Copyright & trademark notices

This edition applies to version 6.1 of the licensed program AXIGEN and to all subsequent releases and modifications until otherwise indicated in new editions.

Notices

References in this publication to GECAD TECHNOLOGIES S.A. products, programs, or services do not imply that GECAD TECHNOLOGIES S.A. intends to make these available in all countries in which GECAD TECHNOLOGIES S.A. operates. Evaluation and verification of operation in conjunction with other products, except those expressly designated by GECAD TECHNOLOGIES S.A., are the user's responsibility. GECAD TECHNOLOGIES S.A. may have patents or pending patent applications covering subject matter in this document. Supplying this document does not give you any license to these patents. You can send license inquiries, in writing, to the GECAD TECHNOLOGIES S.A. sales department, sales@AXIGEN.com.

Copyright Acknowledgement

(c) GECAD TECHNOLOGIES S.A. 2008. All rights reserved.

All rights reserved. This document is copyrighted and all rights are reserved by GECAD TECHNOLOGIES S.A. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage and retrieval system without the permission in writing from GECAD TECHNOLOGIES S.A.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. GECAD TECHNOLOGIES S.A. will not be responsible for any loss, costs or damages incurred due to the use of this documentation.

AXIGEN™ Mail Server is a SOFTWARE PRODUCT of GECAD TECHNOLOGIES S.A. GECAD TECHNOLOGIES and AXIGEN™ are trademarks of GECAD TECHNOLOGIES S.A. Other company, product or service names may be trademarks or service marks of others.

GECAD TECHNOLOGIES S.A.
10A Dimitrie Pompeiu Blvd., Connect Business Center, 2nd fl., Bucharest 2, ROMANIA; phone: +40-21-303-2080; fax: +40-21-303-2081; e-mail:

Sales: sales@axigen.com
Technical support: support@axigen.com
Website: http://www.axigen.com

(c) Copyright GECAD TECHNOLOGIES S.A. 2008. All rights reserved.
# Table of Contents

## INTRODUCTION
- Welcome 4
- Purpose of this document 4
- Audience of this document 4
- Related documentation 4
- About AXIGEN® Mail Server 5

## 1. HSP Overview 6

## 2. Language structure 7
- 2.1 Variables 7
  - 2.1.1 Declarations 7
  - 2.1.2 Types of Variable Values 7
- 2.2 HSP Instructions 9
- 2.3 Operators 11
- 2.4 Call Methods 11
  - 2.4.1 Special requests 12
  - 2.4.2 Errors 12
  - 2.4.3 General HSP Language Methods 13
  - 2.4.4 WebMail Specific Methods 21
  - 2.4.5 WebAdmin Specific Methods 58
  - 2.4.5 WebmailProxy Specific Methods 95

## 2.5 LANGUAGE CONSTRAINTS 95

## 2.6 HSP COMMENTS 96

## 2.7 OTHER SPECIFICATIONS 96

## 2.8 HSP LIMITATIONS 96
Introduction

Welcome

Thank you for choosing AXIGEN Mail Server as your messaging solution. This document applies to the current version of AXIGEN and subsequent releases until otherwise noticed. Our product has reached version 6.1 and currently runs on Linux, BSD and Solaris platforms. Our development roadmap includes versions for Windows and Mac OS, to be released soon.

Purpose of this document

The purpose of this document is to provide a complete description of AXIGEN proprietary server side scripting language and instructions on how to use it, in order to benefit from AXIGEN Mail Server's extension capabilities.

Audience of this document

This document is written for network/system administrators who would like to change the look and feel, or even the functionalities or the AXIGEN web-based modules - WebMail and WebAdmin.

In order to understand the information in this manual, the audience should have:

• A detailed knowledge of general mail server abilities and functions.
• Good knowledge of the platform (Operating System) on which AXIGEN Mail Server is installed or evaluated.
• Good comprehension of the RFC requirements.
• Good knowledge of HTTP and server side scripting.

Related documentation

Additional information regarding AXIGEN can be found in the following documents:

• AXIGEN Online Documentation:
  http://www.axigen.com/docs/en/
• AXIGEN System Administrator's Manual, available at the following address:
About AXIGEN® Mail Server

AXIGEN Mail Server is an Internet-based mail server that provides messaging services over the Internet via connections using a Transmission Control Protocol/Internet Protocol (TCP/IP) network.

AXIGEN Mail Server sends mail messages using the Simple Mail Transfer Protocol (SMTP). Messages can be retrieved using the Post Office Protocol version 3 (POP3), the Internet Message Access Protocol (IMAP) and the HTTP protocol (HTTP). For remote users AXIGEN offers as a WebMail module.

The administration is easily done through a web interface (WebAdmin), Command Line Interface (CLI) or configuration file.

AXIGEN MAIL SERVER is specifically designed for use in small businesses, corporate environments or at ISPs.

Modular architecture

AXIGEN® Mail Server works on a modular architecture including current advanced Internet messaging protocols/services. The figure below explains all modules of AXIGEN mail server:

Figure 1: AXI GEN modular structure

The above figure presents the mail messages flow inside the AXIGEN mail server. The message goes through various modules in the server and gets delivered securely to its destination.
1. HSP Overview

HSP is the AXIGEN proprietary server-side scripting language, used to generate HTML code for WebMail and WebAdmin, the two web-based components of AXIGEN. HSP code is combined with HTML, to integrate dynamic elements in static HTML pages.

AXIGEN Mail Server uses its own HTTP server and the proprietary server-side scripting HSP language, so that the content, functionality and look of web-based modules can be easily controlled, managed and expanded. WebMail and WebAdmin are expansion-ready modules - this means that you can rebuild them, or merely add functionalities or graphic elements that you may need, by using HSP.

WebMail and WebAdmin content is stored in .hsp type files on the machine where AXIGEN Mail Server was installed. The files for each module are stored at the location indicated by the corresponding ‘path’ parameter, found in the WebMail and WebAdmin contexts from the AXIGEN configuration file (axigen.cfg). This parameter can also be edited using other AXIGEN Administration tools (CLI and WebAdmin).

This diagram illustrates a typical Request-Response process involving the AXIGEN HTTP Server interacting with HSP.

The HSP delimiters are "<%" and "%>", and you can use HSP syntax only between these separators.

Example:

```html
<html>
<% FOR i = 1 TO 10 %>
  <b> <%i%> </b>
  <br>
<% ENDFOR %>
</html>
```
2. Language structure

2.1 Variables

A variable can have one of the following types:
- NUMBER
- STRING
- MAP (associative array)
- ARRAY

Each of these variable types is explained below in section 2.1.2.

2.1.1 Declarations

Variables of different types are declared differently in HSP. NUMBER and STRING variables are declared automatically, when they are used for the first time.

Example:
```hsp
<% a = "some string" %>
<% n = 10 %> 
```

Only MAP and ARRAY variables must be declared explicitly.

Example:
```hsp
<%MAP person%>
<%ARRAY obj%>
```

A variable name MUST start with a letter or underscore character, can contain letters, numbers or the underscore character and can have a maximum length of 255 characters.

VARIABLE = `[a-zA-Z_]([a-zA-Z_0-9]{0,255})`

2.1.2 Types of Variable Values

Type NUMBER

Values of the variables that have this type must follow the regular expression:

NUMBER = (-)?[1-9][0-9]{0,9}|[0]

Type STRING

Strings are defined as follows:

STRING = QUOTE (\^["\n\"]|\"\"|\"\")\* QUOTE
QUOTE = '\"'

Type ARRAY

Elements of an array can have any of the HSP types (NUMBER, STRING, MAP or ARRAY). Each array element is uniquely identified by an index. These are the actions you can perform on an element of an array:

- access
HSP – AXIGEN Server Side Scripting Language

- modify
- push (add an element at the end of an array)
- pop (fetch the element from the end of an array – the array becomes one element shorter)

Example:

```hsp
<%! array declaration %>
<%ARRAY obj%>

<%! push elements of any type %>
<%a = 20%>
<%PUSH(obj, a)%>

<%ARRAY b%>
<%PUSH(obj, b)%>

<%PUSH(obj, 10)%>
<%PUSH(obj, "string")%>

<%! pop an element from array and store it into c %>
<%c = POP(obj)%>

<%! modify EXISTING element %>
<%obj[0] = 2%>

<%! get element value %>
<%a = obj[2]%>
```

The access to an element of an ARRAY must use the following the expression:

```
ARRAY_ACCESS = VARIABLE "[" (VARIABLE | NUMBER) "]"
```

**Type MAP**

Elements of a map can have any of the HSP types. You can access, modify, or insert an element of a map.

Example:

```
<%! map declaration %>
<%MAP person%>

<%! insert new elements of any type %>
<%person.name = s%>
```
HSP – AXIGEN Server Side Scripting Language

```hsp
<person.age = 24>
</* modify existing element */>
person.age = 23
</* get element value */>
a = person.age
```

The access to an element of a MAP must use the expression:

```
MAP_ACCESS = VARIABLE "." VARIABLE
```

## 2.2 HSP Instructions

Five different types of instructions are available in HSP:

### IF - ELSE Statement

The IF-ELSE statement in HSP has the following syntax:

```
IF_EXPR = "<" ("IF" | "IFNOT") METHOD "+" >
ELIF_EXPR = "<" ("ELIF" | "ELIFNOT") METHOD "+" >
ELSE_EXPR = "<" "ELSE" "+" >
ENDIF_EXPR = "<" "ENDIF" "+" >
```

Example:

```
<IFNOT EXISTS(person) %>
  <i> <"variable person doesn't exist"> </i>
</IFNOT %>
<ELIF EXISTS(person.name) %>
  <"The name is "+person.name>
<ELSE %>
  <i> <"The person doesn't have a name"> </i>
</ENDIF %>
```

### FOR Statement

The FOR loop increments a counter from a start index to a stop index. It is used to execute a block of instructions a number of times. It can also be used to iterate through a range of elements of an ARRAY. It has the following syntax:

```
FOR_EXPR = "<" "FOR" VARIABLE "+" PARAM "+" TO" PARAM "+" >
ENDFOR_EXPR = "<" "ENDFOR" "+" >
```

Example:

```
<FOR i = 1 TO 10 %>
  <i> <"%i"> </i>
</FOR %>
```
FOREACH Statement

The FOREACH statement is mainly used to iterate through the elements of an ARRAY or through the properties of a MAP. It has the following syntax:

FOREACH_EXPR = "<% "FOREACH" VARIABLE "IN" (VARIABLE|MAP_ACCESS|ARRAY_ACCESS)
                 [ "INDEXED" "BY" VARIABLE ] ">%>
ENDFOR_EXPR = "<% "ENDFOR" ">%>

Example 1:
<%ARRAY a%>
<%PUSH(a,10)%>
<%PUSH(a,20)%>
<%PUSH(a,30)%>
<%FOREACH el IN a%>
    <el><br>
<%ENDFOR%>

Example 2:
<%MAP person%>
<%person.name = "John"%>
<%person.age = 24%>
<%FOREACH p IN person INDEXED BY property%>
    Person property: <%property%>, value: <%p%>
    <br>
<%ENDFOR%>

BREAK Statement

The BREAK statement is used for breaking loops (FOR, FOREACH). It has the following syntax:

BREAK_EXPR = "<% "BREAK" ">%>

Example:
<%FOR i = 1 TO 10%>
    <i>
    <br>
    <%IF EQ(i,5)%>
        <%BREAK%>
    <%ENDIF%>
<%ENDFOR%>

INCLUDE Statement

The INCLUDE statement is used to insert another HSP source in a HSP file. It has the following syntax:

INCLUDE_EXPR = "<% "INCLUDE" PARAM ">%>"
HSP – AXIGEN Server Side Scripting Language

Here, PARAM represents the path of the HSP file that will be inserted in the current HSP. This path is relative to the value of the "path" parameter configured for WebMail (in the axigen.cfg configuration file).

HSP files from another folder can be inserted only in the private zone.

Also, keep in mind the following limitations/rules:
1. The HTTP server rejects paths like "*/../*".
2. The inclusion depth for paths is limited to 16.
3. Any HSP file MUST have .hsp extension.

