshake from initial WebSocket connection

private
   -- implementation removed
end AWS.Response;
```

13.45 AWS.Response.Set

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2002-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with AWS.Net;

package AWS.Response.Set is

   type Encoding_Direction is (Encode, Decode);
   --  Server side would do gzip or deflate encoding,
   --  Client side would do gzip or deflate decoding.

   ------------
   -- Header --
   ------------

   procedure Add_Header
     (D     : in out Data;
      Name  : String;
      Value : String)
     with Inline;
   --  Add header name/value to the header container.
   --  Should be used inside of server's callback when the user want
   --  to add its own header lines to the response.
```

(continues on next page)

(continued from previous page)

```
   procedure Update_Header
     (D     : in out Data;
      Name  : String;
      Value : String;
      N     : Positive := 1)
     with Inline;
   --  Update N-th header name/value in the header container.
   --  Should be used inside of server's callback when the user want
   --  to add/modify its own header lines to the response.

   procedure Read_Header
     (Socket : Net.Socket_Type'Class; D : in out Data);
   --  Read all header data from the socket

   procedure Parse_Header (D : in out Data);
   --  Fill appropriate data fields in D from header list (for fast access)

   procedure Headers
     (D       : in out Data;
      Headers : AWS.Headers.List);
   --  Set response's Headers

   procedure Status_Code
     (D     : in out Data;
      Value : Messages.Status_Code)
     with Inline;
   --  Set the status code

   procedure Content_Type
     (D     : in out Data;
      Value : String)
     with Inline;
   --  Set the MIME type for the message body

   procedure Content_Length (D : in out Data);
   --  Set Content-Length header field appropriate to message body size.

   procedure Expires
     (D     : in out Data;
      Value : Calendar.Time)
     with Inline;
   --  Set the Expires date

   procedure Expires
     (D     : in out Data;
      Value : String)
     with Inline;
   --  As above but with a preformatted HTTP_Date

   procedure Cache_Control
     (D     : in out Data;
```

(continues on next page)

(continued from previous page)

```
    Value : Messages.Cache_Option)
  with Inline;
--  Set the Cache_Control mode for the message

procedure Location
  (D     : in out Data;
   Value : String)
  with Inline;
--  Set the location for the new page in the case of a moved
--  message. Should be used with redirection 3xx status codes.

procedure Authentication
  (D     : in out Data;
   Realm : String;
   Mode  : Authentication_Mode := Basic;
   Stale : Boolean             := False)
  with Inline;
--  Set the authentication mode requested by server. Set the status code to
--  the 401.

procedure Clear_Session (D : in out Data);
--  Send a command to clear the cookie on the client side. This will remove
--  the session Id from the client. This routine should be used when a
--  client logout from the Web application.

----------
-- Data --
----------

procedure Clear (D : in out Data);
--  Clear all internal data

procedure Mode
  (D     : in out Data;
   Value : Data_Mode)
  with Inline;
--  Set the data mode:
--  Header, Message, File, Stream, Socket_Taken or No_Data.

procedure Filename
  (D     : in out Data;
   Value : String)
  with Inline;
--  Set the filename which should be sent back.
--  It also set the Mode field to File.

procedure Stream
  (D        : in out Data;
   Handle   : not null access Resources.Streams.Stream_Type'Class;
   Encoding : Messages.Content_Encoding := Messages.Identity)
  with Inline;
--  Set the user defined data stream.
```

(continues on next page)

(continued from previous page)

```
   --  Encoding mean additional encoding would be applied on top of given
   --  Handler stream.

   procedure Close_Resource
     (D     : in out Data;
      State : Boolean);
   --  Set the server close state, if State if False the resource will not be
   --  closed. This is needed to build transient resources as the closing must
   --  be controlled by the transient task cleaner and not the server.

   procedure Keep_Alive (D : in out Data; State : Boolean) with Inline;
   --  Keep alive connection control. Setting this flag to False will send
   --  "Connection: close" in server's response header line and the socket
   --  will be closed after the response. This flag is True by default.

   procedure Data_Encoding
     (D         : in out Data;
      Encoding  : Messages.Content_Encoding;
      Direction : Encoding_Direction := Encode);
   --  Set data encoding, the encoding will be used for the Message_Body and
   --  Append_Body routines below.
   --  Direction Encode is for server side, Direction Decode is for client
   --  side. This routine have to be called before calling Message_Body or
   --  Append_Body routines to activate the encoding. Note that by default no
   --  encoding is done if Data_Encoding is not called (Encoding => Identity).

   procedure Message_Body
     (D     : in out Data;
      Value : Streams.Stream_Element_Array)
     with Inline;
   --  Set message body as a binary content. Set the Mode field to Message

   procedure Message_Body
     (D     : in out Data;
      Value : Strings.Unbounded.Unbounded_String)
     with Inline;
   --  Set the message body content as a unbounded_string. Set the Mode field
   --  to Message.

   procedure Message_Body
     (D     : in out Data;
      Value : String)
     with Inline;
   --  Set the message body content as a string. Set the Mode field to Message

   procedure Append_Body
     (D    : in out Data;
      Item : Streams.Stream_Element_Array);
   --  Add Item to the message

   procedure Append_Body (D : in out Data; Item : String);
   --  Add Item to the message
```

(continues on next page)

(continued from previous page)

```
   ---------------
   -- Other API --
   ---------------

   function Is_Valid (D : Data) return Boolean;
   --  Checking validity of the HTTP response

end AWS.Response.Set;
```

13.46 AWS.Server

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with Ada.Task_Identification;

with AWS.Config;
with AWS.Default;
with AWS.Dispatchers;
with AWS.Exceptions;
with AWS.Net.Poll_Events;
with AWS.Net.SSL;
with AWS.Net.Std;
with AWS.Response;
with AWS.Status;

private with Ada.Calendar;
private with Ada.Exceptions;
private with Ada.Finalization;
private with Ada.Task_Attributes;
private with Ada.Real_Time;
private with GNATCOLL.Refcount;
private with System;
```

(continues on next page)

(continued from previous page)

```
private with AWS.Log;
private with AWS.Net.Acceptors;
private with AWS.HTTP2.Connection;
private with AWS.HTTP2.HPACK.Table;
private with AWS.Hotplug;
private with AWS.Utils;

package AWS.Server is

   type HTTP is limited private;
   --  A Web server

   ---------------------------
   -- Server initialization --
   ---------------------------

   --  Note that starting a sercure server if AWS has not been configured to
   --  support HTTPS will raise Program_Error.

   procedure Start
     (Web_Server : in out HTTP;
      Callback   : Response.Callback;
      Config     : 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 : Dispatchers.Handler'Class;
      Config     : 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 keept into Web_Server. Any
   --  changes done to the Dispatcher object will not be part of the Web
   --  server dispatcher.

   procedure Get_Message_Body;
   --  If size of message body is bigger than Upload_Size_Limit configuration
   --  parameter, server do not receive message body before calling user's
   --  callback routine. If user decide to get the message body he should call
   --  this routine.

   procedure Start
     (Web_Server                : in out HTTP;
      Name                      : String;
      Callback                  : Response.Callback;
      Max_Connection            : Positive  := Default.Max_Connection;
      Admin_URI                 : String    := Default.Admin_URI;
      Host                      : String    := "";
      Port                      : Natural   := Default.Server_Port;
```

(continues on next page)

(continued from previous page)

```
   Security                  : Boolean   := False;
   Session                   : Boolean   := False;
   Case_Sensitive_Parameters : Boolean   := True;
   Upload_Directory          : String    := Default.Upload_Directory;
   Line_Stack_Size           : 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
--  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 : 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 : HTTP) return AWS.Config.Object;
--  Returns configuration object for Web_Server

procedure Set_Unexpected_Exception_Handler
  (Web_Server : in out HTTP;
   Handler    : 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.
```

(continues on next page)

(continued from previous page)

```
   procedure Set
     (Web_Server : in out HTTP;
      Dispatcher : 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. Note that any change in a dispatcher associated with a server
   --  using Register or Unregister must be reset into the server using this
   --  routine.

   procedure Set_Security
     (Web_Server           : in out HTTP;
      Certificate_Filename : String;
      Security_Mode        : Net.SSL.Method := Net.SSL.TLS_Server;
      Key_Filename         : 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_SSL_Config
     (Web_Server : in out HTTP; SSL_Config : Net.SSL.Config);
   --  Set the SSL configuration for this server

   function SSL_Config
     (Web_Server : in out HTTP) return not null access Net.SSL.Config;
   --  Returns the access to SSL config of the server. Allow to change SSL
   --  config on the already created server.

   procedure Set_Socket_Constructor
     (Web_Server         : in out HTTP;
      Socket_Constructor : 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 not null access HTTP;
   --  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.
```

(continues on next page)

(continued from previous page)

```
   function Session_Name return String;
   --  Returns the current session cookie name

   function Session_Private_Name return String;
   --  Returns the current private session cookie name

   ---------------
   -- Other API --
   ---------------

   procedure Give_Back_Socket
     (Web_Server : in out HTTP; Socket : 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     : not null Net.Socket_Access);
   --  Idem.
   --  Use Socket_Access to avoid memory reallocation for already allocated
   --  sockets.

   procedure Set_Field (Id, Value : String);
   --  Set the extended log field value for the server the controlling the
   --  current task.

   procedure Skip_Log_Record;
   --  Disable logging only for the current processing request

   procedure Add_Listening
     (Web_Server    : in out HTTP;
      Host          : String;
      Port          : Natural;
      Family        : Net.Family_Type := Net.Family_Unspec;
      Reuse_Address : Boolean         := False;
      IPv6_Only     : Boolean         := False);
   --  Add the binded/listening socket on host, port and protocol family. To be
   --  able to connect web enabled application with others in the internal
   --  network, and then give access for external clients by listening on
   --  externally available address. Also it could be used to bind one server
   --  to IPv4 and IPv6 protocols simultaneously.
   --  IPv6_Only allows restrict IPv6 server to accept only IPv6 connections.

   type Task_Id_Array is
     array (Positive range <>) of Ada.Task_Identification.Task_Id;

   function Line_Tasks (Web_Server : HTTP) return Task_Id_Array;
   --  Returns line tasks identifiers

private
   -- implementation removed
```

(continues on next page)

(continued from previous page)

```
end AWS.Server;
```

13.47 AWS.Server.Hotplug

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 aws_password
   --                 tool to generate such password
   --  host         : The host name where requests will be redirected
   --  port         : and the corresponding port
```

(continues on next page)

(continued from previous page)

```
   procedure Activate
     (Web_Server         : not null access HTTP;
      Port               : Natural;
      Authorization_File : String;
      Register_Mode      : AWS.Hotplug.Register_Mode := AWS.Hotplug.Add;
      Host               : String                    := "";
      Bound_Port         : access Positive           := null);
   --  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 securite the host of redictection
   --  must also be specified.
   --  If Port is zero then the hotplug will be bound on any free port. The
   --  Bound_Port access parameter should be defined in this case and bound
   --  port will be written there.

   procedure Shutdown;
   --  Shutdown hotplug server

end AWS.Server.Hotplug;
```

13.48 AWS.Server.Log

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.Log;

package AWS.Server.Log is

   ------------------
   -- Standard Log --
   ------------------

   procedure Start
     (Web_Server      : in out HTTP;
      Split_Mode      : AWS.Log.Split_Mode := AWS.Log.None;
      Filename_Prefix : String             := "";
      Auto_Flush      : Boolean            := False);
   --  Activate server's logging activity. See AWS.Log. If Auto_Flush is True
   --  the file will be flushed after all written data.

   procedure Start
     (Web_Server : in out HTTP;
      Callback   : AWS.Log.Callback;
      Name       : String);
   --  Activate the Web_Server access log and direct all data to the Callback.
   --  The Name String is returned when the Name function is called. It is a
```

(continues on next page)

(continued from previous page)

```
   --  simple identifier, that serves no other purpose than to give the
   --  Callback a label.

   function Name (Web_Server : HTTP) return String;
   --  Return the name of the Log or an empty string if one is not active. If
   --  an external writer is used to handle the access log, then the name of
   --  that writer is returned. See the Start procedure for starting the access
   --  log with a Callback.

   procedure Stop (Web_Server : in out HTTP);
   --  Stop server's logging activity. See AWS.Log

   function Is_Active (Web_Server : 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. If a Callback procedure is used to handle the log
   --  data, then calling Flush does nothing.