Example:
```html
<%INCLUDE "hello.hsp"%>
```

2.3 Operators
The only operator used in HSP is "+". This operator is used to concatenate values.

Example:
```html
<%value = "I have " + 23 + " years and my name is" + person.name + "." %>
```

2.4 Call Methods
The HSP module has 4 sets of methods.
1. A set of methods shared by the WebMail, WebmailProxy and WebAdmin modules which contains general language methods.
2. A set of WebMail specific methods.
3. A set of WebAdmin related methods.
4. A set of WebmailProxy related methods

The general syntax for all HSP methods is as follows:
```
METHOD = VARIABLE "(" ARG_LIST ")"
ARG_LIST = PARAM | (PARAM "," PARAM)
PARAM = (NUMBER | STRING | VARIABLE | ARRAY_ACCESS | MAP_ACCESS | CONCAT)
CONCAT = (PARAM "+" CONCAT) | (PARAM "+" PARAM)
VAR = VARIABLE | ARRAY_ACCESS | MAP_ACCESS
```

Definitions:
In this syntax, MAP is a VAR that has a map value; and ARRAY is a VAR that has an array value;
For the purpose of simplification, from this point on in the current document, we’ll refer to STRING NUMBER, MAP and ARRAY as the variables that have these value types. Also:
VALUE can be STRING or NUMBER.
OBJECT can be STRING, NUMBER, MAP or ARRAY.

Notes:
- The syntax `<%METHOD(el, ARG_LIST)%>` is equivalent to: `<%el = METHOD(ARG_LIST)%>`.
  In this syntax, `<el>` MUST be VAR.
- Method names are case insensitive.

The following sections list methods from all three HSP method categories, and their function.
2.4.1 Special requests

If the WebMail or WebAdmin Server receives a URI (by GET or POST) that contains an "action" query, it will treat this query as a special one, triggering the following actions:

− login: account login, that requests the following queries:
  ➢ username
  ➢ password
  ➢ disableCookie (if enabled, the cookie authentication will be disabled)
− logout: if exists, the current session will be discarded

HSP files are organized in two main zones: public zone and private zone. Access to the private zone is only granted to successfully authenticated clients.

If authentication is not successful, the client has access only to the public zone.

After a successful login, the client receives a session key (and a session cookie key, if the disableCookie option doesn't exist). This session key MUST be used as a value for the "_h" variable in all HTTP queries, for session authentication.

Example: login.hsp

```html
<form action="/?action=login" method = "POST">
    Username: <input type="text" name="username"><br>
    Password: <input type="password" name="password"><br>
    <input type = "checkbox" name = "disableCookie">Disable Cookie<br>
    <input type="submit" name = "login" value = "Login"><br>
</form>
```

Example of an authenticated request: request.hsp

```html
<%h = getSessionId()%>

<form action="test.hsp?_h=<%h%>">
    <input type=submit name="test"><br>
</form>
```

2.4.2 Errors

To find out if a HSP error has occurred after a method call, use the getLastError method.

There are several possible method errors:

− "METHOD_INVALID_NAME"
− "METHOD_PRIVATE"
− "ARG_TOO_FEW"
− "ARG_TOO_MANY"
− "ARG_INVALID_TYPE"
− "ARG_INVALID_VALUE"
− "ARG_NULL_VALUE"

An error can also occur at protocol level. To find out if a protocol error has occurred, first call the getLastError method in a HSP page. Possible protocol errors:

− "HTTP_HEADERS_TOO_LARGE"
HSP – AXIGEN Server Side Scripting Language

- "HTTP_HEADER_TOO_LONG"
- "HTTP_BODY_TOO_LARGE"
- "RESOURCE_TOO_LARGE"

In HSP, any syntax, language or I/O error is reported. In case of such an error, the HSP file is replaced with a hard-coded error page.

Syntax errors
Misspelling method names is not reported as an error. If any of the specifications of this document is violated, then a syntax error is generated.

Language errors
A language error is generated when an instruction is not properly implemented. Such a case is illustrated below:

Example:

```html
<html>
  <%FOR i = 0 TO 10%>
  <%i%>
  <%ENDIF%><%/* this is a language error */%>
</html>
```

If any of the specifications of this document is violated, then a syntax error is generated.

2.4.3 General HSP Language Methods
These general methods are used for WebMail, WebMailProxy and WebAdmin.

1.

<table>
<thead>
<tr>
<th>Name</th>
<th>inc (Increment)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>inc(NUMBER contor [, NUMBER step])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Increments &lt;contor&gt; by &lt;step&gt;. If &lt;step&gt; is missing, &lt;contor&gt; will be incremented by 1.</td>
</tr>
</tbody>
</table>

Example:

```html
<%a = 4%>
<%inc(a)%>          /* 4 + 1 = 5 */
<%inc(a, 2)%>       /* 5 + 2 = 7 */
<%inc(a, "test")%>  /* 7 + 0 = 7; "test" casted to NUMBER is 0 */
<%inc(a, test)%>    /* 7 + 0 = 7; test variable doesn't exist */
<%MAP m%>
<%inc(a, m)%>       /* 7 + 0 = 7; m variable doesn't have a value */
```

2.

<table>
<thead>
<tr>
<th>Name</th>
<th>dec (Decrement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>dec(NUMBER contor [, NUMBER step])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Decrements &lt;contor&gt; by &lt;step&gt;. If &lt;step&gt; is missing, &lt;contor&gt; will be decremented by 1.</td>
</tr>
</tbody>
</table>
### 3. mul

<table>
<thead>
<tr>
<th>Name</th>
<th>mul</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>NUMBER mul(NUMBER val1 , NUMBER val2)</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>Returns (&lt;val1&gt; \times &lt;val2&gt;).</td>
</tr>
</tbody>
</table>

### 4. mod

<table>
<thead>
<tr>
<th>Name</th>
<th>mod</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>NUMBER mod(NUMBER val1 , NUMBER val2)</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>Returns (&lt;val1&gt; \mod &lt;val2&gt;).</td>
</tr>
</tbody>
</table>

### 5. div

<table>
<thead>
<tr>
<th>Name</th>
<th>div</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>NUMBER div(NUMBER val1 , NUMBER val2)</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>Returns (&lt;val1&gt; / &lt;val2&gt;).</td>
</tr>
</tbody>
</table>

### 6. eq

<table>
<thead>
<tr>
<th>Name</th>
<th>eq</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>eq(VALUE arg1, VALUE arg2[, ... VALUE arg6])</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>Returns true if (&lt;arg1&gt;) is equal to one of the other arguments (case sensitive), returns false otherwise. A maximum of 6 arguments is supported.</td>
</tr>
</tbody>
</table>

Example:

```html
<%a = 1%>
<%b = 2%>
<%IF eq(a, b)%>
    equal
<%ELSE%>
    not equal
<%ENDIF%>
```

### 7. eqcase

<table>
<thead>
<tr>
<th>Name</th>
<th>eqcase</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>eqcase(STRING arg1, STRING arg2[, ... VALUE arg6])</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>Returns true if (&lt;arg1&gt;) is equal to one of the other arguments (case INSENSITIVE), returns false otherwise. A maximum of 6 arguments is supported.</td>
</tr>
</tbody>
</table>

### 8. eqmatch

<table>
<thead>
<tr>
<th>Name</th>
<th>eqmatch</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>eqmatch(VALUE arg1, VALUE arg2)</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>Returns true if (&lt;arg1&gt;) is matched by the regular expression arg2, returns false otherwise.</td>
</tr>
</tbody>
</table>

Example:

```html
<%a = 1%>
<%b = "[0-9]+" %>
```
<%IF eq(a, b)%>
    equal
<%ELSE%>
    not equal
<%ENDIF%>

9. Name | eqmatchcase
Syntax | eqmatchcase(STRING arg1, STRING arg2)
Explanation | Returns true if <arg1> is matched by the regular expression arg2 (case sensitive). Returns false otherwise.

10. Name | gt (greater)
Syntax | gt(NUMBER arg1, NUMBER arg2)
Explanation | Returns true if <arg1> is greater than <arg2>.

11. Name | gte (greater than or equal to)
Syntax | gte(NUMBER arg1, NUMBER arg2)
Explanation | Returns true if <arg1> is greater than or equal to <arg2>.

12. Name | lt (less than)
Syntax | lt(NUMBER arg1, NUMBER arg2)
Explanation | Returns true if <arg1> is less than <arg2>.

13. Name | lte (less than or equal to)
Syntax | lte(NUMBER arg1, NUMBER arg2)
Explanation | Returns true if <arg1> is less than or equal to <arg2>.

14. Name | eval (evaluate)
Syntax | eval(STRING eval1, STRING eval2):
Explanation | Assigns the value of the variable that has the identifier <eval2> to the variable that has the identifier <eval1>

Example:
<%i2 = 4%>
<%i10 = 8%>
<%i2%>  /* 4 */%
<%eval("i"+2, "i"+10)%>
<%i2%>  /* 8 */%
### 15. `exists`

<table>
<thead>
<tr>
<th>Name</th>
<th>exists</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>exists(VAR arg)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns true if <code>&lt;arg&gt;</code> exists</td>
</tr>
</tbody>
</table>

### 16. `isempty`

<table>
<thead>
<tr>
<th>Name</th>
<th>isempty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>isempty(VAR arg)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns true if value of <code>&lt;arg&gt;</code> is empty; if <code>&lt;arg&gt;</code> is an ARRAY or a MAP returns true if it has no element.</td>
</tr>
</tbody>
</table>

### 17. `pop`

<table>
<thead>
<tr>
<th>Name</th>
<th>pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>OBJECT pop(ARRAY A)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Pops an element of array A and returns it</td>
</tr>
</tbody>
</table>

### 18. `push`

<table>
<thead>
<tr>
<th>Name</th>
<th>push</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>push(ARRAY A, OBJECT el)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Pushes element <code>&lt;el&gt;</code> into ARRAY <code>&lt;A&gt;</code></td>
</tr>
</tbody>
</table>

Example:
```hsp
<%ARRAY a%>
<%push(a, 1)%>
<%push(a, "two")%>
<%three = 3%>
<%push(a, three)%>
<%i = pop(a)%>
<%i%> /* 3 */
```

### 19. `map_getValue`

<table>
<thead>
<tr>
<th>Name</th>
<th>map_getValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>ANY map_getValue(MAP map, STRING prop)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>It returns the value of <code>&lt;prop&gt;</code> property, contained in <code>&lt;map&gt;</code>.</td>
</tr>
</tbody>
</table>

### 20. `map_setValue`

<table>
<thead>
<tr>
<th>Name</th>
<th>map_setValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>map_setValue(MAP map, STRING prop, OBJECT data)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Sets <code>&lt;data&gt;</code> to the property <code>&lt;prop&gt;</code> of <code>&lt;map&gt;</code>.</td>
</tr>
</tbody>
</table>

### 21. `type`

<table>
<thead>
<tr>
<th>Name</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>STRING type(VAR data)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>It returns the type of <code>&lt;data&gt;</code>. The type can be &quot;value&quot;, &quot;map&quot;, or &quot;vector&quot;.</td>
</tr>
</tbody>
</table>
22. Name getSize
Syntax NUMBER getSize(VAR arg)
Explanation Returns size of <arg> (<arg> can be an ARRAY or a MAP)

23. Name str_escape
Syntax STRING str_escape(STRING arg)
Explanation Escapes string <arg> and returns this value.

Example:
```%
<a = "test\"test"%> <%/* test"test */%>
<str%> <%/* test\"test */%>
```

24. Name str_str
Syntax NUMBER str_str(STRING haystack, STRING needle)
Explanation It returns the first occurrence of the substring <needle> in the string <haystack>. -1 is returned if there is no occurrence.

Example:
```%
<idx = str_str("aabbaa", "bb")%>
<idx%> <%/* 2 */%>
```

25. Name str_replace
Syntax STRING str_replace(STRING haystack, STRING needle, STRING newsub)
Explanation It replaces all <needle> occurrences, from <haystack>, with <newsub> and stores the new value into <newstr>.

Example:
```%
<str = "abcabc"%>
<newstr = str_replace(str, "ab", "a")%>
<newstr%> <%/* acac */%>
```

26. Name str_substring
Syntax STRING str_substring(STRING str, NUMBER begin, NUMBER end)
Explanation Returns the substring of <str>, from <begin> to <end>-1.

27. Name str_len
Syntax NUMBER str_len(STRING str)
Explanation Returns the length of <str>. 

Copyright © 2008 Gecad Technologies S.A.
http://www.axigen.com
28. 
<table>
<thead>
<tr>
<th>Name</th>
<th>str_cmp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>NUMBER str_cmp(STRING str1, STRING str2)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns strcmp(str1, str2).</td>
</tr>
</tbody>
</table>

29. 
<table>
<thead>
<tr>
<th>Name</th>
<th>str_caseCmp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>NUMBER str_caseCmp(STRING str1, STRING str2)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns:strcasecmp(str1, str2).</td>
</tr>
</tbody>
</table>

30. 
<table>
<thead>
<tr>
<th>Name</th>
<th>str_split</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY str_split(STRING str, STRING delim)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns an array of tokens(strings) from &lt;str&gt;, delimited by &lt;delim&gt;.</td>
</tr>
</tbody>
</table>

31. 
<table>
<thead>
<tr>
<th>Name</th>
<th>str_join</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING str_join(ARRAY strArray, STRING delim)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns a string which contains all tokens from &lt;strArray&gt; delimited by &lt;delim&gt;.</td>
</tr>
</tbody>
</table>

32. 
<table>
<thead>
<tr>
<th>Name</th>
<th>str_trim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING str_trim(STRING str[, STRING type])</td>
</tr>
</tbody>
</table>
| Explanation | Trims white spaces from the beginning and/or the end of <str>. <type> can be:  
- "left": white spaces from the beginning of string are trimmed  
- "right": white spaces from the end of string are trimmed  
- parameter can miss: white spaces from the beginning AND the end of string are trimmed |

33. 
<table>
<thead>
<tr>
<th>Name</th>
<th>encode_url</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING encode_url(STRING arg):</td>
</tr>
<tr>
<td>Explanation</td>
<td>Encodes &lt;arg&gt; in URL format and returns this value; transforms non-alphanumeric characters in %{hex}{hex} and spaces in &quot;+&quot;. (For example, &quot;.&quot; becomes &quot;%2E&quot;).</td>
</tr>
</tbody>
</table>

Example: 
<%str = "a.b c"%> 
<% newstr = encode_url(str)%> 
<%newstr%> <%/* a%2E+c */%>
### 34. `decode_url`

**Syntax**
```plaintext
STRING decode_url(STRING arg)
```

**Explanation**
Decodes `<arg>` from url format and returns this value.

### 35. `encode_html`

**Syntax**
```plaintext
STRING encode_html(STRING arg):
```

**Explanation**
Encodes `<arg>` in html format and returns this value; '<' becomes "<", '>' becomes ">" and "&" becomes "&".

#### Example:
```
<%str = "<html>"%>
<%newstr = encode_html(str)%>
<%newstr%> <%/* <html> */%>
```

### 36. `url_ok`

**Syntax**
```plaintext
url_ok(STRING url)
```

**Explanation**
Returns true if the servers considers `<url>` as a valid external url.

### 37. `getLastError`

**Syntax**
```plaintext
STRING getLastError()
```

**Explanation**
Returns the last error, after a method call

#### Example:
```
<%str = "aaa"%>
<%newstr = str_replace(str)%>
<%error = getLastError()%>
<%error%> <% /* ARG_TOO_FEW */%>
```

### 38. `random`

**Syntax**
```plaintext
NUMBER random(NUMBER n, NUMBER m)
```

**Explanation**
Returns a random value, between `<n>` and `<m>-1`

### 39. `getSessionId`

**Syntax**
```plaintext
STRING getSessionId()
```

**Explanation**
Returns the current session key.

### 40. `setHeaders`

**Syntax**
```plaintext
setHeaders(NUMBER httpCode, [STRING h1 [, STRING h2 [...:]]])
```
**Explanation**: It resets HTTP response headers. It is used for a personalized response to the client.

**WARNING**: Don't write any other HTML or HSP code before this method!

41.

<table>
<thead>
<tr>
<th>Name</th>
<th>headers_getUserAgent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>STRING headers_getUserAgent()</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>Returns User-Agent HTTP header value.</td>
</tr>
</tbody>
</table>

42.

<table>
<thead>
<tr>
<th>Name</th>
<th>json2hsp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax</strong></td>
<td>ANY json2hsp(STRING json)</td>
</tr>
<tr>
<td><strong>Explanation</strong></td>
<td>It returns the hsp entity with the associated json representation. If the json string is illegal the following errors can be returned:</td>
</tr>
<tr>
<td></td>
<td>- JSON_FLEX_ERROR - usually when an illegal character was encountered in the json string (e.g. unescaped control character U+0000:U+001F);</td>
</tr>
<tr>
<td></td>
<td>- JSON_SYNTAX_ERROR - when a token is encountered in an illegal context (e.g. {&quot;abc&quot;  &quot;xyz&quot;} -&gt; missing ':' between key &quot;abc&quot; and value &quot;xyz&quot;);</td>
</tr>
<tr>
<td></td>
<td>- JSON_DATA_STACK_FULL or JSON_STATE_STACK_FULL - if the resulting object depth exceeded 128 levels (e.g. &quot;[[]...]]&quot; - with 129 left/right brackets);</td>
</tr>
</tbody>
</table>

**Note:**
- the full json syntax is supported by method;
- in addition(to strings) the keys in maps can be HSP identifiers, so the json '{abc: 1}' is successfully parsed like '{"abc": 1}';
- the U+0000 unicode is not encoded in string values (the json string "abc\u0000xyz" will result in the "abcxyz" hsp string);
- strings are limited to 4096 memory bytes; all json atomic values(strings, numbers, true, false, null) will be stringified by this method (eg. the hsp json strings "{abc: 123, xyz:[true, false, null]}" and "{\"abc\" : \"123\", \"xyz\": [\"true\", \"false\", \"null\"]}" will produce the same hsp entity);

Example:

```hsp
<% string = "\"numar\": 1234.5678e+90, string:\"abc\u1234xyz\u5678uvt\", array:[true, false, null]]" %>
<% IFNOT json2hsp(obj, string) %>
  <% getLastError (error) %>
  <% /* check error: */ %>
  Method json2hsp failed with error code: |<% error %>|.
<% ELSE %>
  <% /* use obj: lets dump it */ %>
  <% IFNOT hsp2json (json, obj) %>
    <% getLastError(error) %> 
```
**Method hsp2json failed with error code: |<% error %>|.**

**Test PASS: the json of object is: |<% json %>|.**

<table>
<thead>
<tr>
<th>Name</th>
<th>hsp2json</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING hsp2json(ANY)</td>
</tr>
<tr>
<td>Explanation</td>
<td>It returns json string representaion of the hsp entity (string, array, map). If the object is too big then the method fails with error code JSON_OBJECT_TOOBIG; an object is too big if its json string is greater than one mega (1024K) or its depth greater than 128.</td>
</tr>
</tbody>
</table>

### 2.4.4 WebMail Specific Methods

The following methods are only used for WebMail related operations:

<table>
<thead>
<tr>
<th>Name</th>
<th>isMobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>isMobile()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns true if the client was detected as a mobile browser.</td>
</tr>
</tbody>
</table>

**Folder specific methods**

1. **folder_create**

<table>
<thead>
<tr>
<th>Name</th>
<th>folder_create</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>folder_create(STRING folderName, STRING parentPath[, STRING folderType])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Creates a folder with name &lt;folderName&gt;. The parent's folder will be &lt;parentPath&gt;. The root parent is &quot;/&quot; or &quot;.&quot;. If &lt;folderType&gt; is present the folder will have that type: &quot;calendar&quot;, &quot;tasks&quot;, &quot;journal&quot;, &quot;notes&quot;, &quot;contacts&quot;.</td>
</tr>
</tbody>
</table>

Example:

```hsp
<%folder_create("test")%>
```

2. **folder_delete**

<table>
<thead>
<tr>
<th>Name</th>
<th>folder_delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>folder_delete(STRING folderPath)</td>
</tr>
</tbody>
</table>
| Explanation | Deletes the folder with absolute path <folderPath>. If the folder is a special folder (like INBOX), <folder_empty> method will be called.

- Special folders for an account are: "INBOX", "Sent", "Drafts" and "Trash".
- Special folders for a mail list are the following: "INBOX", "Sent", "Drafts", "Trash", "Requests", "PendingRequests". |
### HSP – AXIGEN Server Side Scripting Language

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ConfirmedRequests&quot;, &quot;Pending&quot;, &quot;Reject&quot;, &quot;ToBeRejected&quot; and &quot;Deleted&quot;.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

```html
<%folder_delete("test")%>
```

3. **folder_empty**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder_empty</td>
<td>folder_empty(STRING folderPath)</td>
<td>Empties the folder with absolute path &lt;folderPath&gt;.</td>
</tr>
</tbody>
</table>

**Example:**

```html
<%folder_empty("test")%>
```

4. **folder_move**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder_move</td>
<td>folder_move(STRING oldFolderPath, STRING newFolderParent)</td>
<td>Moves a folder with absolute path &lt;oldFolderPath&gt; into the new location &lt;newFolderParent&gt;.</td>
</tr>
</tbody>
</table>

5. **folder_rename**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder_rename</td>
<td>folder_rename(STRING oldFolderPath, STRING newFolderName)</td>
<td>Renames a folder with absolute path &lt;oldFolderPath&gt;. The new name will be &lt;newFolderName&gt;.</td>
</tr>
</tbody>
</table>

**Example:**

```html
<%folder_rename("test/old", "new")%>
```

6. **folder_setSortType**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder_setSortType</td>
<td>folder_setSortType(NUMBER fid, STRING type, STRING order)</td>
<td>Sets the sort &lt;type&gt; and &lt;order&gt; for &lt;fid&gt; folder. After this method will be called, mails from &lt;fid&gt; folder will be sorted in &lt;order&gt;(&quot;ascending&quot; or &quot;descending&quot;), by &lt;type&gt; (&quot;subject&quot;, &quot;to&quot;, &quot;date&quot;, &quot;from&quot; or &quot;size&quot;).</td>
</tr>
</tbody>
</table>

7. **folder_setGroupType**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder_setGroupType</td>
<td>folder_setGroupType(NUMBER fid, STRING type, STRING order)</td>
<td>The method sets the group &lt;type&gt; and &lt;order&gt; for &lt;fid&gt; folder. After this method will be called, mails from &lt;fid&gt; folder will be grouped in &lt;order&gt;(&quot;ascending&quot; or &quot;descending&quot;), by &lt;type&gt; (&quot;subject&quot;, &quot;from&quot;, &quot;conversation&quot;, or &quot;none&quot;).</td>
</tr>
</tbody>
</table>

8. **folder_setResolutionView**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>folder_setResolutionView</td>
<td>folder_setResolutionView(NUMBER fid, STRING resolutionView)</td>
<td></td>
</tr>
</tbody>
</table>
The method sets the `<resolutionView>` for `<fid>` folder. Depending on the folder type you can have the following resolution views:
- EVENTS folder: "day", "week", "workWeek", "month", "list"
- TASKS folder: "all", "uncompleted", "completed"
- JOURNAL folder: "day", "month", "list"

<table>
<thead>
<tr>
<th>Name</th>
<th>folder_loadList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY folder_loadList()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns current account's folders. Every element has MAP type and the following properties:</td>
</tr>
<tr>
<td></td>
<td>STRING name: folder's name</td>
</tr>
<tr>
<td></td>
<td>STRING path: absolute path</td>
</tr>
<tr>
<td></td>
<td>NUMBER fid: folder ID</td>
</tr>
<tr>
<td></td>
<td>NUMBER count: number of all mails from folder</td>
</tr>
<tr>
<td></td>
<td>NUMBER news: number of unseen mails from folder</td>
</tr>
<tr>
<td></td>
<td>STRING type: type of the folder - can be: &quot;local&quot;, &quot;events&quot;, &quot;journal&quot;, &quot;tasks&quot;, &quot;notes&quot;, &quot;public&quot; or &quot;shared&quot;.</td>
</tr>
<tr>
<td></td>
<td>VALUE isSelectable: this property exists if the folder is selectable.</td>
</tr>
<tr>
<td></td>
<td>VALUE hasChildren: this property exists if the folder has one or more children.</td>
</tr>
</tbody>
</table>

Example of a tree listing of folders:
```
<%folderList = folder_loadLocalList()%>
<%FOREACH folder IN folderList%>
    <%FOR i = 1 TO folder.level%> <%ENDFOR%>
    <%IF exists(folder.isSelectable)%>
        <%(folder.name%)(<%folder.news%>/<%folder.count%>)
    <%ELSE%>
        <%(folder.name%>
    <%ENDIF%>
    <br>
<%ENDFOR%>
```

<table>
<thead>
<tr>
<th>Name</th>
<th>folder_loadById</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP folder_loadById(NUMBER folderId)</td>
</tr>
</tbody>
</table>
| Explanation  | Returns the following properties for the folder identified by `<folderId>`.
|              | STRING name: folder name |
|              | STRING path: absolute path |
|              | NUMBER count: number of mails from folder |
|              | NUMBER news: number of unseen mails from folder |
## HSP – AXIGEN Server Side Scripting Language