   ---------------
   -- Error Log --
   ---------------

   procedure Start_Error
     (Web_Server      : in out HTTP;
      Split_Mode      : AWS.Log.Split_Mode := AWS.Log.None;
      Filename_Prefix : String             := "");
   --  Activate server's logging activity. See AWS.Log

   procedure Start_Error
     (Web_Server : in out HTTP;
      Callback   : AWS.Log.Callback;
      Name       : String);
   --  Activate the Web_Server error log and direct all data to the Callback.
   --  The Name String is returned when the Error_Name function is called. It
   --  is a simple identifier, that serves no other purpose than to give the
   --  Callback a label.

   function Error_Name (Web_Server : HTTP) return String;
   --  Return the name of the Error Log or an empty string if one is not
   --  active. If a Callback is used to handle the error log, then the name of
   --  the Callback is returned. See the Start_Error procedure for starting the
   --  error log with a Callback.

   procedure Stop_Error (Web_Server : in out HTTP);
   --  Stop server's logging activity. See AWS.Log

   function Is_Error_Active (Web_Server : HTTP) return Boolean;
   --  Returns True if the Web Server error log has been activated
```

(continues on next page)

(continued from previous page)

```
end AWS.Server.Log;
```

13.49 AWS.Server.Push

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2017, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.Calendar;
with Ada.Streams;
with Ada.Strings.Unbounded;

with AWS.Containers.Tables;
with AWS.Default;
with AWS.Net;

with System;

private with Ada.Containers.Indefinite_Hashed_Sets;
private with Ada.Containers.Indefinite_Hashed_Maps;
private with Ada.Containers.Indefinite_Doubly_Linked_Lists;
private with Ada.Strings.Hash;

generic
```

(continues on next page)

(continued from previous page)

```
   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 : Client_Output_Type;
      Client : 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;
   use Ada.Streams;
   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.
   --  Chuncked  : 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 Wait_Counter_Type is mod System.Max_Binary_Modulus;

   subtype Group_Set is Containers.Tables.VString_Array;

   Empty_Group : constant Group_Set := (1 .. 0 => Null_Unbounded_String);
```

(continues on next page)

(continued from previous page)

```
   procedure Register
     (Server            : in out Object;
      Client_Id         : Client_Key;
      Socket            : Net.Socket_Access;
      Environment       : Client_Environment;
      Init_Data         : Client_Output_Type;
      Init_Content_Type : String             := "";
      Kind              : Mode               := Plain;
      Duplicated_Age    : Duration           := Duration'Last;
      Groups            : Group_Set          := Empty_Group;
      Timeout           : Duration           := Default.Send_Timeout);
   --  Add client identified by Client_Id to the server subscription
   --  list and send the Init_Data (as a Init_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).
   --  The Timeout is not for socket send timeout, but for internal waiting for
   --  write availability timeout.

   procedure Register
     (Server            : in out Object;
      Client_Id         : Client_Key;
      Socket            : Net.Socket_Type'Class;
      Environment       : Client_Environment;
      Init_Data         : Client_Output_Type;
      Init_Content_Type : String             := "";
      Kind              : Mode               := Plain;
      Duplicated_Age    : Duration           := Duration'Last;
      Groups            : Group_Set          := Empty_Group;
      Timeout           : Duration           := Default.Send_Timeout);
   --  Same as above but with Socket_Type'Class parameter.
   --  Is not recommended, use above one with Socket_Access parameter.

   procedure Register
     (Server         : in out Object;
      Client_Id      : Client_Key;
      Socket         : Net.Socket_Type'Class;
      Environment    : Client_Environment;
      Content_Type   : String             := "";
      Kind           : Mode               := Plain;
      Duplicated_Age : Duration           := Duration'Last;
      Groups         : Group_Set          := Empty_Group;
      Timeout        : Duration           := Default.Send_Timeout);
   --  Same as above but without sending initial data.
   --  Content_Type applicable only when Kind parameter is Plain or Chunked,
   --  in Multipart server push mode each server push message would have own
   --  Content_Type defined.
   --  Is not recommended, use above one with Socket_Access parameter.

   procedure Unregister
     (Server       : in out Object;
```

(continues on next page)

(continued from previous page)

```
   Client_Id    : Client_Key;
   Close_Socket : 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 : 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 Subscribe
  (Server : in out Object; Client_Id : Client_Key; Group_Id : String);
--  Subscribe client to the group

procedure Subscribe_Copy
  (Server : in out Object; Source : String; Target : String);
--  Subscribe everybody in the group Source to the group Target.
--  If Source is empty then subscribe all clients to the group Target.

procedure Unsubscribe
  (Server : in out Object; Client_Id : Client_Key; Group_Id : String);
--  Remove group from client's group list

procedure Unsubscribe_Copy
  (Server : in out Object; Source : String; Target : String);
--  Unsubscribe everybody in the group Source from the group Target.
--  If Source is empty then unsubscribe all clients from the group Target.

procedure Send_To
  (Server       : in out Object;
   Client_Id    : Client_Key;
   Data         : Client_Output_Type;
   Content_Type : String             := "";
   Thin_Id      : String             := "");
--  Push data to a specified client identified by Client_Id
--  Thin_Id is to be able to replace messages in the send client queue
--  with the newer one with the same Thin_Id.

procedure Send
  (Server       : in out Object;
   Data         : Client_Output_Type;
   Group_Id     : String             := "";
   Content_Type : String             := "";
   Thin_Id      : String             := "";
   Client_Gone  : access procedure (Client_Id : String) := null);
--  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.
--  Thin_Id is to be able to replace messages in the send client queue
```

(continues on next page)

(continued from previous page)

```
   --  with the newer one with the same Thin_Id.

   generic
      with procedure Client_Gone (Client_Id : String);
   procedure Send_G
     (Server       : in out Object;
      Data         : Client_Output_Type;
      Group_Id     : String             := "";
      Content_Type : String             := "";
      Thin_Id      : String             := "");
   --  Same like before, but generic for back compatibility

   function Count (Server : Object) return Natural;
   --  Returns the number of registered clients in the server

   procedure Info
     (Server  : in out Object;
      Clients : out Natural;
      Groups  : out Natural;
      Process : access procedure
                  (Client_Id   : Client_Key;
                   Address     : String;
                   State       : String;
                   Environment : Client_Environment;
                   Kind        : Mode;
                   Groups      : Group_Set) := null);
   --  Returns the number of registered clients and groups in the server.
   --  Call Process routine for each client if defined.
   --  Test internal integrity.

   function Is_Open (Server : 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
   --  finalization data.

   procedure Shutdown
     (Server : in out Object; Close_Sockets : 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         : Client_Output_Type;
      Final_Content_Type : String             := "");
   --  Idem as above but it send Final_Data (as a Data_Content_Type mime
   --  content) before closing connections.
```

(continues on next page)

(continued from previous page)

```
   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.

   procedure Info
     (Size        : out Natural;
      Max_Size    : out Natural;
      Max_Size_DT : out Calendar.Time;
      Counter     : out Wait_Counter_Type);
   --  Size would return number of currently waiting sockets.
   --  Counter would return total number of waited sockets from start.

   function Wait_Send_Completion (Timeout : Duration) return Boolean;
   --  Wait for all data sending in all server_push objects of the current
   --  package instance.
   --  Return True if wait successful. False in timeout.

   type Error_Handler is not null access procedure (Message : String);

   procedure Set_Internal_Error_Handler (Handler : Error_Handler);
   --  Set the handler of the internal fatal errors

private
   -- implementation removed
end AWS.Server.Push;
```

13.50 AWS.Server.Status

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 : HTTP) return Templates.Translate_Set;
   --  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 Translations (Server : HTTP) return Templates.Translate_Table;
   pragma Obsolescent ("Use Translate_Set return value instead");
   --  The same as above but obsolete and keept for backward compartibility

   function Start_Time (Server : HTTP) return Ada.Calendar.Time;
   --  Returns the server's start time

   function Resources_Served (Server : HTTP) return Natural;
   --  Returns the total number of resources (static file, templates,
```

(continues on next page)

(continued from previous page)

```
   --  in-memory string) served by the server.

   function Socket (Server : HTTP) return Net.Socket_Type'Class;
   --  Returns the main server's socket

   function Sockets (Server : HTTP) return Net.Socket_List;
   --  Returns all server's sockets

   function Port (Server : HTTP) return Positive;
   --  Returns the server's socket port

   function Host (Server : HTTP) return String;
   --  Returns the server's socket host

   function Is_Any_Address (Server : HTTP) return Boolean;
   --  Returns True if the server accepts connections on any of the host's
   --  network addresses.

   function Is_IPv6 (Server : HTTP) return Boolean;
   --  Returns True if Server is using IPv6

   function Local_URL (Server : HTTP) return String;
   --  Local URL of the server

   function Current_Connections (Server : HTTP) return Natural;
   --  Returns the current number of connections

   function Active_Tasks (Server : HTTP) return Natural;
   --  Returns the current number of active processing tasks

   function Is_Session_Activated (Server : HTTP) return Boolean;
   --  Returns True if the session feature has been activated

   function Is_Security_Activated (Server : HTTP) return Boolean;
   --  Returns True if the HTTPS protocol is used

   function Is_Shutdown (Server : HTTP) return Boolean;
   --  Returns True if server has been stopped (the server could still be in
   --  the shutdown phase).

end AWS.Server.Status;
```

13.51 AWS.Services.Callbacks

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2017, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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

   generic
      Prefix    : String; -- the prefix found in the URI
      Directory : String; -- the directory where the file is
   function File (Request : 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;
```

13.52 AWS.Services.Directory

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.
```

(continues on next page)

(continued from previous page)

```
--
--     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.
```

(continues on next page)

(continued from previous page)

```
--
--     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 : String;
      Request        : AWS.Status.Data) return Translate_Set;
   --  Returns a translation table containing information parsed from
   --  Directory_Name. This is supposed to be used with a directory template.

   function Browse
     (Directory_Name    : String;
      Template_Filename : String;
      Request           : AWS.Status.Data;
      Translations      : Translate_Set := Null_Set) 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;
```

13.53 AWS.Services.Dispatchers

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2013, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

package AWS.Services.Dispatchers with Pure is

   --  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 resource 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.
   --
```

(continues on next page)

(continued from previous page)

```
   --  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;
```

13.54 AWS.Services.Dispatchers.Linker

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2005-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;

   procedure Register
     (Dispatcher    : in out Handler;
      First, Second : 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;
```

13.55 AWS.Services.Dispatchers.Method

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;

   procedure Register
     (Dispatcher : in out Handler;
      Method     : Status.Request_Method;
      Action     : AWS.Dispatchers.Handler'Class);
   --  Register callback to use for a specific request method

   procedure Register
     (Dispatcher : in out Handler;
      Method     : Status.Request_Method;
      Action     : Response.Callback);
   --  Idem as above but take a callback procedure as parameter
```

(continues on next page)

(continued from previous page)

```
   procedure Unregister
     (Dispatcher : in out Handler;
      Method     : Status.Request_Method);
   --  Removes Method from the list of request method to handle

   procedure Register_Default_Callback
     (Dispatcher : in out Handler;
      Action     : 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;
```

13.56 AWS.Services.Dispatchers.URI

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 AWS.Dispatchers;
with AWS.Response;
with AWS.Status;
with AWS.Utils;

private with Ada.Containers.Vectors;
private with Ada.Strings.Unbounded;

package AWS.Services.Dispatchers.URI is

   type Handler is new AWS.Dispatchers.Handler with private;

   procedure Register
     (Dispatcher : in out Handler;
      URI        : String;
      Action     : AWS.Dispatchers.Handler'Class;
      Prefix     : Boolean := False);
   --  Register URI to use the specified dispatcher. URI is the full string
   --  that must match the resource requested (with the leading /). If Prefix
   --  is True, only the URI prefix is checked.
```

(continues on next page)

(continued from previous page)

```
   procedure Register
     (Dispatcher : in out Handler;
      URI        : String;
      Action     : Response.Callback;
      Prefix     : Boolean := False);
   --  Idem as above but take a callback procedure as parameter

   procedure Register_Regexp
     (Dispatcher : in out Handler;
      URI        : String;
      Action     : AWS.Dispatchers.Handler'Class);
   --  Register URI to use the specified dispatcher. URI is a regular
   --  expression that must match the resource requested (with the leading /).

   procedure Register_Regexp
     (Dispatcher : in out Handler;
      URI        : String;
      Action     : Response.Callback);
   --  Idem as above but take a callback procedure as parameter

   procedure Unregister
     (Dispatcher : in out Handler;
      URI        : String);
   --  Removes URI from the list. URI is either a name or a regexp and must
   --  have exactly the value used with Register.

   procedure Register_Default_Callback
     (Dispatcher : in out Handler;
      Action     : 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;
```

13.57 AWS.Services.Dispatchers.Virtual_Host

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.Dispatchers;
with AWS.Response;
with AWS.Status;

private with Ada.Containers.Indefinite_Hashed_Maps;
private with Ada.Strings.Hash;
private with Ada.Strings.Unbounded;

package AWS.Services.Dispatchers.Virtual_Host is

   type Handler is new AWS.Dispatchers.Handler with private;

   procedure Register
     (Dispatcher       : in out Handler;
      Virtual_Hostname : String;
      Hostname         : String);
   --  Register Virtual_Hostname to be a redirection to the specified
   --  hostname.

   procedure Register
     (Dispatcher       : in out Handler;
      Virtual_Hostname : String;
```

(continues on next page)

(continued from previous page)

```
      Action           : AWS.Dispatchers.Handler'Class);
   --  Register Virtual_Hostname to use the specified callback

   procedure Register
     (Dispatcher       : in out Handler;
      Virtual_Hostname : String;
      Action           : Response.Callback);
   --  Idem as above but take a callback procedure as parameter

   procedure Unregister
     (Dispatcher       : in out Handler;
      Virtual_Hostname : String);
   --  Removes Virtual_Hostname from the list of virtual hostnames to handle

   procedure Register_Default_Callback
     (Dispatcher : in out Handler;
      Action     : 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;
```

13.58 AWS.Services.Download

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2005-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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       : AWS.Dispatchers.Handler'Class;
      Main_Dispatcher         : out Services.Dispatchers.Linker.Handler;
      Max_Concurrent_Download : 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
```

(continues on next page)

(continued from previous page)

```
   --  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  : Status.Data;
      Name     : 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;
```

13.59 AWS.Services.Page_Server

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2013, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 resource is a directory. There is two specials files that
--  are recognized:
--
--    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.
--
--    aws_directory.thtml   The template page used for directory browsing.
--                          See AWS.Services.Directory for a full description
--                          of this template usage.

with AWS.Messages;
with AWS.Response;
```

(continues on next page)

(continued from previous page)

```
with AWS.Status;

package AWS.Services.Page_Server is

   procedure Directory_Browsing (Activated : Boolean);
   --  If Activated is set to True the directory browsing facility will be
   --  activated. By default this feature is not activated.

   procedure Set_Cache_Control (Data : Messages.Cache_Data);
   --  Set the Cache-Control header for each response given by the following
   --  callback.

   function Callback (Request : Status.Data) return Response.Data;
   --  This is the AWS callback for the simple static Web pages server

end AWS.Services.Page_Server;
```

13.60 AWS.Services.Split_Pages

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;
```

(continues on next page)

(continued from previous page)

```
--  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  : Splitter;
   Table : 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   : Splitter;
   Page   : Positive;
   URIs   : URI_Table;
   Ranges : Ranges_Table) return Templates.Translate_Set is abstract;
--  Get_Translations builds the translation table for use with the splitter

function Parse
  (Template     : String;
   Translations : Templates.Translate_Set;
   Table        : Templates.Translate_Set;
   Split_Rule   : Splitter'Class;
   Cached       : Boolean := True) return Response.Data;

function Parse
  (Template     : String;
   Translations : Templates.Translate_Table;
   Table        : Templates.Translate_Table;
   Split_Rule   : Splitter'Class;
   Cached       : 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     : String;
   Translations : Templates.Translate_Table;
   Table        : Templates.Translate_Table;
   Max_Per_Page : Positive := 25;
```

(continues on next page)

(continued from previous page)