- NUMBER \( \text{fid} \): folder id
- STRING \( \text{sortType} \): "subject", "from", "size", "date" or "incoming"
- STRING \( \text{sortOrder} \): "ascending" or "descending"
- VALUE \( \text{hasChildren} \): this property exists if the folder has one or more children.
- STRING \( \text{resolutionView} \): depending on the folder type you can have the following resolution views:
  - EVENTS folder: "day", "week", "workWeek", "month", "list"
  - TASKS folder: "all", "uncompleted", "completed"
  - JOURNAL folder: "day", "month", "list"

### 11. Name

<table>
<thead>
<tr>
<th>Name</th>
<th>folder_loadByPath</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP folder_loadByPath(STRING folderPath)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns the same properties as the method &lt;folder_loadById&gt; for the folder identified by &lt;folderPath&gt;.</td>
</tr>
</tbody>
</table>

### Shared Folder Providers specific methods

1. Name

<table>
<thead>
<tr>
<th>Name</th>
<th>sharedFolderProvider_loadList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY sharedFolderProvider_loadList()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns the string list of shared folders providers.</td>
</tr>
</tbody>
</table>

2. Name

<table>
<thead>
<tr>
<th>Name</th>
<th>sharedFolderProvider_add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sharedFolderProvider_add(STRING accountName)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Add &lt;accountName&gt; to shared folders providers list.</td>
</tr>
</tbody>
</table>

3. Name

<table>
<thead>
<tr>
<th>Name</th>
<th>sharedFolderProvider_remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sharedFolderProvider_remove(STRING accountName)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Remove &lt;accountName&gt; to shared folders providers list.</td>
</tr>
</tbody>
</table>

### Folder permission specific methods

1. Name

<table>
<thead>
<tr>
<th>Name</th>
<th>folderPermission_loadNames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY folderPermission_loadNames()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns the list of all possible permissions. The list contains MAPs with following properties:</td>
</tr>
</tbody>
</table>
  - STRING name: permission name
  - STRING description: permission description |
2. **Name** | folderPermission_loadList  
**Syntax** | ARRAY folderPermission_loadList(NUMBER resourceId)  
**Explanation** | Returns the list of permissions for `<resourceId>`. The resource can be a folder or the current account. If the resource is a folder then the `<resourceId>` is a folder id. Otherwise, if the resource is the account then `<resourceId>` is the `<folderContainerId>` property obtained with account_loadInfo method. Every permission has the following properties:  
- STRING name: principal's name  
- ARRAY allow: the string list with allow permissions  
- ARRAY deny: the string list with deny permissions

3. **Name** | folderPermission_loadPrincipalRights  
**Syntax** | ARRAY folderPermission_loadPrincipalRights(NUMBER resourceId, STRING principalName)  
**Explanation** | Returns the permissions list available for `<principalName>` on `<resourceId>`

4. **Name** | folderPermission_loadMyRights  
**Syntax** | ARRAY folderPermission_loadMyRights(NUMBER resourceId)  
**Explanation** | Returns the permissions list available for current account on `<resourceId>`

5. **Name** | folderPermission_set  
**Syntax** | folderPermission_set(NUMBER resourceId, STRING principalName, ARRAY permissions)  
**Explanation** | Try to set `<permissions>` for `<principalName>` on `<resourceId>`. Every permission has the properties:  
- STRING name: permission's name  
- STRING type: allow or deny

6. **Name** | folderPermission_remove  
**Syntax** | folderPermission_remove(NUMBER resourceId, STRING principalName)  
**Explanation** | Try to remove all permissions for `<principalName>` on `<resourceId>`.

7. **Name** | folderPermission_getRecursive  
**Syntax** | STRING folderPermission_getRecursive(NUMBER fid)  
**Explanation** | Returns "true" if permissions are applied recursively on folder `<fid>` and "false" otherwise.
8. **folderPermission_setRecursive**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>folderPermission_setRecursive</td>
<td>folderPermission_setRecursive(NUMBER fid, STRING recursiveValue)</td>
</tr>
</tbody>
</table>

**Explanation**
Try to apply recursively permissions on folder `<fid>` if `<recursiveValue>` is “true” or to disable that otherwise.

### Contacts specific methods

1. **contact_loadList**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>contact_loadList</td>
<td>ARRAY contact_loadList(NUMBER fid[, NUMBER startIdx])</td>
</tr>
</tbody>
</table>

**Explanation**
Loads a list of contacts from folder `<fid>`. If `<startIdx>` is present the list will begin from this index and will contain maximum pageSize webmail option. Otherwise, the list will contain all contacts from folder. Every contact has the properties:
- STRING firstName
- STRING lastName
- STRING nickName
- NUMBER id

2. **contact_getListSize**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>contact_getListSize</td>
<td>NUMBER contact_getListSize(NUMBER fid)</td>
</tr>
</tbody>
</table>

**Explanation**
Returns the number of contacts from folder `<fid>`.

3. **contact_load**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>contact_load</td>
<td>MAP contact_load(NUMBER fid, NUMBER id)</td>
</tr>
</tbody>
</table>

**Explanation**
Loads the contact `<id>` from folder `<fid>` and returns:
- NUMBER id
- STRING email
- STRING firstName
- STRING lastName
- STRING nickName
- STRING personalEmail
- STRING businessEmail
- STRING phone
- STRING mobilePhone
- STRING homePhone
- STRING businessPhone
- STRING homeAddress
- STRING businessAddress
- STRING notes
4. **Name** contact_add  
   **Syntax** contact_add(NUMBER fid, MAP contactInfo)  
   **Explanation** Try to add the contact with the properties contained in the MAP element <contactInfo> in folder <fid>.

5. **Name** contact_update  
   **Syntax** contact_update(NUMBER fid, NUMBER id, MAP contactInfo)  
   **Explanation** Try to update the properties contained in <contactInfo> for the contact <id> from folder <fid>.

6. **Name** contact_remove  
   **Syntax** contact_remove(NUMBER fid, NUMBER id)  
   **Explanation** Removes the contact <id> from folder <fid>.

7. **Name** contact_loadImportList  
   **Syntax** ARRAY contact_loadImportList()  
   **Explanation** Returns a list of contacts imported from a CSV file. The file must be submitted before, with action=upload.

8. **Name** addressBook_loadList  
   **Syntax** ARRAY addressBook_loadList([NUMBER startIdx])  
   **Explanation** Loads the ordered list of personal contacts, domain contacts (if useDomainContacts webmail option is enabled) and public contacts (if useRcptContacts webmail option is enabled). If <startIdx> is present the list will begin from this index and will contain maximum pageSize webmail option. Otherwise, the list will contain all contacts. Every contact has the properties:  
   - STRING firstName  
   - STRING lastName  
   - STRING nickname  
   - STRING type: personal, domain or public  
   - NUMBER id

9. **Name** addressBook_getSize  
   **Syntax** NUMBER addressBook_getSize()  
   **Explanation** Returns the number of contacts from address book.
### Alias specific methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias_loadTemporaryList</td>
<td>ARRAY alias_loadTemporaryList()</td>
<td>Returns the list of all temporary aliases. Every alias has the properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- STRING name: alias name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- STRING expire: expiry date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias_createTemporary</td>
<td>MAP alias_createTemporary()</td>
<td>Try to create a temporary alias. In success case, returns:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- STRING name: alias name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- STRING expire: expiry date</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>alias_removeTemporary</td>
<td>alias_removeTemporary(STRING aliasName)</td>
<td>Try to remove &lt;aliasName&gt; temporary alias.</td>
</tr>
</tbody>
</table>

### Filters specific methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sieveFilter_isEmailBlackListed</td>
<td>sieveFilter_isEmailBlackListed(STRING email)</td>
<td>Returns true if the email is blacklisted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sieveFilter_orderMoveUp</td>
<td>sieveFilter_orderMoveUp(INTEGER id)</td>
<td>Moved the filter with the given id up one position.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>sieveFilter_orderMoveDown</td>
<td>sieveFilter_orderMoveDown(INTEGER id)</td>
<td>Moved the filter with the given id down one position.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY sieveFilter_loadBlackList</td>
<td>ARRAY sieveFilter_loadBlackList()</td>
<td>Loads the list of email addresses that have been blacklisted. The ARRAY contains STRINGS.</td>
</tr>
</tbody>
</table>
5. **Name** sieveFilter_addToBlackList  
**Syntax** sieveFilter_addToBlackList(STRING email)  
**Explanation** Adds an email address to the black list.

6. **Name** sieveFilter_removeFromBlackList  
**Syntax** sieveFilter_removeFromBlackList(NUMBER email_index)  
**Explanation** Removes the entry with the email_index position from the black list.

7. **Name** sieveFilter_loadList  
**Syntax** ARRAY sieveFilter_loadList()  
**Explanation** Loads the list of WebMail filters for the current account. Every element has MAP type and contains the following properties:  
- STRING name: the name of the filter  
- NUMBER id: the ID of the filter  
- STRING enabled: can be "yes" or "no".

8. **Name** sieveFilter_add  
**Syntax** sieveFilter_add(MAP filter)  
**Explanation** Adds the filter with the properties specified by <filter>. The MAP can contain the following properties:  
- STRING name: name of the filter  
- STRING match: can be "all" or "any"  
- ARRAY expressions: expressions to be matched; each element has MAP type and can contain the following properties:  
  - STRING header: header to be matched; can be "no header", "subject", "to", "cc", "to or cc", "from", "size", "custom"  
  - STRING customHeader: custom header to be matched; exists only if <header> is "custom"  
  - STRING value: value to be matched on header  
  - STRING condition: can be "is greater than", "is lower than" if <header> is "size" or the <customHeader> is defined and considered to have a numerical value, or it can be "contains", "is", "begins with", "ends with", "does not contain", "is not", "does not begin with", "does not end with" for any other <header> value  
- ARRAY actions: a list of actions to be taken when matching occurs; each element has MAP type and can contain the following properties:  
  - STRING type: can be "no action", "move", "copy", "delete", "forward" or "vacation"
HSP – AXIGEN Server Side Scripting Language

If <type> is "vacation" then the following properties must exist for an action element:
- STRING subject: subject of the message
- STRING text: body of the message
- STRING days: number of days after another message will be sent to the same address

If <type> is not "vacation", then the following property must be defined:
- STRING destination: if <type> is "forward", then <destination> property should contain the e-mail address where the mail will be forwarded; in case <type> is "move" or "copy", then it must contain the name of the folder where the message will be moved or copied.

For more clarity, examples will follow for each type of filter that can be created. The example below adds a simple filter, which forwards any message that contains the string "[Jokes]" to "jokes@localdomain" email address.

```%
MAP filter

filter.name="Jokes"

filter.match="all"

ARRAY expressions

MAP expression

expression.header="subject"

expression.value="[Jokes]"

expression.condition="contains"

push(expressions,expression)

filter.expressions=expressions

ARRAY actions

MAP action

action.type="forward"

action.destination="jokes@localdomain"

push(actions,action)

filter.actions=actions

IF sieveFilter_add(filter)

Filter added

ELSE

Adding filter failed

ENDIF%
```

The following example illustrates filtering on a custom header. It checks the custom header "X-Spam" and if it is "***" then it moves the message into "Spam" folder.

```%
MAP filter

filter.name="Spam"
```
The last example illustrates the creation of a responder (<type> is "vacation").

```axigen
<%MAP filter%>
<%filter.name="Out of Office"%>
<%filter.match="any"%>
<%ARRAY expressions%>
<%filter.expressions=expressions%>
<%ARRAY actions%>
<%MAP action%>
<%action.type="vacation"%>
<%action.subject="Out of office reply"%>
<%action.text="I will be out of office until the 20th October."%>
<%action.days=7%>
<%push(actions,action)%>
<%filter.actions=actions%>
<%IF sieveFilter_add(filter)%>
    Filter added
<%ELSE%>
    Adding filter failed
<%ENDIF%>
```
9. **Name** sieveFilter_load  
**Syntax** MAP sieveFilter_load(NUMBER filterId)  
**Explanation** Loads the filter with the ID `<filterId>`. This is the current active filter. The returned MAP element contains the same properties as for filter_add, with the same meanings, and two more properties:  
- STRING enabled: it can be "true" or "false"  
- NUMBER id: the ID of the filter.

10. **Name** sieveFilter_enable  
**Syntax** sieveFilter_enable(NUMBER filterId)  
**Explanation** Enables the filter with the ID `<filterId>`.

11. **Name** sieveFilter_disable  
**Syntax** sieveFilter_disable(NUMBER filterId)  
**Explanation** Disables the filter with the ID `<filterId>`.