```
      Max_In_Index : Positive := 20;
      Cached       : 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.thtml
   --  template for displaying the index.

private
   -- implementation removed
end AWS.Services.Split_Pages;
```

13.61 AWS.Services.Split_Pages.Alpha

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.
```

(continues on next page)

(continued from previous page)

```
   --  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.

   Splitter_Error : exception renames Split_Pages.Splitter_Error;

   type Splitter is new Split_Pages.Splitter with private;

   overriding function Get_Page_Ranges
     (This  : Splitter;
      Table : Templates.Translate_Set) return Ranges_Table;

   overriding function Get_Translations
     (This   : Splitter;
      Page   : Positive;
      URIs   : URI_Table;
      Ranges : Ranges_Table) return Templates.Translate_Set;

   procedure Set_Key (This : in out Splitter; Key : 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 : 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;
```

13.62 AWS.Services.Split_Pages.Alpha.Bounded

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.
```

(continues on next page)

(continued from previous page)

```
   type Splitter (Max_Per_Page : Positive) is new Alpha.Splitter with private;

   overriding function Get_Page_Ranges
     (This  : Splitter;
      Table : Templates.Translate_Set) return Ranges_Table;

   overriding function Get_Translations
     (This   : Splitter;
      Page   : Positive;
      URIs   : URI_Table;
      Ranges : Ranges_Table) return Templates.Translate_Set;

private
   -- implementation removed
end AWS.Services.Split_Pages.Alpha.Bounded;
```

13.63 AWS.Services.Split_Pages.Uniform

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;

   overriding function Get_Page_Ranges
```

(continues on next page)

(continued from previous page)

```
     (This  : Splitter;
      Table : Templates.Translate_Set) return Ranges_Table;

   overriding function Get_Translations
     (This   : Splitter;
      Page   : Positive;
      URIs   : URI_Table;
      Ranges : Ranges_Table) return Templates.Translate_Set;

private
   -- implementation removed
end AWS.Services.Split_Pages.Uniform;
```

13.64 AWS.Services.Split_Pages.Uniform.Alpha

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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
```

(continues on next page)

(continued from previous page)

```
   --                "<>", "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.

   Splitter_Error : exception renames Split_Pages.Splitter_Error;

   type Splitter is new Uniform.Splitter with private;

   overriding function Get_Page_Ranges
     (This  : Splitter;
      Table : Templates.Translate_Set) return Ranges_Table;

   overriding function Get_Translations
     (This   : Splitter;
      Page   : Positive;
      URIs   : URI_Table;
      Ranges : Ranges_Table) return Templates.Translate_Set;

   procedure Set_Key (This : in out Splitter; Key : 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;
```

13.65 AWS.Services.Split_Pages.Uniform.Overlapping

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2004-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;

   overriding function Get_Page_Ranges
     (This  : Splitter;
      Table : Templates.Translate_Set) return Ranges_Table;

private
   -- implementation removed
end AWS.Services.Split_Pages.Uniform.Overlapping;
```

13.66 AWS.Services.Transient_Pages

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with AWS.Config;
with AWS.Resources.Streams;

package AWS.Services.Transient_Pages is

   function Get_URI return String with
     Post => Get_URI'Result'Length > 0;
   --  Create a unique URI, must be used to register a transient web page

   procedure Register
     (URI      : String;
      Resource : Resources.Streams.Stream_Access;
      Lifetime : Duration := Config.Transient_Lifetime);
   --  Register a new transient page, this page will be deleted after Lifetime
   --  seconds.

   function Get (URI : String) return Resources.Streams.Stream_Access;
   --  Returns the stream access for the URI or null if this URI has not been
   --  registered.
```

(continues on next page)

(continued from previous page)

```
private
   -- implementation removed
end AWS.Services.Transient_Pages;
```

13.67 AWS.Services.Web_Block

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2007-2013, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  Enhanced Contextual Web Framework

package AWS.Services.Web_Block with Pure is

end AWS.Services.Web_Block;
```

13.68 AWS.Services.Web_Block.Context

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2007-2015, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

private with Ada.Containers.Indefinite_Hashed_Maps;
private with Ada.Strings.Hash;

private with GNAT.SHA1;

package AWS.Services.Web_Block.Context is

   type Object is tagged private;
   --  A context object, can be used to record key/name values

   Empty : constant Object;

   type Id is private;
   --  An object Id, the Id depends only on the context content. Two context
   --  with the very same content will have the same Id.

   function Image (CID : Id) return String;
   --  Returns CID string representation

   function Value (CID : String) return Id;
```

(continues on next page)

(continued from previous page)

```
   --  Returns Id given it's string representation

   function Register (Context : Object) return Id
     with Post => Exist (Register'Result);
   --  Register the context into the database, returns its Id

   function Exist (CID : Id) return Boolean;
   --  Returns True if CID context exists into the database

   function Get (CID : Id) return Object;
   --  Returns the context object corresponding to CID

   procedure Set_Value (Context : in out Object; Name, Value : String)
     with Post => Context.Exist (Name);
   --  Add a new name/value pair (replace name/value if already present)

   function Get_Value (Context : Object; Name : String) return String
     with Post => (if not Context.Exist (Name) then Get_Value'Result = "");
   --  Returns the value for the key Name or an empty string if does not exist

   function Exist (Context : Object; Name : String) return Boolean;
   --  Returns true if the key Name exist in this context

   procedure Remove (Context : in out Object; Name : String)
     with Post => not Context.Exist (Name);
   --  Remove the context for key Name

   generic
      type Data is private;
      Null_Data : Data;
   package Generic_Data is

      procedure Set_Value
        (Context : in out Object;
         Name    : String;
         Value   : Data)
        with Post => Context.Exist (Name);
      --  Set key/pair value for the SID

      function Get_Value (Context : Object; Name : String) return Data
        with
          Inline,
          Post => (if not Context.Exist (Name)
                   then Get_Value'Result = Null_Data);
      --  Returns the Value for Key in the session SID or Null_Data if
      --  key does not exist.

   end Generic_Data;

private
   -- implementation removed
end AWS.Services.Web_Block.Context;
```

13.69 AWS.Services.Web_Block.Registry

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2007-2014, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;
with AWS.Messages;
with AWS.MIME;
with AWS.Response;
with AWS.Services.Web_Block.Context;
with AWS.Status;
with AWS.Templates;

package AWS.Services.Web_Block.Registry is

   use Ada;
   use Ada.Strings.Unbounded;

   type Page is record
      Content      : Unbounded_String;
      --  Rendered page
      Content_Type : Unbounded_String;
      --  The page's content type
      Set          : Templates.Translate_Set;
      --  The translate set used to render the page
```

(continues on next page)

(continued from previous page)

```
   Ctx_Id       : Context.Id;
   --  The page context id
end record;

No_Page : constant Page;

type Data_Callback is access procedure
  (Request      : Status.Data;
   Context      : not null access Web_Block.Context.Object;
   Translations : in out Templates.Translate_Set);

type Callback_Parameters is new Containers.Tables.VString_Array;
Empty_Callback_Parameters : Callback_Parameters (1 .. 0);

type Data_With_Param_Callback is access procedure
  (Request      : Status.Data;
   Context      : not null access Web_Block.Context.Object;
   Parameters   : Callback_Parameters;
   Translations : in out Templates.Translate_Set);

type Template_Callback is access
  function (Request : Status.Data) return String;

procedure Register
  (Key              : String;
   Template         : String;
   Data_CB          : Data_Callback;
   Content_Type     : String  := MIME.Text_HTML;
   Prefix           : Boolean := False;
   Context_Required : Boolean := False);
--  Key is a Lazy_Tag or template page name. Template is the corresponding
--  template file. Data_CB is the callback used to retrieve the translation
--  table to render the page. If Context_Required is True a proper context
--  must be present when rendering the page otherwise Context_Error callback
--  (see Build below) is called.

procedure Register
  (Key              : String;
   Template_CB      : Template_Callback;
   Data_CB          : Data_Callback;
   Content_Type     : String := MIME.Text_HTML;
   Context_Required : Boolean := False);
--  Key is a Lazy_Tag or template page name. Template_CB is the callback
--  used to retrieve the corresponding template file name. Data_CB is the
--  callback used to retrieve the translation table to render the page.

procedure Register_Pattern_URL
  (Prefix           : String;
   Regexp           : String;
   Template         : String;
   Data_CB          : Data_With_Param_Callback;
   Content_Type     : String  := MIME.Text_HTML;
```

(continues on next page)

(continued from previous page)

```
      Context_Required : Boolean := False);
   --  Prefix is the prefix key to match
   --  Then the rest of the url is a regular expression defined by Regexp
   --  All regular-expression groups (inside parenthesis) is captured and pass
   --  to the Data_CB in the Parameters vector
   --  For instance, with:
   --      Prefix = '/page/'
   --      Regexp = '([0-9]+)/section-([a-z]+)/.*'
   --  The url '/page/42/section-b/part2' will be matched and Data_CB will
   --  be called with Parameters = <42, "b">

   procedure Register_Pattern_URL
     (Prefix           : String;
      Regexp           : String;
      Template_CB      : Template_Callback;
      Data_CB          : Data_With_Param_Callback;
      Content_Type     : String  := MIME.Text_HTML;
      Context_Required : Boolean := False);
   --  Same as above but takes a Template_Callback

   function Parse
     (Key           : String;
      Request       : Status.Data;
      Translations  : Templates.Translate_Set;
      Context       : Web_Block.Context.Object := Web_Block.Context.Empty;
      Context_Error : String := "") return Page;
   --  Parse the Web page registered under Key. Context_Error is the key
   --  of the registered template to use when a required context is not
   --  present.

   function Content_Type (Key : String) return String;
   --  Returns the Content_Type recorded for the web object

   function Build
     (Key           : String;
      Request       : Status.Data;
      Translations  : Templates.Translate_Set;
      Status_Code   : Messages.Status_Code := Messages.S200;
      Cache_Control : Messages.Cache_Option := Messages.Unspecified;
      Context       : access Web_Block.Context.Object := null;
      Context_Error : String := "") return Response.Data;
   --  Same as above but returns a standard Web page. If Context is set it
   --  is the initial value and will be setup at the end to correspond to
   --  the recorded new context.

   function Get_Context
     (Request : Status.Data) return Web_Block.Context.Object;
   --  Gets the proper context object for this request. Note that if the
   --  context object is modified outside of the Web_Block framework it must be
   --  passed to the Build or Parse procedure above.

private
```

(continues on next page)

(continued from previous page)

```
   -- implementation removed
end AWS.Services.Web_Block.Registry;
```

13.70 AWS.Session

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2020, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  This is the API to handle session data for each client connected

with Ada.Calendar;

private with AWS.Config;

package AWS.Session is

   use Ada;

   type Id is private;

   type Value_Kind is (Int, Str, Real, Bool, User);

   No_Session : constant Id;

   function Create return Id with
     Post => Create'Result /= No_Session;
   --  Create a new uniq Session Id
```

(continues on next page)

(continued from previous page)

```
   function Creation_Stamp (SID : Id) return Calendar.Time;
   --  Returns the creation date of this session

   function Private_Key (SID : Id) return String;
   --  Return the private key for this session

   procedure Delete (SID : Id) with
     Post => not Exist (SID);
   --  Delete session, does nothing if SID does not exist.
   --  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 Delete_If_Empty (SID : Id) return Boolean;
   --  Delete session only if there is no key/value pairs.
   --  Returns True if session deleted.
   --  Need to delete not used just created session to avoid too many empty
   --  session creation.

   function Image (SID : Id) return String with Inline;
   --  Return ID image

   function Value (SID : String) return Id with Inline;
   --  Build an ID from a String, returns No_Session if SID is not recongnized
   --  as an AWS session ID.

   function Exist (SID : Id) return Boolean;
   --  Returns True if SID exist

   procedure Touch (SID : Id);
   --  Update to current time the timestamp associated with SID. Does nothing
   --  if SID does not exist.

   procedure Set (SID : Id; Key : String; Value : String);
   --  Set key/value pair for the SID

   procedure Set (SID : Id; Key : String; Value : Integer);
   --  Set key/value pair for the SID

   procedure Set (SID : Id; Key : String; Value : Float);
   --  Set key/value pair for the SID

   procedure Set (SID : Id; Key : String; Value : Boolean);
   --  Set key/value pair for the SID

   function Get (SID : Id; Key : String) return String with
     Inline => True,
     Post   => (not Exist (SID, Key) and then Get'Result'Length = 0)
```

(continues on next page)

(continued from previous page)

```
            or else Exist (SID, Key);
   --  Returns the Value for Key in the session SID or the emptry string if
   --  key does not exist.

   function Get (SID : Id; Key : String) return Integer with
     Inline => True,
     Post   => (not Exist (SID, Key) and then Get'Result = 0)
               or else Exist (SID, Key);
   --  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 : Id; Key : String) return Float with
     Inline => True,
     Post   => (not Exist (SID, Key) and then Get'Result = 0.0)
               or else Exist (SID, Key);
   --  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 : Id; Key : String) return Boolean with
     Inline => True,
     Post   => (not Exist (SID, Key) and then Get'Result = False)
               or else Exist (SID, Key);
   --  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 : Id; Key : String; Value : Data);
      --  Set key/value pair for the SID

      function Get (SID : Id; Key : String) return Data with Inline;
      --  Returns the Value for Key in the session SID or Null_Data if
      --  key does not exist.

   end Generic_Data;

   procedure Remove (SID : Id; Key : String) with
     Post => not Exist (SID, Key);
   --  Removes Key from the specified session

   function Exist (SID : Id; Key : String) return Boolean;
   --  Returns True if Key exist in session SID

   function Server_Count return Natural;
   --  Returns number of servers with sessions support

   function Length return Natural;
   --  Returns number of sessions
```

(continues on next page)

(continued from previous page)

```
   function Length (SID : Id) return Natural;
   --  Returns number of key/value pairs in session SID

   procedure Clear with Post => Length = 0;
   --  Removes all sessions data

   ---------------
   -- Iterators --
   ---------------

   generic
      with procedure Action
        (N          : Positive;
         SID        : Id;
         Time_Stamp : 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          : Positive;
         Key, Value : String;
         Kind       : Value_Kind;
         Quit       : in out Boolean);
   procedure For_Every_Session_Data (SID : 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 : 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 : Id) return Boolean;
   --  Returns true if SID should be considered as expired (ie there hasn't
```

(continues on next page)

(continued from previous page)

```
   --  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 : Id);
   --  Callback procedure called when a sesssion is deleted from the server

   procedure Set_Callback (Callback : 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 : String);
   --  Save all sessions data into File_Name

   procedure Load (File_Name : String);
   --  Restore all sessions data from File_Name

private
   -- implementation removed
end AWS.Session;
```

13.71 AWS.SMTP

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2017, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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 AWS.Net;

private with Ada.Strings.Unbounded;
limited with AWS.SMTP.Authentication;

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;

   --------------
```

(continues on next page)

(continued from previous page)

```
   -- Receiver --
   --------------

   type Secure_Connection is (No, TLS, STARTTLS);

   type Receiver is private;
   --  The receiver part (i.e. a server) of SMTP messages as defined in
   --  RFC 821. This is the SMTP server.

   function Initialize
     (Server_Name : String;
      Port        : Natural := Default_SMTP_Port;
      Security    : Secure_Connection := No;
      Family      : Net.Family_Type := Net.Family_Unspec;
      Credential  : access constant Authentication.Credential'Class := null;
      Timeout     : Duration := Net.Forever)
      return Receiver;

   ----------------
   -- Reply_Code --
   ----------------

   type Reply_Code is range 200 .. 554;

   Service_Ready       : constant Reply_Code := 220;
   Service_Closing     : constant Reply_Code := 221;
   Auth_Successful     : constant Reply_Code := 235;
   Requested_Action_Ok : constant Reply_Code := 250;
   Provide_Watchword   : constant Reply_Code := 334;
   Start_Mail_Input    : constant Reply_Code := 354;
   Syntax_Error        : constant Reply_Code := 500;

   function Image (R : 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 : 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 : Reply_Code) return String;
   --  This returns the value: Image (R) & ' ' & Name (R)

   ------------
   -- Status --
   ------------

   type Status is private;

   function Is_Ok (Status : SMTP.Status) return Boolean with Inline;
   --  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
```

(continues on next page)

(continued from previous page)

```
   --  warning because something wrong (but not unrecoverable) has happen.

   function Status_Message (Status : 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 Warnings (Status : SMTP.Status) return String with Inline;
   --  Returns warnings during recipient addresses processing

   function Status_Code (Status : SMTP.Status) return Reply_Code with Inline;
   --  Returns the code replied by the server

   procedure Clear (Status : in out SMTP.Status) with Inline;
   --  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 : E_Mail_Data;
      Mode   : 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 : String; Address : String) return E_Mail_Data;
   --  Returns an e-mail address

   function Parse (E_Mail : String) return E_Mail_Data;
   --  Parse an e-mail with format "Name <address>" or "address (Name)"
   --  and Returns the corresponding E_Mail_Data. Raises Contraint_Error
   --  if E_Mail can't be parsed.

   type Recipients is array (Positive range <>) of E_Mail_Data;

   No_Recipient : constant Recipients;

private
   -- implementation removed
end AWS.SMTP;
```

13.72 AWS.SMTP.Client

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2017, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 implements an API to send email messages. It is possible to send
--  simple email [RFC 821] and email with MIME attachments [RFC 2045 & 2049].
--
--  How to send an email:
--
--  1) Initialize a Server to send the messages.
--
--     Wanadoo : SMTP.Receiver := SMTP.Client.Initialize ("smtp.wanadoo.fr");
--
--     Optionally, request Authentication
--
--     Auth : aliased SMTP.Authentication.Credential :=
--              SMTP.Authentication.Plain.Initialize ("id", "password");
--
--     Wanadoo : SMTP.Receiver :=
--                 SMTP.Client.Initialize
--                   ("smtp.wanadoo.fr", Credential => Auth'Access);
--
--  2) Send a message via the server.
--
--     Result : SMTP.Status;
```

(continues on next page)

(continued from previous page)

```
--
--     SMTP.Client.Send
--        (Server  => Wanadoo,
--         From    => SMTP.E_Mail ("Pascal Obry", "pascal@obry.net"),
--         To      => SMTP.E_Mail
--                      ("Dmitriy Anisimkov", "anisimkov@ada-ru.org"),
--         Subject => "Latest Ada news",
--         Message => "now Ada can send SMTP mail!",
--         Status  => Result);

with AWS.Attachments;

package AWS.SMTP.Client is

   Server_Error : exception renames SMTP.Server_Error;

   function Initialize
     (Server_Name : String;
      Port        : Positive := Default_SMTP_Port;
      Security    : Secure_Connection := No;
      Family      : Net.Family_Type := Net.Family_Unspec;
      Credential  : access constant Authentication.Credential'Class := null;
      Timeout     : Duration := Net.Forever)
      return Receiver renames SMTP.Initialize;

   procedure Send
     (Server  : Receiver;
      From    : E_Mail_Data;
      To      : E_Mail_Data;
      Subject : String;
      Message : String;
      Status  : out SMTP.Status;
      CC      : Recipients := No_Recipient;
      BCC     : Recipients := No_Recipient;
      To_All  : Boolean    := True);
   --  Send a message via Server. The email 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).
   --  If To_All is False email is sent even if some email addresses
   --  in recipient list are not correct.

   type Attachment is private;
   --  This is an attachment object, either a File or some Base64 encoded
   --  content.
   --  only simple attachments are supported. For full attachment support use
   --  AWS.Attachments with the corresponding Send routine below.

   function File (Filename : String) return Attachment;
   --  Returns a file attachment. Filename point to a file on the file system

   function Base64_Data (Name, Content : String) return Attachment;
   --  Returns a base64 encoded attachment. Content must already be Base64
```

(continues on next page)

(continued from previous page)

```
   --  encoded data. The 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      : Receiver;
      From        : E_Mail_Data;
      To          : E_Mail_Data;
      Subject     : String;
      Message     : String := "";
      Attachments : Attachment_Set;
      Status      : out SMTP.Status;
      CC          : Recipients := No_Recipient;
      BCC         : Recipients := No_Recipient;
      To_All      : Boolean    := True);
   --  Send a message via Server. The email is a MIME message composed of a
   --  subject, a message and a set of MIME encoded files. Raise Server_Error
   --  in case of an unrecoverable error (e.g. can't contact the server).
   --  Raises Constraint_Error if a file attachment cannot be opened.
   --  If To_All is False email is sent even if some email addresses in
   --  recipient list are not correct.

   type Message_File is new String;

   procedure Send
     (Server   : Receiver;
      From     : E_Mail_Data;
      To       : E_Mail_Data;
      Subject  : String;
      Filename : Message_File;
      Status   : out SMTP.Status;
      CC       : Recipients := No_Recipient;
      BCC      : Recipients := No_Recipient;
      To_All   : Boolean    := True);
   --  Send filename content via Server. The email 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.

   --
   --  Extentded interfaces to send a message to many recipients
   --

   procedure Send
     (Server  : Receiver;
      From    : E_Mail_Data;
      To      : Recipients;
      Subject : String;
      Message : String;
      Status  : out SMTP.Status;
```

(continues on next page)

(continued from previous page)

```
      CC      : Recipients := No_Recipient;
      BCC     : Recipients := No_Recipient;
      To_All  : Boolean    := True);
   --  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).
   --  If To_All is False email is sent even if some email addresses
   --  in recipient list are not correct.

   procedure Send
     (Server : Receiver;
      From   : E_Mail_Data;
      To     : Recipients;
      Source : String;
      Status : out SMTP.Status;
      CC     : Recipients := No_Recipient;
      BCC    : Recipients := No_Recipient;
      To_All : Boolean    := True);
   --  Send a message via Server. The email Source has already been composed by
   --  other means, such as the GNATcoll email facilities.
   --  Raise Server_Error in case of an unrecoverable error, e.g. can't contact
   --  the server.
   --  If To_All is False email is sent even if some email addresses in
   --  recipient list are not correct.

   procedure Send
     (Server      : Receiver;
      From        : E_Mail_Data;
      To          : Recipients;
      Subject     : String;
      Message     : String := "";
      Attachments : Attachment_Set;
      Status      : out SMTP.Status;
      CC          : Recipients := No_Recipient;
      BCC         : Recipients := No_Recipient;
      To_All      : Boolean    := True);
   --  Send a message via Server. The email 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 if a file attachment cannot be opened.
   --  If To_All is False email is sent even if some email addresses in
   --  recipient list are not correct.

   procedure Send
     (Server      : Receiver;
      From        : E_Mail_Data;
      To          : Recipients;
      Subject     : String;
      Attachments : AWS.Attachments.List;
      Status      : out SMTP.Status;
      CC          : Recipients := No_Recipient;
      BCC         : Recipients := No_Recipient;
```

(continues on next page)

(continued from previous page)

```
      To_All      : Boolean    := True);
   --  As above but takes an attachment list which support complex attachments
   --  like multiplart/alternative.

private
   -- implementation removed
end AWS.SMTP.Client;
```

13.73 AWS.Status

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                    --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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.Real_Time;
with Ada.Streams;
with Ada.Strings.Unbounded;

with AWS.Attachments;
with AWS.Headers;
with AWS.Messages;
with AWS.Net;
with AWS.Parameters;
with AWS.Resources.Streams.Memory;
with AWS.Session;
with AWS.URL;

private with GNAT.Calendar;
private with GNAT.SHA256;
```

(continues on next page)

(continued from previous page)

```
package AWS.Status is

   use Ada;
   use Ada.Streams;
   use Ada.Strings.Unbounded;

   type Data is private;

   type Request_Method is
     (OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT, EXTENSION_METHOD);
   --  EXTENSION_METHOD indicates that a method is an extension-method,
   --  ie none of the eight method tokens predefined in the RFC 2616.

   type Authorization_Type is (None, Basic, Digest);

   type Protocol_State is (HTTP_1, Upgrade_To_H2C, H2C, H2);
   --  Protocoal status and upgrade request

   ------------------
   -- Request-Line --
   ------------------

   function Method       (D : Data) return Request_Method with Inline;
   --  Returns the request method

   function Method       (D : Data) return String with Inline;
   --  Returns the request method as a String. Useful to get the method String
   --  for an extension-method, ie a method that is not already predefined
   --  in the RFC 2616.

   function Protocol     (D : Data) return Protocol_State with Inline;
   --  Get the current state of the protocol

   function URI          (D : Data) return String with Inline;
   --  Returns the requested resource

   function URI          (D : Data) return URL.Object with Inline;
   --  As above but return an URL object

   function URL          (D : Data) return String with Inline;
   --  Returns the requested URL

   function Parameters   (D : Data) return Parameters.List with Inline;
   --  Returns the list of parameters for the request. This list can be empty
   --  if there was no form or URL parameters.

   function Parameter
     (D : Data; Name : String; N : Positive := 1) return String with Inline;

   function HTTP_Version (D : Data) return String with Inline;
   function HTTP_Version (D : Data) return HTTP_Protocol with Inline;
```

(continues on next page)

(continued from previous page)

```
   --  Returns the HTTP version used by the client

   function Request_Time (D : Data) return Calendar.Time with Inline;
   function Request_Time (D : Data) return Real_Time.Time with Inline;
   --  Returns the time of the request

   ------------
   -- Header --
   ------------

   function Header          (D : Data) return Headers.List with Inline;
   --  Returns the list of header lines for the request

   function Accept_Encoding (D : Data) return String with Inline;
   --  Get the value for "Accept-Encoding:" header

   function Connection      (D : Data) return String with Inline;
   --  Get the value for "Connection:" header

   function Content_Length  (D : Data) return Stream_Element_Count with Inline;
   --  Get the value for "Content-Length:" header, this is the number of
   --  bytes in the message body.

   function Content_Type    (D : Data) return String with Inline;
   --  Get value for "Content-Type:" header

   function Transfer_Encoding (D : Data) return String with Inline;
   --  Get value for "Transfer-Encoding:" header

   function Expect            (D : Data) return String with Inline;
   --  Get value for "Expect:" header

   function Host              (D : Data) return String with Inline;
   --  Get value for "Host:" header

   function If_Modified_Since (D : Data) return String with Inline;
   --  Get value for "If-Modified-Since:" header

   function Keep_Alive        (D : Data) return Boolean with Inline;
   --  Returns the flag if the current HTTP connection is keep-alive

   function User_Agent        (D : Data) return String with Inline;
   --  Get value for "User-Agent:" header

   function Referer           (D : Data) return String with Inline;
   --  Get value for "Referer:" header

   function Cache_Control     (D : Data) return Messages.Cache_Option
     with Inline;
   --  Get value for "Cache-Control:" header

   function Cache_Control     (D : Data) return Messages.Cache_Data
```

(continues on next page)

(continued from previous page)

```
   with Inline;
--  Returns the cache control data specified for the request

function Is_Supported
  (D        : Data;
   Encoding : Messages.Content_Encoding) return Boolean;
--  Returns True if the content encoding scheme is supported by the client

function Preferred_Coding  (D : Data) return Messages.Content_Encoding;
--  Returns supported by AWS coding preferred by client from the
--  Accept-Coding header.

function Upgrade           (D : Data) return String with Inline;
--  Get value for "Upgrade:" header

function Sec_WebSocket_Key (D : Data) return String with Inline;
--  Get value for "Sec-WebSocket-Key:" header

----------------------------------------
-- Cross-Origin Resource Sharing Headers --
----------------------------------------

function Origin (D : Data) return String with Inline;
--  Get value for "Origin:" header

function Access_Control_Request_Headers (D : Data) return String
  with Inline;
--  Get value for "Access-Control-Request-Headers:" header

function Access_Control_Request_Method (D : Data) return String with Inline;
--  Get value for "Access-Control-Request-Method:" header

----------------
-- Connection --
----------------

function Peername (D : Data) return String with Inline;
--  Returns the address of the peer (the IP address of the client computer)

function Socket   (D : Data) return Net.Socket_Type'Class with Inline;
--  Returns the socket used to transfer data between the client and
--  server.

function Socket   (D : Data) return Net.Socket_Access with Inline;
--  Returns the socket used to transfer data between the client and
--  server. Use Socket_Access to avoid memory allocation if we would need
--  socket access further.

----------
-- Data --
----------
```