12. **Name** sieveFilter_update  
**Syntax** sieveFilter_update(NUMBER filterId, MAP filter)  
**Explanation** Updates the properties contained in `<filter>` for the filter specified by `<filterId>` (MAP element has the same properties as for `#filter_load` method).

13. **Name** sieveFilter_remove  
**Syntax** sieveFilter_remove(NUMBER filterId)  
**Explanation** Removes the filter identified by `<filterId>` from the list of filters.

14. **Name** sieveFilter_commit  
**Syntax** sieveFilter_commit()  
**Explanation** Commits the changes made on the current active filter.

15. **Name** sieveFilter_reset  
**Syntax** sieveFilter_reset()  
**Explanation** Resets the changes made on the current active filter.

16. **Name** sieveFilter_setOrder  
**Syntax** sieveFilter_setOrder(STRING order)  
**Explanation** Sets the filters order. The `<order>` parameter must contain the id's of the filters in the desired order, comma separated.
Example for 3 filters:

`sieveFilter_setOrder("2, 3, 1")`

### Mail specific methods

1. 

<table>
<thead>
<tr>
<th>Name</th>
<th>mail_compose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP mail_compose(STRING action[, NUMBER fid, NUMBER mid])</td>
</tr>
</tbody>
</table>
| Explanation | Tries to send a composed mail. First it verifies its correctness. If mail_compose returns true the syntax is OK. If the returned MAP is empty, the mail was sent, otherwise you can identify bad fields:
- toErr (exists only if to is wrong)
- ccErr (exists only if cc is wrong)
- bccErr (exists only if bcc is wrong)

<action> can have one of the following values:
- "save": the mail is saved to "Drafts" folder
- "reply": the mail is send as a reply to the mail identified by by <fid> (folder id of the folder that contains the mail that is replied) and <mid> (mail id of the mail that is replied)
- "replyall": the same action as "reply", but the mail will be replied to all RCPT's from the mail identified by <fid> and <mid>.
- "forward": the mail identified by <fid> and <mid> will be forwarded
- "compose": default action, for simple compose

Any other value of <action> is considered "compose".

Possible errors:
- INTERNAL_SERVER_ERROR
- MAIL_COMPOSE_NO_DATA
- MAIL_COMPOSE_BAD_FORMAT
- MAIL_COMPOSE_SIZE_LIMIT
- MAIL_COMPOSE_INVALID_RCPTS
- MAIL_COMPOSE_SEND_LIMIT
- MAIL_FROM_ERROR

Before calling this method you should upload the mail to the server, otherwise mail_compose will take no action. Webmail Server recognizes a mail upload try by a query "action=compose". The form's enctype has to be "multipart/form-data" for the upload to be recognized.

Example 1:

`mail_edit.hsp:`

```hsp
<%h=getSessionId()%>
<form action="mail_compose.hsp?_h=<%h%>&action=compose" method="POST" enctype="multipart/form-data">
```

Copyright © 2008 Gecad Technologies S.A.

http://www.axigen.com
To:<input type="text" name = "To"><br>
From:<input type="text" name = "From"><br>
Cc:<input type="text" name = "Cc"><br>
Bcc:<input type="text" name = "Bcc"><br>
Subject:<input type="text" name = "Subject"><br>
<input type="hidden" name = "Content-Type" value = "text/plain"><br>
Body:<textarea name="Body"></textarea><br>
<input type="submit" name="send" value="Send">
</form>

Example 2:
mail_compose.hsp:

<%IF mail_compose(errFields, "compose")%>
    Success
<%ENDIF%>

If you want to attach a file to this mail you should follow the example (where query action=attach is used):
<form action="mail_compose?_h=<%h%>&action=attach" method="POST" enctype="multipart/form-data">
    File:<input type="file" name="file">
    <input type="submit" name="attach" value="Attach">
</form>

2.

<table>
<thead>
<tr>
<th>Name</th>
<th>mail_composeLight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP mail_composeLight(MAP composeInfo)</td>
</tr>
</tbody>
</table>
| Explanation   | Tries to send a mail. <composeInfo> contains:  
                           - From  
                           - Reply-To  
                           - To  
                           - Cc  
                           - Bcc  
                           - Subject  
                           - Body  
                           - action: save | compose | reply | replyall | forward  
                           - replyFid and replyMid: the mail replied  
                           If the returned MAP is empty, the mail was sent, otherwise you can identify bad fields:  
                           - toErr (exists only if to is wrong)  
                           - ccErr (exists only if cc is wrong)  
                           - bccErr (exists only if bcc is wrong) |

Copyright © 2008 Gecad Technologies S.A.
http://www.axigen.com
### Possible errors:
- INTERNAL_SERVER_ERROR
- MAIL_COMPOSE_SIZE_LIMIT
- MAIL_COMPOSE_INVALID_RCPTS
- MAIL_COMPOSE_SEND_LIMIT
- MAIL_FROM_ERROR

#### 3. mail_copy

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail_copy</td>
<td>mail_copy(NUMBER folderId, NUMBER mailId, NUMBER newFolderId)</td>
<td>Copies the mail with id <code>&lt;mailId&gt;</code>, from the folder with id <code>&lt;folderId&gt;</code>, to the folder with id <code>&lt;newFolderId&gt;</code>.</td>
</tr>
</tbody>
</table>

#### 4. mail_delete

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail_delete</td>
<td>mail_delete(NUMBER folderId, NUMBER mailId)</td>
<td>Deletes the mail identified by <code>&lt;mailId&gt;</code> and <code>&lt;folderId&gt;</code>.</td>
</tr>
</tbody>
</table>

#### 5. mail_move

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail_move</td>
<td>mail_move(NUMBER folderId, NUMBER mailId, NUMBER newFolderId)</td>
<td>Moves the mail identified by <code>&lt;mailId&gt;</code> and <code>&lt;folderId&gt;</code>, to the folder with id <code>&lt;newFolderId&gt;</code>.</td>
</tr>
</tbody>
</table>

#### 6. mail_setFlag

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail_setFlag</td>
<td>mail_setFlag(NUMBER folderId, NUMBER mailId, STRING flag)</td>
<td>Sets the <code>&lt;flag&gt;</code> for the mail (<code>&lt;folderId&gt;</code>, <code>&lt;mailId&gt;</code>); <code>&lt;flag&gt;</code> can be: &quot;seen&quot;, &quot;flagged&quot; or &quot;deleted&quot;. Any other values are ignored.</td>
</tr>
</tbody>
</table>

#### 7. mail_unsetFlag

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail_unsetFlag</td>
<td>mail_unsetFlag(NUMBER folderId, NUMBER mailId, PARAM flag)</td>
<td>Unsets the <code>&lt;flag&gt;</code> for the mail (<code>&lt;folderId&gt;</code>, <code>&lt;mailId&gt;</code>); <code>&lt;flag&gt;</code> can be: &quot;seen&quot;, &quot;flagged&quot; or &quot;deleted&quot;. Any other values are ignored.</td>
</tr>
</tbody>
</table>

#### 8. mail_loadComposeInfo

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail_loadComposeInfo</td>
<td>MAP mail_loadComposeInfo(NUMBER fid, NUMBER mid)</td>
<td>Loads some info, needed for composing mail (<code>&lt;fid&gt;,&lt;mid&gt;</code>). It returns a MAP object that contains:</td>
</tr>
</tbody>
</table>
### HSP – AXIGEN Server Side Scripting Language

- **STRING from**
- **STRING to**
- **STRING cc**
- **STRING bcc**
- **STRING replyto: used to fill "To" field in case of a "Reply" action**
- **STRING replytoall: used to fill "Cc" field in case of a "Reply To All" action**
- **STRING subject**

#### 9. Name: `mail_load`

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MAP mail_load(NUMBER folderId, NUMBER mailId)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation</td>
<td>Returns a MAP object that contains the headers of the mail (folderId, mailId). MAP object properties:</td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING from</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING subject</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING to</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING cc</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING bcc</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING date</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING dateFriendly</strong>: the date transformed in the user timezone; also the date has a short format ([hh:mm] if date is today, [WeekDay, dd Month] if date is this year and [dd Month yyyy] if date is not this year.</td>
</tr>
<tr>
<td></td>
<td>- <strong>VALUE seen</strong>: exists if the mail has the flag &quot;seen&quot;</td>
</tr>
<tr>
<td></td>
<td>- <strong>VALUE flagged</strong>: exists if the mail has the flag &quot;flagged&quot;</td>
</tr>
<tr>
<td></td>
<td>- <strong>VALUE deleted</strong>: exists if the mail has the flag &quot;deleted&quot;</td>
</tr>
<tr>
<td></td>
<td>- <strong>VALUE hasNext</strong>: exists if the mail has a successor</td>
</tr>
<tr>
<td></td>
<td>- <strong>VALUE hasPrev</strong>: exists if the mail has a predecessor</td>
</tr>
<tr>
<td></td>
<td>- <strong>NUMBER mid</strong>: mail id</td>
</tr>
<tr>
<td></td>
<td>- <strong>VALUE att</strong>: exists only if mail has attachments</td>
</tr>
</tbody>
</table>

#### 10. Name: `mail_loadList`

<table>
<thead>
<tr>
<th>Syntax</th>
<th>ARRAY mail_loadList(NUMBER folderId[, NUMBER startIndex])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation</td>
<td>Returns the list of mails from the folder with id &lt;folderId&gt;. The first mail from the list has &lt;startIndex&gt;. Every element has MAP type and properties:</td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING subject</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING from</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING to</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING date</strong></td>
</tr>
<tr>
<td></td>
<td>- <strong>STRING dateFriendly</strong>: the date transformed in the user timezone; also the date has a short format ([hh:mm] if date is today, [WeekDay, dd Month] if date is this year and [dd Month yyyy] if date is not this year.</td>
</tr>
</tbody>
</table>
HSP – AXIGEN Server Side Scripting Language

- NUMBER size: mail size
- NUMBER id: mail index
- NUMBER mid: mail ID
- VALUE seen: exists if the mail has the flag "seen"
- VALUE flagged: exists if the mail has the flag "flagged"
- VALUE deleted: exists if the mail has the flag "deleted"
- STRING hasNext: exists if the mail has a successor
- VALUE hasPrev: exists if the mail has a predecessor
- VALUE att: exists only if mail has attachments

11.

<table>
<thead>
<tr>
<th>Name</th>
<th>mail_getSearchFolderId</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>NUMBER mail_getSearchFolderId()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns the id of the temporary search folder created after a search operation. Before calling this method you should submit a search request through the Webmail Server (action=search).</td>
</tr>
</tbody>
</table>

Example:

```html
<form action="search.hsp?_h=<%h%>&fid=<%fid>&action=search" method="POST">
    <input type = "text" name = "from" value = "john">
    <input type = "text" name = "subject[]" value = "hello">
    <input type = "text" name = "subject[]" value = "you">
    <input type = "submit">
</form>
```

After this request, the server will search all the mails from folder <fid>, that matches the inputted <subject>. A search action can be done using the following searching keys:
- match ("any"|"all"): applies any/all of following criterias:
- before ("dd-mm-yyyy"): searches messages whose internal date (disregarding time and timezone) is earlier than the specified date.
- sentbefore ("dd-mm-yyyy"): searches messages whose "Date" header (disregarding time and timezone) is earlier than the specified date.
- on ("dd-mm-yyyy"): searches messages whose internal date (disregarding time and timezone) is within the specified date.
- senton ("dd-mm-yyyy"): searches messages whose "Date" header (disregarding time and timezone) is within the specified date.
- since ("dd-mm-yyyy"): searches messages whose internal date (disregarding time and timezone) is within or later than the specified date.
- sentsince ("dd-mm-yyyy"): searches messages whose "Date" header (disregarding time and timezone) is within or later than the specified date.
- subject[] (string): searches messages that contain the specified string in the "Subject" header; [] means that there can be more than one subject search criteria.
- notsubject[] (string): searches messages that don't contain the specified string in the "Subject" header.
- from[] (string): searches messages that contain the specified string in the "From" header.
- notfrom[] (string): searches messages that don't contain the specified string in the "From" header.
- to[] (string): searches messages that contain the specified string in the "To" header.