(continues on next page)

(continued from previous page)

```
   function Is_Body_Uploaded       (D : Data) return Boolean with Inline;
   --  Returns True if the message body has been uploaded and False if not.
   --  The reason being that the body size is above Upload_Size_Limit.
   --  User can upload the file using AWS.Server.Get_Message_Body, the size
   --  being returned by Content_Length.

   function Multipart_Boundary     (D : Data) return String with Inline;
   --  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.

   function Binary_Data (D : Data) return Stream_Element_Array with Inline;
   --  Returns the binary data message content.
   --  Note that only the root part of a multipart/related message is returned.

   function Binary_Data (D : Data) return Unbounded_String;
   --  Returns the binary data message content in a Unbounded_String
   --  Note that only the root part of a multipart/related message is returned.

   function Binary_Data
     (D : Data)
      return not null access Resources.Streams.Memory.Stream_Type'Class;
   --  Returns the binary data message as a memory resource stream

   function Binary_Size (D : Data) return Stream_Element_Offset with Inline;
   --  Returns size of the binary data message content

   procedure Reset_Body_Index (D : Data) with Inline;
   --  Reset message body read position to the start

   procedure Read_Body
     (D      : Data;
      Buffer : out Stream_Element_Array;
      Last   : out Stream_Element_Offset)
   with Inline;
   --  Read a chunk of data from message body and put them into Buffer.
   --  Last is the index of the last item returned in Buffer.

   function End_Of_Body (D : Data) return Boolean with Inline;
   --  Returns true if there is no more data to read from the message body

   -----------------
   -- Attachments --
   -----------------

   function Attachments (D : Data) return AWS.Attachments.List with Inline;
   --  Returns the list of Attachments for the request

   -------------
   -- Session --
   -------------
```

(continues on next page)

(continued from previous page)

```
   function Has_Session            (D : Data) return Boolean with Inline;
   --  Returns true if a session ID has been received

   function Session_Private        (D : Data) return String with Inline;
   --  Returns the private Session ID for the request. Raises Constraint_Error
   --  if server's session support not activated.

   function Session                (D : Data) return Session.Id with Inline;
   --  Returns the Session ID for the request. Raises Constraint_Error if
   --  server's session support not activated.

   function Session_Created        (D : Data) return Boolean;
   --  Returns True if session was just created and is going to be sent to
   --  client.

   function Session_Timed_Out      (D : Data) return Boolean;
   --  Returns True if a previous session was timeout (even if a new session
   --  has been created).

   ----------
   -- SOAP --
   ----------

   function Is_SOAP    (D : Data) return Boolean with Inline;
   --  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 : Data) return String with Inline;
   --  Get value for "SOAPAction:" parameter. This is a standard header to
   --  support SOAP over HTTP protocol.

   function Payload    (D : Data) return String with Inline;
   --  Returns the XML Payload message. XML payload is the actual SOAP
   --  request. This is the root part of multipart/related SOAP message.

   function Payload    (D : Data) return Unbounded_String;
   --  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 : Data; Password : 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
```

(continues on next page)

(continued from previous page)

```
   --  (RFC 2617 3.2.2, 3.2.2.5),
   --  and Messages.S401 in case of authentication error.

   function Check_Digest (D : Data; Password : String) return Boolean;
   --  The same as above, but do not distinguish wrong requests and
   --  authentication errors.

   function Authorization_Mode     (D : Data) return Authorization_Type
     with Inline;
   --  Returns the type of the "Authorization:" parameter

   function Authorization_Name     (D : Data) return String with Inline;
   --  Returns "username" value in the "Authorization:" parameter

   function Authorization_URI      (D : Data) return String with Inline;
   --  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 : Data) return String with Inline;
   --  Returns "password" value in the "Authorization:" parameter

   function Authorization_Realm    (D : Data) return String with Inline;
   --  Returns "realm" value in the "Authorization:" parameter

   function Authorization_Nonce    (D : Data) return String with Inline;
   --  Returns "nonce" value in the "Authorization:" parameter

   function Authorization_NC       (D : Data) return String with Inline;
   --  Returns "nc" value in the "Authorization:" parameter

   function Authorization_CNonce   (D : Data) return String with Inline;
   --  Returns "cnonce" value in the "Authorization:" parameter

   function Authorization_QOP      (D : Data) return String with Inline;
   --  Retruns "qop" value in the "Authorization:" parameter

   function Authorization_Response (D : Data) return String with Inline;
   --  Returns "response" value in the "Authorization:" parameter

   function Authorization_Tail     (D : Data) return String with Inline;
   --  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
```

(continues on next page)

(continued from previous page)

```
   -- implementation removed
end AWS.Status;
```

13.74 AWS.Templates

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2012, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;
```

13.75 AWS.Translator

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2019, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

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

with AWS.Resources.Streams.Memory.ZLib;
with AWS.Utils;

package AWS.Translator is

   use Ada.Streams;
   use Ada.Strings.Unbounded;

   package ZL renames AWS.Resources.Streams.Memory.ZLib;

   ------------
   -- Base64 --
   ------------

   type Base64_Mode is (MIME, URL);
   --  Base64 encoding variants for encoding routines,
   --  RFC4648
```

(continues on next page)

(continued from previous page)

```
   --  MIME - section 4
   --  URL  - section 5

   subtype Base64_Common is Character with
     Static_Predicate => Base64_Common
       in 'A' .. 'Z' | 'a' .. 'z' | '0' .. '9' | '=';

   subtype Base64_String is String with
     Dynamic_Predicate =>
       (for all C of Base64_String =>
          C in Base64_Common | '+' | '-' | '_' | '/');

   subtype Base64_UString is Unbounded_String with
     Dynamic_Predicate =>
       (for all K in 1 .. Length (Base64_UString) =>
          Element (Base64_UString, K)
            in Base64_Common | '+' | '-' | '_' | '/');

   --
   --  Decoding does not have to have Base64_Mode parameter, because data
   --  coding easy detected automatically.

   procedure Base64_Encode
     (Data     : Unbounded_String;
      B64_Data : out Base64_UString;
      Mode     : Base64_Mode := MIME)
   with
     Post =>
       (Mode = MIME
          and then
        (for all K in 1 .. Length (B64_Data) =>
           Element (B64_Data, K) not in '-' | '_'))
      or else
        (Mode = URL
           and then
         (for all K in 1 .. Length (B64_Data) =>
           Element (B64_Data, K) not in '+' | '/'));

   function Base64_Encode
     (Data : Stream_Element_Array;
      Mode : Base64_Mode := MIME) return Base64_String
   with
     Post =>
       (Mode = MIME
          and then
        (for all C of Base64_Encode'Result => C not in '-' | '_'))
      or else
        (Mode = URL
           and then
         (for all C of Base64_Encode'Result => C not in '+' | '/'));
   --  Encode Data using the base64 algorithm
```

(continues on next page)

(continued from previous page)

```
   function Base64_Encode
     (Data : String; Mode : Base64_Mode := MIME) return Base64_String
   with
     Post =>
       (Mode = MIME
          and then
        (for all C of Base64_Encode'Result => C not in '-' | '_'))
      or else
       (Mode = URL
          and then
        (for all C of Base64_Encode'Result => C not in '+' | '/'));
   --  Same as above but takes a string as input

   procedure Base64_Decode
     (B64_Data : Base64_UString;
      Data     : out Unbounded_String);

   function Base64_Decode
     (B64_Data : Base64_String) return Stream_Element_Array;
   --  Decode B64_Data using the base64 algorithm

   function Base64_Decode (B64_Data : Base64_String) return String;

   --------
   -- QP --
   --------

   function QP_Decode (QP_Data : String) return String;
   --  Decode QP_Data using the Quoted Printable algorithm

   ------------------------------------
   -- String to Stream_Element_Array --
   ------------------------------------

   function To_String
     (Data : Stream_Element_Array) return String with Inline;
   --  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 : String) return Stream_Element_Array with Inline;
   --  Convert a String to a Stream_Element_Array

   function To_Stream_Element_Array
     (Data : String) return Utils.Stream_Element_Array_Access;
   --  As above but designed to be used for large objects

   function To_Unbounded_String
     (Data : Stream_Element_Array) return Unbounded_String;
   --  Convert a Stream_Element_Array to an Unbounded_String
```

(continues on next page)

(continued from previous page)

```
   -------------------------
   --  Compress/Decompress --
   -------------------------

   subtype Compression_Level is ZL.Compression_Level;

   Default_Compression : constant Compression_Level := ZL.Default_Compression;

   function Compress
     (Data   : Stream_Element_Array;
      Level  : Compression_Level                := Default_Compression;
      Header : 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   : Stream_Element_Array;
      Header : 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;
```

13.76 AWS.URL

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with Ada.Strings.Maps;
with Ada.Strings.Unbounded;

with AWS.Parameters;

package AWS.URL is

   use Ada;
   use Ada.Strings.Unbounded;

   --  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
```

(continues on next page)

(continued from previous page)

```
   --  URL supported:
   --
   --  http://user:pass@www.here.com:80/dir1/dir2/xyz.html?p=8&x=doh#anchor
   --   |                   |       | |          |       |         |
   --   protocol            host port path       file   parameters fragment
   --
   --                                 <--  pathname  -->

   type Object is private;

   URL_Error : exception;

   Default_FTP_Port   : constant := 21;
   Default_HTTP_Port  : constant := 80;
   Default_HTTPS_Port : constant := 443;

   function Parse
     (URL            : String;
      Check_Validity : Boolean := True;
      Normalize      : 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 : Object) return Boolean;
   --  Returns True if the URL is valid (does not reference directory above
   --  the Web root).

   function URL (URL : Object) return String;
   --  Returns full URL string, this can be different to the URL passed if it
   --  has been normalized.

   function Protocol_Name (URL : Object) return String;
   --  Returns "http" or "https" depending on the protocol used by URL

   function Host
     (URL : Object; IPv6_Brackets : Boolean := False) return String;
   --  Returns the hostname in IPv6 breakets if necessary

   function Port (URL : Object) return Positive;
   --  Returns the port as a positive

   function Port (URL : Object) return String;
   --  Returns the port as a string

   function Port_Not_Default (URL : Object) return String;
```

(continues on next page)

(continued from previous page)

```
   --  Returns the port image (preceded by character ':') if it is not the
   --  default port. Returns the empty string otherwise.

   function Abs_Path (URL : Object) return String;
   --  Returns the absolute path. This is the complete resource reference
   --  without the query part.

   function Query (URL : Object) 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 : Object) return String;
   --  Returns user name part of the URL. Returns the empty string if user was
   --  not specified.

   function Password (URL : Object) return String;
   --  Returns user's password part of the URL. Returns the empty string if
   --  password was not specified.

   function Server_Name
     (URL : Object; IPv6_Brackets : Boolean := False) return String
     renames Host;

   function Security (URL : Object) return Boolean;
   --  Returns True if it is a Secure HTTP (HTTPS) URL

   function Path (URL : Object) 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 : Object) 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 diretory into the filesystem.
   --  Parent and current directories are part of the path.

   function Parameters (URL : Object) 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 : Object) return String renames Abs_Path;

   function Pathname_And_Parameters (URL : Object) return String;
   --  Returns the pathname and the parameters. This is equivalent to:
   --  Pathname & Parameters.

   function Parameter
```

(continues on next page)

(continued from previous page)

```
     (URL : Object; Name : String; N : Positive := 1) return String
     with Inline;
   --  Returns the Nth value associated with Key into Table. Returns
   --  the emptry string if key does not exist.

   function Parameters (URL : Object) return AWS.Parameters.List with Inline;
   --  Return the parameter list associated with the URL

   function Fragment (URL : Object) return String with Inline;
   --  Return the part after the # sign (included)

   --
   --  URL Resolution
   --

   function Resolve (URL : Object; Base_URL : Object) return Object;
   --  Resolve an URL relative to a Base_URL. Uses RFC 3986, section 5.2
   --  algorithm.

   function Resolve (URL : String; Base_URL : String) return String;
   --  Resolve an URL relatively to a Base_URL. Same function as above, but
   --  working with Strings.

   --
   --  URL Encoding and Decoding
   --

   Parameters_Encoding_Set : constant Strings.Maps.Character_Set;
   --  Encoding set enought for HTTP parameters

   Default_Encoding_Set : constant Strings.Maps.Character_Set;
   --  Encoding set enought for all URL parts

   function Encode
     (Str          : String;
      Encoding_Set : Strings.Maps.Character_Set := Default_Encoding_Set)
      return String;
   --  Encode Str into a URL-safe form. Many characters are forbiden 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 : String) return String;
   --  This is the opposite of Encode above

   function Decode (Str : Unbounded_String) return Unbounded_String;

private
   -- implementation removed
end AWS.URL;
```

13.77 SOAP

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2020, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 := "3.0.0";
   --  Version number for this implementation

   No_SOAPAction : constant String := (1 => ASCII.NUL);
   --  Value used to specify that there was no SOAPAction specified

private
   -- implementation removed
end SOAP;
```

13.78 SOAP.Client

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2021, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with AWS.Client;
with SOAP.Message.Payload;
with SOAP.Message.Response;
with SOAP.WSDL.Schema;

package SOAP.Client is

   Not_Specified : String renames AWS.Client.No_Data;

   function Call
     (URL          : String;
      P            : Message.Payload.Object;
      SOAPAction   : String                     := No_SOAPAction;
      User         : String                     := Not_Specified;
      Pwd          : String                     := Not_Specified;
      Proxy        : String                     := Not_Specified;
      Proxy_User   : String                     := Not_Specified;
      Proxy_Pwd    : String                     := Not_Specified;
      Timeouts     : AWS.Client.Timeouts_Values := AWS.Client.No_Timeout;
      Asynchronous : Boolean                    := False;
```

(continues on next page)

(continued from previous page)

```
      Schema       : WSDL.Schema.Definition     := WSDL.Schema.Empty;
      HTTP_Version : AWS.HTTP_Protocol          := AWS.Client.HTTP_Default)
      return Message.Response.Object'Class
   with Pre => URL'Length > 0;
   --  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.
   --
   --  If Asynchronous is set to True the response from the server may be
   --  empty. In this specific case the success of the call depends on the
   --  HTTP status code.

   function Call
     (Connection   : AWS.Client.HTTP_Connection;
      SOAPAction   : String;
      P            : Message.Payload.Object;
      Asynchronous : Boolean := False;
      Schema       : WSDL.Schema.Definition := WSDL.Schema.Empty)
      return Message.Response.Object'Class
   with Pre => AWS.Client.Host (Connection)'Length > 0;
   --  Idem as above, but use an already opened connection

end SOAP.Client;
```

13.79 SOAP.Dispatchers

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2015, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;
with SOAP.WSDL.Schema;

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

   function Schema
     (Dispatcher : Handler;
      SOAPAction : String)
      return WSDL.Schema.Definition;
   --  Returns the schema for the given SOAPAction

   type SOAP_Callback is
     access function (SOAPAction : String;
                      Payload    : Message.Payload.Object;
```

(continues on next page)

(continued from previous page)

```
                     Request    : 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 : Handler;
      SOAPAction : String;
      Payload    : Message.Payload.Object;
      Request    : AWS.Status.Data)
      return AWS.Response.Data is abstract;
   --  This dispatch function is called for SOAP requests

   function Dispatch_HTTP
     (Dispatcher : Handler;
      Request    : AWS.Status.Data)
      return AWS.Response.Data is abstract;
   --  This dispatch function is called for standard HTTP requests

private
   -- implementation removed
end SOAP.Dispatchers;
```

13.80 SOAP.Dispatchers.Callback

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2003-2015, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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 SOAP Callback procedures

with SOAP.Message;
with SOAP.WSDL.Schema;

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.

   overriding function Schema
     (Dispatcher : Handler;
      SOAPAction : String)
      return WSDL.Schema.Definition;

   function Create
     (HTTP_Callback : AWS.Response.Callback;
      SOAP_Callback : Dispatchers.SOAP_Callback;
      Schema        : WSDL.Schema.Definition :=
                        WSDL.Schema.Empty) return Handler;
```

(continues on next page)

(continued from previous page)

```
   --  Build a dispatcher for the specified callback

private
   -- implementation removed
end SOAP.Dispatchers.Callback;
```

13.81 SOAP.Message

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2015, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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;
with SOAP.WSDL.Schema;

package SOAP.Message is

   use Ada.Strings.Unbounded;

   type Object is tagged private;

   function XML_Image
     (M      : Object;
      Schema : WSDL.Schema.Definition := WSDL.Schema.Empty)
      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 : Object'Class) return SOAP.Name_Space.Object;
   --  Returns message Namespace
```

(continues on next page)

(continued from previous page)

```
   function Wrapper_Name (M : Object'Class) return String;
   --  Returns wrapper name

   function Parameters   (M : Object'Class) return SOAP.Parameters.List;
   --  Returns the parameter

   procedure Set_Name_Space
     (M  : in out Object'Class;
      NS : SOAP.Name_Space.Object);
   --  Set message's Namespace

   procedure Set_Wrapper_Name
     (M     : in out Object'Class;
      Name  : String);
   --  Set message's wrapper name

   procedure Set_Parameters
     (M     : in out Object'Class;
      P_Set : SOAP.Parameters.List);
   --  Set message's parameters

private
   -- implementation removed
end SOAP.Message;
```

13.82 SOAP.Message.XML

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2015, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with AWS.Client;

with SOAP.Message.Payload;
with SOAP.Message.Response;
with SOAP.WSDL.Schema;

package SOAP.Message.XML is

   SOAP_Error : exception renames SOAP.SOAP_Error;

   function Load_Payload
     (XML      : aliased String;
      Envelope : Boolean := True;
      Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty)
      return Message.Payload.Object;
   --  Build a Payload object by parsing the XML payload string.
   --  If Envelope is False, the message could consists only from body
   --  with arbitrary named root tag without mandatory SOAP Envelope wrapper.

   function Load_Payload
```

(continues on next page)

(continued from previous page)

```
      (XML      : Unbounded_String;
       Envelope : Boolean := True;
       Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty)
       return Message.Payload.Object;
   --  Build a Payload object by parsing the XML payload string

   function Load_Response
     (Connection : AWS.Client.HTTP_Connection;
      Envelope   : Boolean := True;
      Schema     : WSDL.Schema.Definition := WSDL.Schema.Empty)
      return Message.Response.Object'Class;
   --  Build a Response object (either a standard response or an error
   --  response) by parsing the HTTP client connection output.
   --  If Envelope is False, the message could consists only from body
   --  with arbitrary named root tag without mandatory SOAP Envelope wrapper.

   function Load_Response
     (XML      : aliased String;
      Envelope : Boolean := True;
      Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty)
      return Message.Response.Object'Class;
   --  Build a Response object (either a standard response or an error
   --  response) by parsing the XML response string.
   --  If Envelope is False, the message could consists only from body
   --  with arbitrary named root tag without mandatory SOAP Envelope wrapper.

   function Load_Response
     (XML      : Unbounded_String;
      Envelope : Boolean := True;
      Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty)
      return Message.Response.Object'Class;
   --  As above but using an Unbounded_String

   function Image
     (O      : Object'Class;
      Schema : WSDL.Schema.Definition := WSDL.Schema.Empty) return String;
   --  Returns XML representation of object O

   function Image
     (O      : Object'Class;
      Schema : WSDL.Schema.Definition :=
                 WSDL.Schema.Empty) return Unbounded_String;
   --  Idem as above but returns an Unbounded_String instead of a String

end SOAP.Message.XML;
```

13.83 SOAP.Parameters

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2000-2024, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

with Ada.Strings.Unbounded;

with SOAP.Types;

package SOAP.Parameters is

   use Ada.Strings.Unbounded;

   Data_Error : exception renames Types.Data_Error;

   Max_Parameters : constant := 50;
   --  This is the maximum number of parameters supported by this
   --  implementation.

   type List is private;

   function Argument_Count (P : List) return Natural with
     Post => Argument_Count'Result <= Max_Parameters;
   --  Returns the number of parameters in P
```

(continues on next page)

(continued from previous page)

```
   function Argument (P : List; Name : String) return Types.Object'Class;
   --  Returns parameters named Name in P. Raises Types.Data_Error if not
   --  found.

   function Argument (P : List; N : Positive) return Types.Object'Class;
   --  Returns Nth parameters in P. Raises Types.Data_Error if not found

   function Exist (P : List; Name : String) return Boolean;
   --  Returns True if parameter named Name exist in P and False otherwise

   function Get (P : List; Name : String) return Types.Long with Inline;
   --  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 : List; Name : String) return Integer with Inline;
   --  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 : List; Name : String) return Types.Short with Inline;
   --  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 : List; Name : String) return Types.Byte with Inline;
   --  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 : List; Name : String) return Float with Inline;
   --  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 : List; Name : String) return Long_Float with Inline;
   --  Returns parameter named Name in P as a Double value. Raises
   --  Types.Data_Error if this parameter does not exist or is not a Double.

   function Get (P : List; Name : String) return Types.Decimal with Inline;
   --  Returns parameter named Name in P as a Decimal value. Raises
   --  Types.Data_Error if this parameter does not exist or is not a Decimal.

   function Get (P : List; Name : String) return String with Inline;
   --  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 : List; Name : String) return Unbounded_String with Inline;
   --  Idem as above, but return an Unbounded_String

   function Get (P : List; Name : String) return Boolean with Inline;
   --  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 : List; Name : String) return Types.Local_Date_Time with Inline;
   --  Returns parameter named Name in P as a Time value. Raises
```

(continues on next page)

(continued from previous page)

```
   --  Types.Data_Error if this parameter does not exist or is not a Data_Time.

   function Get
     (P : List; Name : String) return Types.Local_Date with Inline;
   --  Returns parameter named Name in P as a Time value. Raises
   --  Types.Data_Error if this parameter does not exist or is not a Date.

   function Get
     (P : List; Name : String) return Types.Local_Time with Inline;
   --  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 : List; Name : String) return Duration with Inline;
   --  Returns parameter named Name in P as a Duration value. Raises
   --  Types.Data_Error if this parameter does not exist or is not a Duration.

   function Get (P : List; Name : String) return Types.Unsigned_Long
     with Inline;
   --  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 : List; Name : String) return Types.Unsigned_Int
     with Inline;
   --  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 : List; Name : String) return Types.Unsigned_Short
     with Inline;
   --  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 : List; Name : String) return Types.Unsigned_Byte
     with Inline;
   --  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 : List; Name : String) return Types.SOAP_Base64 with Inline;
   --  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 : List; Name : String) return Types.SOAP_Record with Inline;
   --  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 : List; Name : String) return Types.SOAP_Array with Inline;
   --  Returns parameter named Name in P as a SOAP Array value. Raises
```

(continues on next page)

(continued from previous page)

```
   --  Types.Data_Error if this parameter does not exist or is not a SOAP
   --  Array.

   ------------------
   -- Constructors --
   ------------------

   function "&" (P : List; O : Types.Object'Class) return List with
     Post => Argument_Count ("&"'Result) = Argument_Count (P) + 1;

   function "+" (O : Types.Object'Class) return List with
     Post => Argument_Count ("+"'Result) = 1;

   ----------------
   -- Validation --
   ----------------

   procedure Check (P : List; N : Natural);
   --  Checks that there is exactly N parameters or raise Types.Data_Error

   procedure Check_Integer (P : List; Name : String);
   --  Checks that parameter named Name exist and is an Integer value

   procedure Check_Float (P : List; Name : String);
   --  Checks that parameter named Name exist and is a Float value

   procedure Check_Boolean (P : List; Name : String);
   --  Checks that parameter named Name exist and is a Boolean value

   procedure Check_Time_Instant (P : List; Name : String);
   --  Checks that parameter named Name exist and is a Time_Instant value

   procedure Check_Duration (P : List; Name : String);
   --  Checks that parameter named Name exists and is a Duration value

   procedure Check_Base64 (P : List; Name : String);
   --  Checks that parameter named Name exist and is a Base64 value

   procedure Check_Null (P : List; Name : String);
   --  Checks that parameter named Name exist and is a Null value

   procedure Check_Record (P : List; Name : String);
   --  Checks that parameter named Name exist and is a Record value

   procedure Check_Array (P : List; Name : String);
   --  Checks that parameter named Name exist and is an Array value

private
   -- implementation removed
end SOAP.Parameters;
```

13.84 SOAP.Types

```
------------------------------------------------------------------------------
--                              Ada Web Server                              --
--                                                                          --
--                     Copyright (C) 2001-2024, AdaCore                     --
--                                                                          --
--  This library is free software;  you can redistribute it and/or modify   --
--  it under terms of the  GNU General Public License  as published by the  --
--  Free Software  Foundation;  either version 3,  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.                     --
--                                                                          --
--  As a special exception under Section 7 of GPL version 3, you are        --
--  granted additional permissions described in the GCC Runtime Library     --
--  Exception, version 3.1, as published by the Free Software Foundation.   --
--                                                                          --
--  You should have received a copy of the GNU General Public License and   --
--  a copy of the GCC Runtime Library Exception along with this program;    --
--  see the files COPYING3 and COPYING.RUNTIME respectively.  If not, see   --
--  <http://www.gnu.org/licenses/>.                                         --
--                                                                          --
--  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.                                     --
------------------------------------------------------------------------------

pragma Ada_2012;

--  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. Finaly add support for this type in SOAP.Message.XML. Add this type
```

(continues on next page)

(continued from previous page)

```
--     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;
with SOAP.WSDL.Schema;

package SOAP.Types is

   use Ada;
   use Ada.Strings.Unbounded;

   subtype Encoding_Style is WSDL.Schema.Encoding_Style;
   --  SOAP encoding style for the entities

   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.

   Empty_Object_Set : constant Object_Set;

   function Image (O : Object) return String;
   --  Returns O value image

   function Is_Empty (O : Object) return Boolean;
   --  Returns True if the object is empty Array, Empty Record or null value

   procedure XML_Image
     (O        : Object;
      Result   : in out Unbounded_String;
      Encoding : Encoding_Style := WSDL.Schema.Encoded;
      Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty);
   --  Returns O value encoded for use by the Payload object or Response
   --  object. The generated characters are appened to Result.
```

(continues on next page)

(continued from previous page)

```
   function XML_Image (O : Object'Class) return String;
   --  Returns O value encoded for use by the Payload object or Response
   --  object.

   function XML_Type (O : Object) return String;
   --  Returns the XML type for the object

   function Name (O : Object'Class) return String;
   --  Returns name for object O

   function Type_Name (O : Object'Class) return String;
   --  Returns the type name for object O

   function "+" (O : Object'Class) return Object_Safe_Pointer;
   --  Allocate an object into the heap and return a safe pointer to it

   function "-" (O : 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 (O : Composite) return Object_Set;

   overriding function Is_Empty (O : Composite) return Boolean;

   --------------
   -- Any Type --
   --------------

   XML_Any_Type : aliased constant String := "xsd:anyType";

   type XSD_Any_Type is new Object with private;

   overriding function  XML_Type  (O : XSD_Any_Type) return String;
   overriding function  Image     (O : XSD_Any_Type) return String;
   overriding procedure XML_Image
     (O        : XSD_Any_Type;
      Result   : in out Unbounded_String;
      Encoding : Encoding_Style := WSDL.Schema.Encoded;
      Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty);

   function Any
     (V         : Object'Class;
      Name      : String := "item";
      Type_Name : String := "";
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Any_Type;
```

(continues on next page)