Copyright © 2008 Gecad Technologies S.A.
http://www.axigen.com
HSP – AXIGEN Server Side Scripting Language

- `notto[]` (string): searches messages that don't contain the specified string in the "To" header.
- `cc[]` (string): searches messages that contain the specified string in the "CC" header.
- `notcc[]` (string): searches messages that don't contain the specified string in the "CC" header.
- `body[]` (string): searches messages that contain the specified string in the body of the message.
- `notbody[]` (string): searches messages that don't contain the specified string in the body of the message.
- `text[]`: searches messages that contain the specified string in the header or body of the message.
- `nottext[]` (string): searches messages that don't contain the specified string in the header or body of the message.
- `header ("Header:hello")`: searches messages that contain the specified string in the specified <Header> header.
- `notheader ("Header:hello")`: searches messages that don't contain the specified string in the specified <Header> header.
- `larger (size)`: searches messages larger than specified size.
- `smaller (size)`: searches messages smaller than specified size.
- `deleted ("yes"|"no")`: searches deleted/undeleted messages.
- `seen ("yes"|"no")`: searches seen/unseen messages.
- `flagged ("yes"|"no")`: searches flagged/unflagged messages.
- `att[]` (string): searches messages that contain the specified string in the attachment filenames; [] means that there can be more than one att search criteria.
- `notatt[]` (string): searches messages that don't contain the specified string in the attachment filenames.
- `anywhere[]`: searches messages that contain the specified string in the header, body or the attachments name of the message.
- `notanywhere[]` (string): searches messages that don't contain the specified string in the header, body or attachments name of the message.
- `quicksearch[]`: searches messages that contain the specified string in the subject, from, to, cc, body or the attachments name of the message.

### 12. `mail_sendBinary`

<table>
<thead>
<tr>
<th>Name</th>
<th>mail_sendBinary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>mail_sendBinary(NUMBER folderId, NUMBER mailId)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Sends original mail (headers + body) in text/plain (RAW) format.</td>
</tr>
</tbody>
</table>

**WARNING:** Don't write any other HTML or HSP code before this method.

### 13. `mail_sendHeaders`

<table>
<thead>
<tr>
<th>Name</th>
<th>mail_sendHeaders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>mail_sendHeaders(NUMBER folderId, NUMBER mailId)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Sends only the headers of mail (&lt;folderId&gt;, &lt;mailId&gt;).</td>
</tr>
</tbody>
</table>

**WARNING:** Don't write any other HTML or HSP code before this method.

### 14. `body_loadParts`

<table>
<thead>
<tr>
<th>Name</th>
<th>body_loadParts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>ARRAY body_loadParts(NUMBER folderId, NUMBER mailId[, STRING action])</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns all body parts of the mail (&lt;folderId&gt;, &lt;mailId&gt;). Every element has MAP type and properties:</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>- NUMBER bid: body id</td>
</tr>
<tr>
<td></td>
<td>- STRING type: mime type</td>
</tr>
<tr>
<td></td>
<td>- STRING subtype: mime subtype</td>
</tr>
<tr>
<td></td>
<td>The following properties are available only for message bodies:</td>
</tr>
<tr>
<td></td>
<td>- STRING from</td>
</tr>
<tr>
<td></td>
<td>- STRING subject</td>
</tr>
<tr>
<td></td>
<td>- STRING to</td>
</tr>
<tr>
<td></td>
<td>- STRING date</td>
</tr>
</tbody>
</table>

### 15.

<table>
<thead>
<tr>
<th>Name</th>
<th>body_show</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>body_show(NUMBER folderId, NUMBER mailId, NUMBER bodyId, MAP showInfo)</td>
</tr>
<tr>
<td>Explanation</td>
<td>This method will display the content of the body identified by &lt;bodyId&gt;, &lt;mailId&gt; and &lt;folderId&gt;. &lt;bodyId&gt; can be obtained with body_loadParts() method. Only text, html or multipart/related bodies are displayed. &lt;typeShow&gt; is a MAP object and can have the following properties:</td>
</tr>
<tr>
<td></td>
<td>- STRING type: specifies what kind of bodies to display:</td>
</tr>
<tr>
<td></td>
<td>- &quot;text&quot;: the body will be displayed only if it is TEXT/nonHTML</td>
</tr>
<tr>
<td></td>
<td>- &quot;html&quot;: displays a body if it's of TEXT/HTML type. It's parsed and securely regenerated.</td>
</tr>
<tr>
<td></td>
<td>- STRING indent</td>
</tr>
<tr>
<td></td>
<td>- &quot;yes&quot;: it is used to display de body indented, on a reply</td>
</tr>
<tr>
<td></td>
<td>- &quot;no&quot;: normal display</td>
</tr>
<tr>
<td></td>
<td>- STRING graphics</td>
</tr>
<tr>
<td></td>
<td>- &quot;yes&quot;: it is used to enable all external resources</td>
</tr>
<tr>
<td></td>
<td>- &quot;no&quot;: it is used to disable all external resources</td>
</tr>
<tr>
<td></td>
<td>- STRING filter: if missing, filter level is taken from webmail options</td>
</tr>
<tr>
<td></td>
<td>- &quot;xhtml&quot;</td>
</tr>
<tr>
<td></td>
<td>- &quot;tidy&quot;</td>
</tr>
<tr>
<td></td>
<td>- &quot;strict&quot;</td>
</tr>
<tr>
<td></td>
<td>- &quot;text&quot;</td>
</tr>
<tr>
<td></td>
<td>After parsing a html body, if there are any external graphics then &quot;hasGraphics&quot; property is added to &lt;showInfo&gt;.</td>
</tr>
</tbody>
</table>

### 16.

<table>
<thead>
<tr>
<th>Name</th>
<th>att_loadList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY att_loadList(NUMBER folderId, NUMBER mailId)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns attachment list of the mail identified by &lt;mailId&gt; and &lt;folderId&gt;. Every element has MAP type and the following properties:</td>
</tr>
</tbody>
</table>
17. **Name** | att_send  
---|---  
**Syntax** | att_send(NUMBER folderId, NUMBER mailId, NUMBER attId, STRING download)  
**Explanation** | Sends the content of the attachment identified by <attid>, <mailId> and <folderId>.  
**WARNING:** This method MUST be called alone, without any other HTML or HSP code, because it sends binary data (ex: images files) to the client and resets HTTP headers.

18. **Name** | att_sendCid  
---|---  
**Syntax** | att_sendCid(NUMBER folderId, NUMBER mailId, NUMBER bodyId, NUMBER cid)  
**Explanation** | Sends the content of an inline attachment identified by content id <cid>, <attid>, <mailId> and <folderId>. It is used only for bodies with multipart/related types.  
**WARNING:** This method MUST be called alone, without any other HTML or HSP code, because it sends binary data (ex: images files) to the client and resets HTTP headers.

19. **Name** | upatt_add  
---|---  
**Syntax** | upatt_add()  
**Explanation** | Tries to add an attachment to a composed message. Possible errors:  
- INTERNAL_SERVER_ERROR  
- UPATT_NO_DATA  
- UPATT_COUNT_LIMIT  
- UPATT_SIZE_LIMIT  
- MESSAGE_SIZE_LIMIT  
Before calling this method you should upload the attachment to the server.

Example:
```html
<form action="upatt_add?_h=<%h%>&action=attach" method="POST" enctype="multipart/form-data">   
  File:<input type="file" name="file">
  
  <input type="submit" name="attach" value="Attach">
</form>
```
### Account specific methods

#### 1.

<table>
<thead>
<tr>
<th>Name</th>
<th>account_loadInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP account_loadInfo()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns a MAP object with following properties (account's informations):</td>
</tr>
<tr>
<td></td>
<td>- STRING accountName</td>
</tr>
<tr>
<td></td>
<td>- STRING domainName</td>
</tr>
<tr>
<td></td>
<td>- NUMBER totalQuota</td>
</tr>
<tr>
<td></td>
<td>- NUMBER remainingQuota</td>
</tr>
<tr>
<td></td>
<td>- NUMBER usedQuota</td>
</tr>
<tr>
<td></td>
<td>- NUMBER overQuotaThreshold</td>
</tr>
<tr>
<td></td>
<td>- NUMBER maxRpopConnections</td>
</tr>
<tr>
<td></td>
<td>- NUMBER minRpopRetrievalInterval</td>
</tr>
<tr>
<td></td>
<td>- NUMBER minHtmlFilterLevel</td>
</tr>
<tr>
<td></td>
<td>- NUMBER folderContainerId</td>
</tr>
<tr>
<td></td>
<td>- STRING accountType: user, maillist or postmaster</td>
</tr>
</tbody>
</table>
### 2. 
**Name**: account_getType  
**Syntax**: STRING account_getType()  
**Explanation**: Returns the type of the account ("user" or "mlist").

### 3. 
**Name**: account_updatePassword  
**Syntax**: account_updatePassword(STRING oldPasswd, STRING newPasswd)  
**Explanation**: Changes the account's password.

### 4. 
**Name**: account_loadWebmailData  
**Syntax**: MAP account_loadWebmailData()  
**Explanation**: Returns a MAP object with following properties (account's Webmail data):  
- NUMBER pageSize: number of mails displayed on a page  
- STRING skin: user's webmail skin  
- VALUE confirmMailDelete: option for confirmation before mail deletion  
- VALUE confirmFolderEmpty: option for confirmation before empty a folder  
- VALUE saveToSent: option for saving a mail to "Sent" folder after send action  
- VALUE deleteToTrash: option for moving mail to "Trash" folder after delete action  
- NUMBER autoRefreshInterval  
- VALUE hideDeletedMessages  
- NUMBER htmlFilterLevel: can have values from 0 up to 3  
- STRING language  
- STRING signature  
- STRING timeZone  
- VALUE usePublicContacts  
- VALUE useRcptContacts  
- STRING weekStartDay

### 5. 
**Name**: account_updateWebmailData  
**Syntax**: MAP account_updateWebmailData(MAP webmailData)  
**Explanation**: Updates the account's webmail data.
# User specific methods

1. **Name** | user_getSenderNotInAB  
**Syntax** | INTEGER user_getSenderNotInAB()  
**Explanation** | Returns an integer that describes the action taken when the user receives an email message from somebody that is not in his address book. Supported values:  
- PASS = 0  
- SEND_NDR = 1  
- MOVE_TO_TRASH = 2  
- DISCARD = 3

2. **Name** | user_setSenderNotInAB  
**Syntax** | user_setSenderNotInAB(INTEGER value)  
**Explanation** | Sets the action taken when the user receives an email message from somebody that is not in his address book. See user_getSenderNotInAB for supported values.

3. **Name** | user_loadContactInfo  
**Syntax** | MAP user_loadContactInfo()  
**Explanation** | Returns a MAP object with following properties (account's contact information):  
- STRING firstName  
- STRING lastName  
- STRING nickName  
- STRING personalEmail  
- STRING businessEmail  
- STRING phone  
- STRING mobilePhone  
- STRING homePhone  
- STRING businessPhone  
- STRING homeAddress  
- STRING businessAddress

4. **Name** | user_updateContactInfo  
**Syntax** | MAP user_updateContactInfo(MAP personalData)  
**Explanation** | Updates account's contact information with the properties in <contactInfo>. The MAP returned by this method contains all wrong properties from <contactInfo> in case updating fails.
Maillist specific methods

1. Name: `mlist_getDescription`
   Syntax: `STRING mlist_getDescription()`
   Explanation: Returns the description of the mail list.

2. Name: `mlist_setDescription`
   Syntax: `mlist_setDescription(STRING description)`
   Explanation: Sets the description for the current mail list.

Domain specific methods

1. Name: `domain_loadList`
   Syntax: `ARRAY domain_loadList()`
   Explanation: Loads the list of domains that have `showWebmailLogin = "yes"` if `showDomainList = "yes"` in Webmail general settings.

2. Name: `domain_loadSPEPolicy`
   Syntax: `MAP domain_loadSPEPolicy()`
   Explanation: Returns the current domain password enforcement policy:
   - `STRING enabled(yes|no)`
   - `NUMBER minimumLength: minimum password length`
   - `NUMBER maximumLength: maximum password length`
   - `STRING requiredCharacters (letters | lettersAndNumbers | lettersAndNumbersAndSpecial)`

Public folder specific methods

1. Name: `publicFolder_getName`
   Syntax: `STRING publicFolder_getName()`
   Explanation: Returns the name of the public folder.

Connection specific methods

1. Name: `connection_isSecure`
   Syntax: `connection_isSecure()`
   Explanation: Returns true if the current connection is secure (SSL) or false otherwise.
## Name

### connection_getHostNameResolver

**Syntax**

```plaintext
STRING connection_getHostNameResolver()
```

**Explanation**

Returns the domain name of the host. "Host" HTTP header MUST be sent by the client in order to resolve host name to domain name.

## Name

### connection_getUrlRedirect

**Syntax**

```plaintext
STRING connection_getUrlRedirect()
```

**Explanation**

Returns an url redirection string for the current connection which is used to redirect from a plain to a secure connection.

## RPOP specific methods

### Name

### rpopConnection_loadList

**Syntax**

```plaintext
ARRAY rpopConnection_loadList()
```

**Explanation**

Returns the list of rpop connections defined by the user. Every element is a MAP and contains the following properties:

- STRING hostname
- STRING username
- NUMBER port

**NOTE:** `<rpopConnectionKey>` is a MAP with hostname, username and port properties.

### Name

### rpopConnection_load

**Syntax**

```plaintext
MAP rpopConnection_load(MAP connKey)
```

**Explanation**

Loads the details `<rpopConnection>` of the rpop connection identified by `<rpopConnectionKey>` and contains:

- STRING hostname
- STRING username
- NUMBER port
- STRING password
- NUMBER retrievalInterval
- STRING folderName
- STRING encryption (none|ssl|tls)
- STRING enableAPOP (yes|no)
- STRING deleteOnRetrieval (yes|no)

### Name

### rpopConnection_add

**Syntax**

```plaintext
MAP rpopConnection_add(MAP connInfo)
```

**Explanation**

Adds `<rpopConnection>`. The MAP returned by this method contains all wrong properties from `<rpopConnection>` if adding failed.

Copyright © 2008 Gecad Technologies S.A.

http://www.axigen.com
4.

<table>
<thead>
<tr>
<th>Name</th>
<th>rpopConnection_update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP rpopConnection_update(MAP connKey, MAP connInfo)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Updates the rpop connection identified by &lt;rpopConnectionKey&gt; with &lt;rpopConnection&gt; values. The MAP returned by this method contains all wrong properties from &lt;rpopConnection&gt; if updating failed.</td>
</tr>
</tbody>
</table>

5.

<table>
<thead>
<tr>
<th>Name</th>
<th>rpopConnection_remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>rpopConnection_remove(MAP connKey)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Removes the rpop connection identified by &lt;rpopConnectionKey&gt;.</td>
</tr>
</tbody>
</table>

Grouping specific methods

1.

<table>
<thead>
<tr>
<th>Name</th>
<th>thread_loadList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY thread_loadList(NUMBER folderId[, NUMBER startId[, NUMBER count]])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads the list of threads from de folder &lt;fid&gt;, starting with index &lt;startId&gt;. The number of threads is passed through &lt;count&gt;. Every element of list is a MAP and contains: STRING subject (the key of the thread) This method should be called after a call to folder_setGroupType() for the folder on which thread_loadList() acts.</td>
</tr>
</tbody>
</table>

2.

<table>
<thead>
<tr>
<th>Name</th>
<th>thread_load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY thread_load(NUMBER folderId, STRING subject)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads information about all mails from the thread identified by &lt;subject&gt;, folder &lt;fid&gt;. The list contains MAPs with properties: NUMBER id: mail id NUMBER level: level in tree</td>
</tr>
</tbody>
</table>

3.

<table>
<thead>
<tr>
<th>Name</th>
<th>group_loadList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY group_loadList(NUMBER folderId[, NUMBER startId[, NUMBER count]])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns the list of groups from folder &lt;fid&gt;; the first in list is &lt;startIndex&gt;. The number of groups is passed through &lt;count&gt;. Every group (MAP) contains: STRING key This method should be used after folder_setGroupType() was called.</td>
</tr>
</tbody>
</table>
HSP – AXIGEN Server Side Scripting Language

4.

<table>
<thead>
<tr>
<th>Name</th>
<th>group_load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY group_load(NUMBER folderId, STRING groupKey)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns informations about all mails from the group identified by &lt;groupKey&gt;, from folder &lt;fid&gt;. &lt;groupBy&gt; specifies the group type: &quot;subject&quot; or &quot;from&quot;. Every element from list contains:</td>
</tr>
<tr>
<td></td>
<td>− NUMBER id: mail id</td>
</tr>
</tbody>
</table>

Calendar Specific Methods

Data structures (MAPS):

**WEBDATE**
- NUMBER year - allowed values: 1970-2037
- NUMBER month - allowed values: 1-12
- NUMBER day - allowed values: 1-31
- NUMBER hour - allowed values: 0-23
- NUMBER min - allowed values: 0-59

**OBJECTINFO**: full description of calendar object
- NUMBER id - id of the object used for delete/update operations
- NUMBER recurid - recurring id. If 0 means there is no recurrence or recurring master object
- FLAG isVirtual - readonly - means this is a new recurring instance. Id is set to the master object. This object is not yet saved and an update operation is needed on this to make it persistent. After update this object will have a new id
- STRING summary - summary of the event
- WEBDATE startDate - starting date of the event. Might be missing for tasks
- WEBDATE endDate - end/due date of the event/task. Journal doesn't have this property.
- STRING type - can be one of (event, task, journal)
- STRING description - full description of the event
- STRING location - location of the event
- STRING status - readonly - status of the event can be one of:
  1. "need-action"
  2. "completed" - valid only in task
  3. "in-process" - valid only in task
  4. "cancelled"
  5. "draft" - valid only in journal
  6. "final" - valid only in journal
  7. "confirmed"
  8. "tentative"
- FLAG isAllDayEvent - Only appears for events. If set means that only year,month,day in the startDate/endDate map
- STRING email - Email address of the organizer
- STRING name - Email address of the organizer
- MAP emailReminder: Only valid for events and tasks
  - STRING description - Text to appear in the description of the reminder (not used right now)
STRING type - this is the trigger type which can be absolute or relative. "Relative" can only be set for events and absolute can only be set for tasks.

WEBSITE/NUMBER trigger - This is a WEBSITE if type is absolute. If type is relative then the trigger represents the number of minutes before the start(for the event) when the popup will be shown

VECTOR(STRING) attendees - A list of email addresses (xxx@xxx) where the notification emails will be sent.

- MAP displayReminder
  - STRING description - Text associated with this reminder (not used)
  - STRING type - Same as for email reminders
  - WEBSITE/NUMBER trigger - same as for email reminders

- VECTOR(MAP) attendees - The attendee list. The item MAP is described below
  - STRING name - its the name of the attendee (Ex: John Doe)
  - STRING email - its the email address of the attendee (ex: john.doe@lost.com)
  - STRING status - Readonly - Its the acceptance status of the attendee. Can be one of the following values
    1. "need-action" - user didnt take any action yet
    2. "accepted" - user accepted the invite
    3. "declined" - user declined the invite
    4. "tentative" - user tentatively accepted the invite
    5. "delegated" - user was delegated
    6. "completed" - Only valid for task. User completed its task
    7. "in-process" - Only valid for task.
    8. "unknown" - This should not happen anyway

- WEBSITE completedDate - optional - Only valid for task. Date when task was completed.

- NUMBER priority - Only valid for task. Priority of the task. Default is 0 and growing

- NUMBER percentComplete - Only valid for task.

- NUMBER duration - Only valid for journal. Minutes representing duration of the journal

- NUMBER category - Only valid for journal. Category string

- MAP recurrency - Only appears if object is recurrent and can only be set for the master object. Instances of recurring objects cannot be recurrent.
  - NUMBER count - optional - Maximum number of recurring occurances
  - WEBSITE until - optional - Last date of a recurring occurrence. Count and Until cannot be simultaniously present. In this acase then until has priority over count.
  - NUMBER period - default 1, optional - The skip interval (ex. every 2 days)
  - STRING type - Type of recurrence and can be one of:
    ▪ "daily"
    ▪ "weekly"
    ▪ "monthly"
    ▪ "yearly"

- VECTOR(STRING) weekMask - optional - Used for daily specification. Contains a list of days when occurrence can take place. Possible values are ("sun", "mon", "tue","wed","thu","fri","sat"). If missing then no restrictions are assumed (all days)

- VECTOR(STRING) weekly - Required when type is weekly. Contains list of days of week when is possible to have occurrences. Possible values are ("sun", "mon", "tue","wed","thu","fri","sat"). If missing then no restrictions are assumed (all days)

- MAP monthly - It is used for monthly recurrences. If monthDay excludes (weekDay, occurence)
HSP – AXIGEN Server Side Scripting Language

- **STRING type** - Type of monthly recurrence. Can be one of "byMonthDay" or "byWeekDay". If "byMonthDay" is specified then "monthDay" must be present. If "byWeekDay" is specified then "weekDay" and "occurence" must be specified.
- **NUMBER monthDay** - Day of month (1-31)
- **VECTOR(STRING) weekMask** - Mask of days. The item value can be one of ("sun", "mon", "tue", "wed", "thu", "fri", "sat")
- **STRING occurence** - Possible values: ("first", "second", "third", "forth", "last")

MAP yearly

- **STRING type** - Type of monthly recurrence and can be one of "byMonthDay" or "byWeekDay". If "byMonthDay" is specified then "month" and "monthDay" must be present. If "byWeekDay" is specified then "month", "weekDay" and "occurence" must be present.
- **STRING month** - Month when occurance appears. Can be one of "january", "february", "march", "may", "april", "june", "july", "august", "september", "october", "november", "december"
- **NUMBER monthDay** - Day of month (1-31)
- **VECTOR(STRING) weekMask** - Mask of days. The item value can be one of ("sun", "mon", "tue", "wed", "thu", "fri", "sat")
- **STRING occurence** - Can be one of: ("first", "second", "third", "forth", "last")
- **STRING transparency** - Can be one of: ("opaque", "transparent")

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY(MAP) event_loadList(NUMBER folderId [, MAP resolutionTime])</td>
</tr>
</tbody>
</table>

**Explanation**

Returns an array of events from folder <folderId> for interval <resolutionTime>. resolutionTime (MAP):

- **NUMBER year** - allowed values are: 1970-2037
- **NUMBER month** - allowed values are: 1-12
- **NUMBER weekOrDay** - If this is a week then allowed values are 0-5 and represent the number of the week within the month. Week 0 means the week that contains the 1st day of the month. If this is a day then allowed value is 1-31

Each element in returned array consists of following map:

- **MAP startDate** - start date of event or recurring instance.
- **MAP endDate** - end date of event or recurring instance. If isAllDayEvent=true this is the inclusive end of the event.
- **STRING summary** - summary field for the event
- **NUMBER id** - storage object id used for reference for further ops
- **NUMBER recurid** - recurring instance, 0 - means no recurrence
- **FLAG isAllDayEvent** - true if object is all day.
<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_load</td>
<td>MAP event_load(NUMBER folderId, NUMBERobjectId [, NUMBER recurrenceId])</td>
<td>Loads event &lt;objectId&gt; from folder &lt;folderId&gt;. If need to load an instance of recurring event specify instance time in &lt;recurrenceId&gt;</td>
</tr>
<tr>
<td>event_add</td>
<td>event_add(NUMBER folderId, MAP object)</td>
<td>Adds &lt;object&gt; to folder &lt;folderId&gt;.</td>
</tr>
<tr>
<td>event_update</td>
<td>event_update(NUMBER folderId, OBJECTINFO objectInfo, NUMBER objectId [, NUMBER recurrenceId])</td>
<td>Modifies object with &lt;objectId&gt; and &lt;recurrenceId&gt; with info from &lt;objectInfo&gt;. If flag isVirtual is set in the MAP then this represents an object instance, objectId represents the id of the master object.</td>
</tr>
<tr>
<td>event_remove</td>
<td>event_remove(NUMBER folderId, NUMBER objectId [, NUMBER recurrenceId])</td>
<td>Removes object with &lt;objectId&gt; event from folder &lt;folderId&gt;. If the object is a recurring object and &lt;recurrenceId&gt; = 0 then removes the object and all saved instances. If object is recurring object and &lt;recurrenceId&gt; is not null then deletes a certain instance.</td>
</tr>
<tr>
<td>event_copy</td>
<td>event_copy(NUMBER folderId, NUMBER objectId, NUMBER newFolderId)</td>
<td>Copies object with &lt;objectId&gt; event from folder &lt;folderId&gt; to &lt;newFolderId&gt;.</td>
</tr>
<tr>
<td>event_move</td>
<td>event_move(NUMBER folderId, NUMBER objectId, NUMBER newFolderId)</td>
<td>Moves object with &lt;objectId&gt; event from folder &lt;folderId&gt; to &lt;newFolderId&gt;.</td>
</tr>
</tbody>
</table>
## Tasks

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| task_loadList | ARRAY(MAP) task_loadList(NUMBER folderId)   | Returns an array of events from folder `<folderId>` filtered with the folder saved filtering type and sorted by due date. If no due date is present events are returned first. Each element in returned array consists of following map:  
  - MAP `startDate` - optional - start date of event or recurring instance  
  - MAP `endDate` - optional - end date of event or recurring instance  
  - STRING `summary` - summary field for the event  
  - NUMBER `id` - storage object id used for reference for further ops  
  - NUMBER `recurid` - recurring instance, 0 - means no recurrence  
  - NUMBER `percentComplete` - this is the percent complete for the task |
| task_load  | MAP task_load(NUMBER folderId, NUMBER objectId [, NUMBER recurrenceId]) | Loads task `<objectId>` from folder `<folderId>`. If need to load an instance of recurring task specify instance time in `<recurrenceId>`. |
| task_add   | task_add(NUMBER folderId, OBJECTINFO objectInfo) | Adds an task to folder |
| task_update | task_update(NUMBER folderId, OBJECTINFO objectInfo, NUMBER objectId [, NUMBER recurrenceId]) | Modifies object with `<objectId>` and `<recurrenceId>` with info from `<objectInfo>`. If flag isVirtual is set in the MAP then this represents an object instance, objectId represents the id of the master object. |
| task_remove | task_remove(NUMBER folderId, NUMBER objectId [, NUMBER recurrenceId]) | Removes object with `<objectId>` event from folder `<folderId>`. If the object is a recurring object and `<recurrenceId> = 0` then removes the object and all saved |
instances. If object is recurring object and `<recurrenceId>` is not null then deletes a certain instance.