(continued from previous page)

```
function V (O : XSD_Any_Type) return Object_Access;

-------------
-- Any URI --
-------------

subtype Any_URI is String;

XML_Any_URI : aliased constant String := "xsd:anyURI";

type XSD_Any_URI is new Object with private;

overriding function  Image (O : XSD_Any_URI) return String;

function AnyURI
  (V         : String;
   Name      : String := "item";
   Type_Name : String := "";
   NS        : Name_Space.Object := Name_Space.No_Name_Space)
   return XSD_Any_URI;

function V (O : XSD_Any_URI) return String;

function V (O : XSD_Any_URI) return Unbounded_String;

-----------
-- Array --
-----------

XML_Array     : constant String := "soapenc:Array";
XML_Undefined : aliased constant String := "xsd:ur-type";

type SOAP_Array is new Composite with private;

overriding function Image (O : SOAP_Array) return String;
overriding procedure XML_Image
  (O        : SOAP_Array;
   Result   : in out Unbounded_String;
   Encoding : Encoding_Style := WSDL.Schema.Encoded;
   Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty);

function A
  (V         : Object_Set;
   Name      : String;
   Type_Name : String := "";
   NS        : Name_Space.Object := Name_Space.No_Name_Space)
   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 : SOAP_Array) return Natural;
```

(continues on next page)

(continued from previous page)

```
   --  Returns the number of item into the array

   function V (O : SOAP_Array; N : Positive) return Object'Class;
   --  Returns SOAP_Array item at position N

   ----------
   --  Set --
   ----------

   type SOAP_Set is new Composite with private;
   --  A set is like an array but to record multi-occurence of parameters. The
   --  SOAP message does not contain the enclosing SOAP array XML tag.

   overriding function Image (O : SOAP_Set) return String;
   overriding procedure XML_Image
     (O        : SOAP_Set;
      Result   : in out Unbounded_String;
      Encoding : Encoding_Style := WSDL.Schema.Encoded;
      Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty);

   function Set
     (V         : Object_Set;
      Name      : String;
      Type_Name : String := "";
      NS        : SOAP.Name_Space.Object := SOAP.Name_Space.No_Name_Space)
      return SOAP_Set;
   --  Type_Name of the array's elements, if not specified it will be computed
   --  based on element's name.

   ------------
   -- Base64 --
   ------------

   XML_Base64        : aliased constant String := "soapenc:base64";
   XML_Base64_Binary : aliased constant String := "xsd:base64Binary";

   type SOAP_Base64 is new Scalar with private;

   overriding function Image (O : SOAP_Base64) return String;

   function B64
     (V         : String;
      Name      : String := "item";
      Type_Name : String := XML_Base64;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return SOAP_Base64;

   function V (O : SOAP_Base64) return String;

   -------------
   -- Boolean --
   -------------
```

(continues on next page)

(continued from previous page)

```
   XML_Boolean : aliased constant String := "xsd:boolean";

   type XSD_Boolean is new Scalar with private;

   overriding function Image (O : XSD_Boolean) return String;

   function B
     (V         : Boolean;
      Name      : String := "item";
      Type_Name : String := XML_Boolean;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Boolean;

   function V (O : 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;

   overriding function Image (O : XSD_Byte) return String;

   function B
     (V         : Byte;
      Name      : String := "item";
      Type_Name : String := XML_Byte;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Byte;

   function V (O : XSD_Byte) return Byte;

   ------------
   -- Double --
   ------------

   XML_Double : aliased constant String := "xsd:double";

   type XSD_Double is new Scalar with private;

   overriding function Image (O : XSD_Double) return String;

   function D
     (V         : Long_Float;
      Name      : String := "item";
      Type_Name : String := XML_Double;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
```

(continues on next page)

(continued from previous page)

```
      return XSD_Double;

   function V (O : XSD_Double) return Long_Float;

   -----------
   -- Float --
   -----------

   XML_Float : aliased constant String := "xsd:float";

   type XSD_Float is new Scalar with private;

   overriding function Image (O : XSD_Float) return String;

   function F
     (V         : Float;
      Name      : String := "item";
      Type_Name : String := XML_Float;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Float;

   function V (O : XSD_Float) return Float;

   -------------
   -- Decimal --
   -------------

   --  Ensure a supported value is selected depending on the platform
   --  being 32bit or 64bit.

   Max_Digits : constant := (if Standard'Max_Integer_Size >= 128
                             then 38
                             else 18);

   D_Delta    : constant := (if Standard'Max_Integer_Size >= 128
                             then 8
                             else 4);

   type Decimal is delta 10.0 ** (-D_Delta) digits Max_Digits;

   XML_Decimal : aliased constant String := "xsd:decimal";

   type XSD_Decimal is new Scalar with private;

   overriding function Image (O : XSD_Decimal) return String;

   function D
     (V         : Decimal;
      Name      : String := "item";
      Type_Name : String := XML_Decimal;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Decimal;
```

(continues on next page)

(continued from previous page)

```
function V (O : XSD_Decimal) return Decimal;

-------------
-- Integer --
-------------

XML_Int : aliased constant String := "xsd:int";

type XSD_Integer is new Scalar with private;

overriding function Image (O : XSD_Integer) return String;

function I
  (V         : Integer;
   Name      : String := "item";
   Type_Name : String := XML_Int;
   NS        : Name_Space.Object := Name_Space.No_Name_Space)
   return XSD_Integer;

function V (O : 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;

overriding function Image (O : XSD_Long) return String;

function L
  (V         : Long;
   Name      : String := "item";
   Type_Name : String := XML_Long;
   NS        : Name_Space.Object := Name_Space.No_Name_Space)
   return XSD_Long;

function V (O : XSD_Long) return Long;

----------
-- Null --
----------

type XSD_Null is new Scalar with private;

overriding procedure XML_Image
  (O        : XSD_Null;
   Result   : in out Unbounded_String;
```

(continues on next page)

(continued from previous page)

```
      Encoding : Encoding_Style := WSDL.Schema.Encoded;
      Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty);

   function N
     (Name      : String;
      Type_Name : String;
      NS        : SOAP.Name_Space.Object := SOAP.Name_Space.No_Name_Space)
      return XSD_Null;

   overriding function Is_Empty (O : XSD_Null) return Boolean;

   ------------
   -- Record --
   ------------

   type SOAP_Record is new Composite with private;

   overriding function Image (O : SOAP_Record) return String;
   overriding procedure XML_Image
     (O        : SOAP_Record;
      Result   : in out Unbounded_String;
      Encoding : Encoding_Style := WSDL.Schema.Encoded;
      Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty);

   function R
     (V         : Object_Set;
      Name      : String;
      Type_Name : String := "";
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      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 : SOAP_Record; Name : String) return Object'Class;
   --  Returns SOAP_Record field named Name

   function V (O : SOAP_Record; Name : String) return Object_Set;
   --  Returns SOAP_Record fields named Name

   function Exists (O : SOAP_Record; Field_Name : String) return Boolean;
   --  Returns True if the record O constains Field_Name

   -----------
   -- 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;
```

(continues on next page)

(continued from previous page)

```
   overriding function Image (O : XSD_Short) return String;

   function S
     (V         : Short;
      Name      : String := "item";
      Type_Name : String := XML_Short;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Short;

   function V (O : XSD_Short) return Short;

   ------------
   -- String --
   ------------

   XML_String : aliased constant String := "xsd:string";

   type XSD_String is new Scalar with private;

   overriding function Image (O : XSD_String) return String;

   function S
     (V         : String;
      Name      : String := "item";
      Type_Name : String := XML_String;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_String;

   function S
     (V         : Unbounded_String;
      Name      : String := "item";
      Type_Name : String := XML_String;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_String;

   function V (O : XSD_String) return String;

   function V (O : XSD_String) return Unbounded_String;

   ----------------------
   -- Normalized String --
   ----------------------

   subtype Normalized_String is String;

   XML_Normalized_String : aliased constant String := "xsd:normalizedString";

   type XSD_Normalized_String is new Scalar with private;

   overriding function Image (O : XSD_Normalized_String) return String;

   function NS
```

(continues on next page)

(continued from previous page)

```
   (V         : String;
    Name      : String := "item";
    Type_Name : String := XML_Normalized_String;
    NS        : Name_Space.Object := Name_Space.No_Name_Space)
    return XSD_Normalized_String;

function NS
   (V         : Unbounded_String;
    Name      : String := "item";
    Type_Name : String := XML_Normalized_String;
    NS        : Name_Space.Object := Name_Space.No_Name_Space)
    return XSD_Normalized_String;

function V (O : XSD_Normalized_String) return String;

function V (O : XSD_Normalized_String) return Unbounded_String;

-----------
-- Token --
-----------

subtype Token is String;

XML_Token : aliased constant String := "xsd:token";

type XSD_Token is new Scalar with private;

overriding function Image (O : XSD_Token) return String;

function T
   (V         : String;
    Name      : String := "item";
    Type_Name : String := XML_Token;
    NS        : Name_Space.Object := Name_Space.No_Name_Space)
    return XSD_Token;

function T
   (V         : Unbounded_String;
    Name      : String := "item";
    Type_Name : String := XML_Token;
    NS        : Name_Space.Object := Name_Space.No_Name_Space)
    return XSD_Token;

function V (O : XSD_Token) return String;

function V (O : XSD_Token) return Unbounded_String;

-----------------
-- TimeInstant --
-----------------

subtype Local_Date_Time is Calendar.Time;
```

(continues on next page)

(continued from previous page)

```
   --  All times are local time. This means that a timeInstant is always
   --  converted to a local time for the running host.

   XML_Time_Instant : aliased constant String := "xsd:timeInstant";
   XML_Date_Time    : aliased constant String := "xsd:dateTime";

   type XSD_Time_Instant is new Scalar with private;

   overriding function Image (O : XSD_Time_Instant) return String;

   function T
     (V         : Local_Date_Time;
      Name      : String := "item";
      Type_Name : String := XML_Time_Instant;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Time_Instant;

   function V (O : XSD_Time_Instant) return Local_Date_Time;
   --  Returns a GMT date and time

   ----------
   -- Date --
   ----------

   type Local_Date is new Calendar.Time;

   XML_Date : aliased constant String := "xsd:date";

   type XSD_Date is new Scalar with private;

   overriding function Image (O : XSD_Date) return String;

   function TD
     (V         : Local_Date;
      Name      : String := "item";
      Type_Name : String := XML_Date;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Date;

   function V (O : XSD_Date) return Local_Date;
   --  Returns a date

   ----------
   -- Time --
   ----------

   type Local_Time is new Calendar.Time;

   XML_Time : aliased constant String := "xsd:time";

   type XSD_Time is new Scalar with private;
```

(continues on next page)

(continued from previous page)

```
   overriding function Image (O : XSD_Time) return String;

   function TT
     (V         : Local_Time;
      Name      : String := "item";
      Type_Name : String := XML_Time;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Time;

   function V (O : XSD_Time) return Local_Time;
   --  Returns a GMT time

   --------------
   -- Duration --
   --------------

   XML_Duration : aliased constant String := "xsd:duration";

   type XSD_Duration is new Scalar with private;

   overriding function Image (O : XSD_Duration) return String;

   function D
     (V         : Duration;
      Name      : String := "item";
      Type_Name : String := XML_Duration;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Duration;

   function V (O : XSD_Duration) return Duration;
   --  Returns the Ada duration

   -------------------
   -- 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;

   overriding function Image (O : XSD_Unsigned_Long) return String;

   function UL
     (V         : Unsigned_Long;
      Name      : String := "item";
      Type_Name : String := XML_Unsigned_Long;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Unsigned_Long;

   function V (O : XSD_Unsigned_Long) return Unsigned_Long;
```

(continues on next page)

(continued from previous page)

```
   ------------------
   -- 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;

   overriding function Image (O : XSD_Unsigned_Int) return String;

   function UI
     (V         : Unsigned_Int;
      Name      : String := "item";
      Type_Name : String := XML_Unsigned_Int;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Unsigned_Int;

   function V (O : 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;

   overriding function Image (O : XSD_Unsigned_Short) return String;

   function US
     (V         : Unsigned_Short;
      Name      : String := "item";
      Type_Name : String := XML_Unsigned_Short;
      NS        : Name_Space.Object := Name_Space.No_Name_Space)
      return XSD_Unsigned_Short;

   function V (O : 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;
```

(continues on next page)

(continued from previous page)

```
overriding function Image (O : XSD_Unsigned_Byte) return String;

function UB
  (V         : Unsigned_Byte;
   Name      : String := "item";
   Type_Name : String := XML_Unsigned_Byte;
   NS        : Name_Space.Object := Name_Space.No_Name_Space)
   return XSD_Unsigned_Byte;

function V (O : XSD_Unsigned_Byte) return Unsigned_Byte;

-----------------
-- Enumeration --
-----------------

type SOAP_Enumeration is new Scalar with private;

overriding function Image      (O : SOAP_Enumeration) return String;
overriding procedure XML_Image
  (O        : SOAP_Enumeration;
   Result   : in out Unbounded_String;
   Encoding : Encoding_Style := WSDL.Schema.Encoded;
   Schema   : WSDL.Schema.Definition := WSDL.Schema.Empty);

function E
  (V         : String;
   Type_Name : String;
   Name      : String := "item";
   NS        : Name_Space.Object := Name_Space.No_Name_Space)
   return SOAP_Enumeration;

function V (O : 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 : 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 : Object'Class) return Long;
--  Returns O value as a Long. Raises Data_Error if O is not a SOAP
--  Long.

function Get (O : Object'Class) return Integer;
--  Returns O value as an Integer. Raises Data_Error if O is not a SOAP
```

(continues on next page)

(continued from previous page)

```
   --  Integer.

   function Get (O : Object'Class) return Short;
   --  Returns O value as a Short. Raises Data_Error if O is not a SOAP
   --  Short.

   function Get (O : Object'Class) return Byte;
   --  Returns O value as a Byte. Raises Data_Error if O is not a SOAP
   --  Byte.

   function Get (O : Object'Class) return Float;
   --  Returns O value as a Long_Float. Raises Data_Error if O is not a SOAP
   --  Float.

   function Get (O : Object'Class) return Long_Float;
   --  Returns O value as a Long_Long_Float. Raises Data_Error if O is not a
   --  SOAP Double.

   function Get (O : Object'Class) return Decimal;
   --  Returns O value as a Decimal. Raises Data_Error if O is not a SOAP
   --  Decimal.

   function Get (O : Object'Class) return String;
   --  Returns O value as a String. Raises Data_Error if O is not a SOAP
   --  String.

   function Get (O : Object'Class) return Unbounded_String;
   --  As above but returns an Unbounded_String

   function Get (O : Object'Class) return Boolean;
   --  Returns O value as a Boolean. Raises Data_Error if O is not a SOAP
   --  Boolean.

   function Get (O : Object'Class) return Local_Date_Time;
   --  Returns O value as a Time. Raises Data_Error if O is not a SOAP
   --  Date_Time.

   function Get (O : Object'Class) return Local_Date;
   --  Returns O value as a Time. Raises Data_Error if O is not a SOAP
   --  Date.

   function Get (O : Object'Class) return Local_Time;
   --  Returns O value as a Time. Raises Data_Error if O is not a SOAP
   --  Time.

   function Get (O : Object'Class) return Duration;
   --  Returns O value as a Duration. Raises Data_Error if O is not a SOAP
   --  Duration.

   function Get (O : Object'Class) return Unsigned_Long;
   --  Returns O value as a Unsigned_Long. Raises Data_Error if O is not a SOAP
   --  Unsigned_Long.
```

(continues on next page)

(continued from previous page)

```
   function Get (O : 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 : 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 : 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 : 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 : 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 : 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 : Name_Space.Object);
   --  Set the name space for object O

   function Name_Space (O : Object'Class) return Name_Space.Object;
   --  Returns name space associated with object O

   procedure Rename (O : in out Object'Class; Name : String);
   --  Set the name to the object

   function Rename (O : Object'Class; Name : String) return Object'Class;
   --  Return the same object with changed name

private
   -- implementation removed
end SOAP.Types;
```

Copyright (C) 2000, Pascal Obry

Copyright (C) 2001, Pascal Obry, Dmitriy Anisimkov

Copyright (C) 2002-2013, 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.

INDEX

A

ABORTABLE_V, 97
ACCEPT_QUEUE_SIZE, 97
Accept_Queue_Size, 18
ACCEPTOR_LENGTH, 97
ACTIVITY_COUNTER_V, 97
ACTIVITY_TIME_STAMP_V, 97
ada2wsdl, 69
ada2wsdl limitations, 78
ADMIN, 97
Admin_Password, 18
Admin_URI, 11, 18
Ajax, 47
authentication, 26
AWS.Attachments, 107
AWS.Client, 112
AWS.Client.Hotplug, 124
AWS.Communication, 126
AWS.Communication.Client, 128
AWS.Communication.Server, 129
AWS.Config, 131
AWS.Config.Ini, 141
AWS.Config.Set, 142
AWS.Containers.Tables, 151
AWS.Cookie, 155
AWS.Default, 159
AWS.Dispatchers, 163
AWS.Dispatchers.Callback, 165
AWS.Exceptions, 166
AWS.Headers, 168
AWS.Headers.Values, 171
aws.ini, 18
AWS.Jabber, 174
AWS.LDAP.Client, 175
AWS.Log, 182
AWS.Messages, 186
AWS.MIME, 194
AWS.Net, 198
AWS.Net.Buffered, 207
AWS.Net.Log, 210
AWS.Net.Log.Callbacks, 212
AWS.Net.SSL, 214
AWS.Net.SSL.Certificate, 222
AWS.Net.Std, 3
AWS.Net.WebSocket, 225
AWS.Net.WebSocket.Registry, 231
AWS.Net.WebSocket.Registry.Control, 235
AWS.Parameters, 236
AWS.POP, 238
AWS.Resources, 243
AWS.Resources.Embedded, 246
AWS.Resources.Files, 248
AWS.Resources.Streams, 250
AWS.Resources.Streams.Disk, 252
AWS.Resources.Streams.Disk.Once, 254
AWS.Resources.Streams.Memory, 255
AWS.Resources.Streams.Memory.ZLib, 257
AWS.Resources.Streams.Pipe, 259
AWS.Response, 261
AWS.Response.Set, 270
AWS.Server, 275
AWS.Server.Hotplug, 281
AWS.Server.Log, 283
AWS.Server.Push, 286
AWS.Server.Status, 292
AWS.Services.Callbacks, 294
AWS.Services.Directory, 295
AWS.Services.Dispatchers, 298
AWS.Services.Dispatchers.Linker, 300
AWS.Services.Dispatchers.Method, 301
AWS.Services.Dispatchers.URI, 303
AWS.Services.Dispatchers.Virtual_Host, 305
AWS.Services.Download, 307
AWS.Services.Page_Server, 309
AWS.Services.Split_Pages, 311
AWS.Services.Split_Pages.Alpha, 314
AWS.Services.Split_Pages.Alpha.Bounded, 316
AWS.Services.Split_Pages.Alpha.Uniform, 318
AWS.Services.Split_Pages.Alpha.Uniform.Alpha, 320
AWS.Services.Split_Pages.Alpha.Uniform.Overlapping, 322
AWS.Services.Transient_Pages, 323
AWS.Services.Web_Block, 325

AWS.Services.Web_Block.Context, 326
AWS.Services.Web_Block.Registry, 328
AWS.Session, 332
AWS.SMTP, 337
AWS.SMTP.Client, 340
AWS.Status, 345
AWS.Templates, 353
AWS.Translator, 354
AWS.URL, 358
aws_action_clear.tjs, 47
aws_action_replace.tjs, 47
awsascb, 84
awsres, 96

B

basic, 26
Building, 4
Building resources, 95

C

CA, 35
CA certificate, 34
Callback, 11, 12, 41, 165
Callback procedure, 12
CASE_SENSITIVE_PARAMETERS, 97
Case_Sensitive_Parameters, 11, 18
certificate, 33
Certificate Authority, 35
Check_Certificate, 20
CHECK_URL_VALIDITY, 97
Check_URL_Validity, 19
CIPHER_PRIORITIES, 99
Cipher_Priorities (*string*), 19
CLEANER_CLIENT_DATA_TIMEOUT, 97
Cleaner_Client_Data_Timeout, 19
CLEANER_CLIENT_HEADER_TIMEOUT, 97
Cleaner_Client_Header_Timeout, 19
CLEANER_SERVER_RESPONSE_TIMEOUT, 97
Cleaner_Server_Response_Timeout, 19
CLEANER_WAIT_FOR_CLIENT_TIMEOUT, 97
Cleaner_Wait_For_Client_Timeout, 19
Client, 27, 78
client, 39
client certificate, 34, 36
client HTTP, 38
client HTTPS, 39
Client protocol, 38
Client_Certificate (*string*), 18
Close_On_Exec, 19
Code generator, 82
Communication, 27, 28
Config_Directory, 19
Configuration options, 18
Cookies, 25
CRL, 19, 36
CRL_File, 19
cross-platforms, 4
CURRENT_CONNECTIONS, 98

D

digest, 26
Directory browser, 41
Directory_Browser_Page, 19
Disable_Program_Ini, 19
dispatcher, 41–43
dispatcher API, 165
Dispatchers, 41
Dispatchers callback, 41
Dispatchers Linker, 42
Dispatchers method, 41
Dispatchers SOAP, 43
Dispatchers Timer, 42
Dispatchers Transient pages, 42
Dispatchers URI, 42
Dispatchers virtual host, 42
Distributing, 14
Down_Image, 19
Download Manager, 44
draft 302, 103

E

ERROR_LOG, 98
Error_Log_Activated, 19
ERROR_LOG_FILE, 98
ERROR_LOG_FILENAME_PREFIX, 98
Error_Log_Filename_Prefix, 20
ERROR_LOG_SPLIT_MODE, 98
Error_Log_Split_Mode, 20
exception handler, 37
Exceptions, 166
Exceptions handler, 166
Exchange_Certificate, 20

F

File upload, 27
FORCE_CLIENT_DATA_TIMEOUT, 98
Force_Client_Data_Timeout, 20
FORCE_CLIENT_HEADER_TIMEOUT, 98
Force_Client_Header_Timeout, 20
FORCE_SERVER_RESPONSE_TIMEOUT, 98
Force_Server_Response_Timeout, 20
FORCE_WAIT_FOR_CLIENT_TIMEOUT, 98
Force_Wait_For_Client_Timeout, 20
Form parameters, 13
FREE_SLOTS_KEEP_ALIVE_LIMIT, 98
Free_Slots_Keep_Alive_Limit, 20

G

GNAT, 3
GNU/Ada, 3
GNUTLS, 3
GNUTLS build, 4

H

Hello world, 12
hotplug, 28
Hotplug_Port, 20
HTML File Upload, 101
HTTP Authentication, 103
HTTP declaration, 10
HTTP state, 25
HTTP/1.0, 101
HTTP/1.1, 102
HTTP2_Activated, 20
HTTPS, 32

I

IMAP/POP, 102
ini file, 18
Installing, 6

J

Jabber, 93
Jabber Binding, 174
Jabber message, 93
Jabber presence, 93

K

KEYS_M, 98

L

LDAP, 91
LDAP Binding, 175
LDAP Directory, 91
LibreSSL, 3
LibreSSL build, 4
Lightweight Directory Access Protocol, 91
LINE_STACK_SIZE, 98
Line_Stack_Size, 20
linker, 42
LOG, 98
Log.Flush, 31
Log.Start, 31
Log.Start_Error, 31
Log.Stop, 31
Log.Stop_Error, 31
Log_Activated, 20
Log_Extended_Fields, 21
LOG_FILE, 98
LOG_FILE_DIRECTORY, 98
Log_File_Directory, 21
LOG_FILENAME_PREFIX, 98
Log_Filename_Prefix, 21
LOG_MODE, 98
Log_Split_Mode, 21
LOGO, 98
Logo_Image, 21
logs, 31

M

MAX_CONCURRENT_DOWNLOAD, 98
Max_Concurrent_Download, 21
MAX_CONNECTION, 98
Max_Connection, 11, 21
Max_POST_Parameters, 21
Max_WebSocket, 21
Max_WebSocket_Handler, 21
method, 41
MIME, 102
MIME_Types, 21

O

OpenLDAP, 3
OpenSSL, 3
OpenSSL build, 4

P

Page server, 13, 43
pages, 43, 44
Parameters, 13
Parameters Get, 14
Parameters Get_Name, 14
Payload, 66
PEER_NAME_V, 98
PHASE_V, 99
POP, 88, 101
Port, 11
Post Office Protocol, 88
program_name.ini, 18
Push, 30

R

RECEIVE_TIMEOUT, 99
Receive_Timeout, 22
References, 101
Resources, 95
resources, 12
Retrieving e-mail, 88
REUSE_ADDRESS, 99
Reuse_Address, 22
Revocation, 36
RFC 0821, 101
RFC 1867, 101

RFC 1939, 101
RFC 1945, 101
RFC 2049, 102
RFC 2109, 102
RFC 2195, 102
RFC 2554, 102
RFC 2616, 102
RFC 2617, 103

S

Secure server, 32
SECURITY, 99
Security, 11
Security level, 36
SECURITY_MODE, 99
Security_Mode, 22
Self dependant, 95
Self-signed certificate, 34
Send, 30
Send_Buffer_Size, 22
SEND_TIMEOUT, 99
Send_Timeout, 22
Send_To, 30
Sending e-mail, 87
Sending message, 27
Server, 28, 79
server, 27
server certificate, 34, 35
Server Push, 30
Server_Certificate (*string*), 18
Server_Header, 22
SERVER_HOST, 99
Server_Host, 22
Server_Key, 19
SERVER_NAME, 99
Server_Name, 22
SERVER_PORT, 99
Server_Port, 22
Server_Priority (*natural*), 22
SERVER_SOCK, 99
Service_Priority (*natural*), 22
SESSION, 99
Session, 11, 22, 24
Session_Cleaner_Priority (*natural*), 23
SESSION_CLEANUP_INTERVAL, 99
Session_Cleanup_Interval (*duration*), 23
Session_Id_Length (*positive*), 22
SESSION_LIFETIME, 99
Session_Lifetime (*duration*), 22
SESSION_NAME, 99
Session_Name, 22
SESSIONS_TERMINATE_V, 99
SESSIONS_TS_V, 99
SESSIONS_V, 99
Simple Mail Transfer Protocol, 87
Simple Object Access Protocol, 65
Simple Page server, 43
Simple server, 13
SLOT_ACTIVITY_COUNTER_V, 99
SMTP, 87, 101
SMTP Authentication, 102
SOAP, 43, 65
SOAP (*API*), 362
SOAP 1.1, 103, 104
SOAP Client, 65
SOAP Dispatcher, 68
SOAP Server, 66
SOAP.Client, 363
SOAP.Dispatchers, 365
SOAP.Dispatchers.Callback, 68, 367
SOAP.Message, 369
SOAP.Message.XML, 371
SOAP.Parameters, 373
SOAP.Types, 377
SOAPAction, 66
SOCK_V, 99
Socket log, 38
split, 44
split pages, 44
SSL, 37, 103
START_TIME, 100
starting server, 10
Static Page server, 43
Status, 97
STATUS_PAGE, 100
Status_Page, 23
Stream resources, 95

T

TCP_No_Delay, 23
Template files, 82
timer, 42
TLS, 37
TLS_Ticket_Support, 23
transient, 43
transient pages, 42, 43
TRANSIENT_CLEANUP_INTERVAL, 100
Transient_Cleanup_Interval, 23
TRANSIENT_LIFETIME, 100
Transient_Lifetime, 23
Trusted_CA, 23

U

Unexpected exceptions, 166
Up_Image, 23
upload, 27, 39
UPLOAD_DIRECTORY, 100
Upload_Directory, 23

UPLOAD_SIZE_LIMIT, 100
URI, 42
User_Agent, 23
Using resources, 95
Utils.SOAP_Wrapper, 67

V

VALUES_M, 100
Verify callback, 33
VERSION, 100
virtual host, 42

W

we_icons, 46
we_js, 46
Web Blocks, 53
web cross-references, 60
Web Elements, 46
Web Service Definition Language, 65, 69
web sockets, 60
Web_Server, 11
WebSocket_Message_Queue_Size, 23
WebSocket_Origin, 23
WebSocket_Priority (*natural*), 23
WebSocket_Timeout, 23
websockets, 60
webxref, 60
Working with Server sockets, 30
WSDL, 65, 69, 78, 79
wsdl2aws, 69, 80
wsdl2aws limitations, 84
WWW_ROOT, 100
WWW_Root, 24