<table>
<thead>
<tr>
<th>Name</th>
<th>task_copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>task_copy(NUMBER folderId, NUMBER objectId, NUMBER newFolderId)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Copies object with <code>&lt;objectId&gt;</code> event from folder <code>&lt;folderId&gt;</code> to <code>&lt;newFolderId&gt;</code>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>task_move</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>task_move(NUMBER folderId, NUMBER objectId, NUMBER newFolderId)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Moves object with <code>&lt;objectId&gt;</code> event from folder <code>&lt;folderId&gt;</code> to <code>&lt;newFolderId&gt;</code>.</td>
</tr>
</tbody>
</table>

**Journal**

<table>
<thead>
<tr>
<th>Name</th>
<th>journal_loadList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>ARRAY(MAP) journal_loadList(NUMBER folderId [, MAP resolutionTime])</code></td>
</tr>
</tbody>
</table>
| Explanation| Returns an array of journals from folder `<folderId>` for interval `<resolutionTime>`. resolutionTime (MAP):

- NUMBER `year` - allowed values are: 1970-2037
- NUMBER `month` - allowed values are: 1-12
- NUMBER `weekOrDay` - If this is a week then allowed values are 0-5 and represent the number of the week within the month. Week 0 means the week that contains the 1st day of the month. If this is a day then allowed value is 1-31

Each element in returned array consists of following map:

- MAP `startDate` - start date of event or recurring instance
- STRING `summary` - summary field for the event
- NUMBER `id` - storage object id used for reference for further ops
- NUMBER `recurid` - always 0 (not used)

<table>
<thead>
<tr>
<th>Name</th>
<th>journal_load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>OBJECTINFO journal_load(NUMBER folderId, NUMBER objectId)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads journal <code>&lt;objectId&gt;</code> from folder <code>&lt;folderId&gt;</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>journal_add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>journal_add(NUMBER folderId, OBJECTINFO objectInfo)</code></td>
</tr>
</tbody>
</table>

Copyright © 2008 Gecad Technologies S.A.

http://www.axigen.com
### Explanation

**Name**: journal_update  
**Syntax**: `journal_update(NUMBER folderId, OBJECTINFO objectInfo, NUMBER objectId)`  
**Explanation**: Modifies object with `<objectId>` with info from `<objectInfo>`.

### Inbox processing

**Name**: calAction_get  
**Syntax**: `MAP calAction_get(NUMBER folderId, NUMBER mailId)`  
**Explanation**: Analyses the message from itip point of view and returns a map containing info used to display information to use. Return value map:

- **STRING title** - its a string describing the result of the analyze. Possible values for this are:
  1. "UNDEFINED" - This should not happen  
  2. "UNSUPPORTED" - The receivedObject cannot be processed because is not calendar compliant and has some unsupported features for the server  
  3. "PUBLISH_UPDATE" - organizer->others - receivedObject is an update for the `<existingObject>`. The following actions are associated with this code:
     - ACTION_ACCEPT  
  4. "PUBLISH_NEW" - organizer->others - receivedObject is a new object for this attendee/assignee. A published event means attendees are readonly. They can accept the event but no reply is sent to the organizer. The following actions are associated with this code:
     - ACTION_ACCEPT  
     - ACTION_REJECT  
  5. "REQUEST_RESCHEDULE" - organizer->attendee - receivedObject is an update for the `<existingObject>`. The existingObject was changed in a critical way which affects the planning. The following actions are associated with this code:
     - ACTION_ACCEPT  
     - ACTION_TENTATIVE  
     - ACTION_REJECT
6. "REQUEST_UPDATE" - organizer->attendee - 
   receivedObject is an update of the <existingObject>. The existingObject was changed 
in a non critical way which doesnt affect its 
planning. The following actions are associated with 
this code:
   - ACTION_ACCEPT

7. "REQUEST_NEW" - organizer->attendee - 
   receivedObject is a new calendar object received. 
The following actions are associated with this code:
   - ACTION_ACCEPT
   - ACTION_TENTATIVE
   - ACTION_REJECT
   - ACTION_PROPOSE

8. "REPLY" - attendee->organizer - receivedObject is 
a reply announcing attendee participation status ( 
for task also announces its percent complete) This 
message is processed automatically and no actions 
are defined.

9. "REPLY_INVALID_REQUEST", - attendee- 
   >organizer: - the operation requested by the 
   organizer could not be processed by the attendee 
   and this is a notification.

10. "CANCEL" - organizer->attendee - The organizer 
    has canceled an object, master recurring object or 
    instance of a recurring object. The following 
    actions are associated with this code:
   - ACTION_ACCEPT

11. "REFRESH" - attendee->organizer - Attendee 
    requested an update for an object. The following 
    actions are associated with this code:
   - ACTION_ACCEPT

12. "COUNTER" - attendee->organizer - The 
    receivedObject contains proposal information from 
    the attendee. The following actions are associated 
    with this code:
   - ACTION_ACCEPT
   - ACTION_REJECT
   - ACTION_PROPOSE - if organizer 
     whats to change the 
     receivedObject and send an 
     update to attendee

13. "DECLINECOUNTER", - organizer->attendee - 
    Organizer rejected the proposal from the attendee. 
The following actions are associated with this code:
   - ACTION_ACCEPT

14. "OBSOLETE" - system - It is an error code 
    announcing that the receivedObject has a 
    sequence number smaller than the existingObject 
    and cannot be processed.

15. "TOONEW" - system - It is an error code
announcing that the receivedObject has a sequence number greater than the existingObject and cannot be processed.

16. "SECURITY_ROLEERROR" - system - it is an error code announcing that the received object has a security error which can be: the from mail is not attendee or organizer, the receiver is not organizer or attendee depending on context.

17. "CANNOTVIEW" - system - error nouncing that this email object cannot be processed as an itip email. Occures when trying to click an email from search folder.

18. "CANNOTVIEWPF" - system - error nouncing that this email object cannot be processed as an itip email in a public folder.

19. "NORIGHTS" - system - error happens when user doesnt have permissions to access the default calendar of this itip message.

- VECTOR (string) actions - contains the possible actions that can be taken by the user:
  1. "ACCEPT"
  2. "ACCEPT_TENTATIVE"
  3. "REJECT"
  4. "PROPOSE"

- "READONLY" - this is not a possible action but rather a status. Instead of buttons a message should be displayed to announce the user he doesnt have write permissions to the personal calendar/task folder

- "READONLY_ITIP" - this is not a possible action but rather a status. Instead of buttons a message should be displayed to announce the user he doesnt have permissions to remove the itip email after processing

- OBJECTINFO receivedObject - contains the object received from email

- OBJECTINFO existingObject - optional - contains the object existing in storage with the same UID as the one received

- STRING calendarFid - container the fid of the calendar mbox that will be used when action will take place

<table>
<thead>
<tr>
<th>Name</th>
<th>calAction_take</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>calAction_take(NUMBER folderId, NUMBER mailId, STRING action [, OBJECTINFO proposedInfo])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Call this when user presses one on the action buttons to actually process &lt;folderId, mailId&gt; with &lt;action&gt;. If action is ACTION_PROPOSE then a &lt;proposedInfo&gt; map must be specified.</td>
</tr>
</tbody>
</table>
### Popup Reminders

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| reminder_loadList     | ARRAY(MAP) reminder_loadList()             | Returns a list of calendar object (task/events) which have a reminder that has expired. The returned map has the following structure:
|                       |                                             | - NUMBER id - id of the object
|                       |                                             | - NUMBER recurid - the recurring id of the object
|                       |                                             | - WEBDATE startDate - optional - start date of the event
|                       |                                             | - WEBDATE endDate - end/due date of the event
|                       |                                             | - STRING summary - summary of the object
|                       |                                             | - STRING type - type of the entry. Can be one of ("event", "task")                                                                  |

### Notes

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| note_loadList         | ARRAY(MAP) note_loadList(NUMBER fid[, NUMBER startIdx]) | Returns the stored notes list form <fid> folder, starting with <startIdx> index. The number of notes is maximum <pageSize> from webmail options. The returned map is:
|                       |                                             | - NUMBER mid - mail id
|                       |                                             | - STRING note - summary of note
|                       |                                             | - WEBDATE date - date of note

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>note_show</td>
<td>note_show(NUMBER fid, NUMBER mailId)</td>
<td>Shows &lt;mailId&gt; note from &lt;fid&gt; folder.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>note_add</td>
<td>note_add(NUMBER fid)</td>
<td>Adds a note to &lt;fid&gt; folder. Before calling this method you should upload the note to the server.</td>
</tr>
</tbody>
</table>

### Example:

```%
<h=getSessionId()>
```
<form action="note_add.hsp?_h=<%h%>&action=compose" method="POST" enctype="multipart/form-data">
  Body:<textarea name="Body"></textarea><br>
  <input type="submit" name="send" value="Send">
</form>

<table>
<thead>
<tr>
<th>Name</th>
<th>note_update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>note_update(NUMBER fid, NUMBER mailId)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Modifies &lt;mailId&gt; note from &lt;fid&gt; folder. Before calling this method you should upload the new note to the server.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>note_remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>note_remove(NUMBER fid, NUMBER mailId)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Removes &lt;mailId&gt; note from &lt;fid&gt; folder.</td>
</tr>
</tbody>
</table>

**Freebusy**

**FBENTRY** - represents a busy interval [startDate, endDate)

- WEBDATE **startDate**
- WEBDATE **endDate**

<table>
<thead>
<tr>
<th>Name</th>
<th>freebusy_query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY(FBENTRY) freebusy_query(STRING userEmail, WEBDATE startDate, WEBDATE endDate)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Retrieve busy intervals in the specified interval [startDate, endDate] for user specified by &lt;userEmail&gt;</td>
</tr>
</tbody>
</table>

**Date functions**

1. **name** date_load

<table>
<thead>
<tr>
<th>Name</th>
<th>date_load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>WEBDATE date_load()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns the user's current date in WEBDATE format.</td>
</tr>
</tbody>
</table>

2. **name** timezone_loadList

<table>
<thead>
<tr>
<th>Name</th>
<th>timezone_loadList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY timezone_loadList()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns a list of strings. Each string represents a timezone.</td>
</tr>
</tbody>
</table>
2.4.5 WebAdmin Specific Methods

The following methods are only used for WebAdmin related operations:

1. **headers_getUserAgent**
   - **Syntax**: STRING headers_getUserAgent()
   - **Explanation**: Returns User-Agent HTTP header value.

2. **getHostName**
   - **Syntax**: STRING getHostName()
   - **Explanation**: Returns server's host name.

3. **server_getVersion**
   - **Syntax**: STRING server_getVersion()
   - **Explanation**: Returns the server's version.

4. **server_getLK**
   - **Syntax**: MAP server_getLK()
   - **Explanation**: Returns a map object that contains informations about the license key.

5. **config_save**
   - **Syntax**: config_save()
   - **Explanation**: Makes all the changes permanent.

6. **config_discard**
   - **Syntax**: config_discard()
   - **Explanation**: Discards all the uncommitted changes (cached on this session).

7. **licenseKey_load**
   - **Syntax**: MAP licenseKey_load()
   - **Explanation**: Loads informations about an uploaded license key and returns them in a map object. Before calling this method you should upload the license key to the server. Webadmin server recognizes a license key upload try by a query "action=licenseKey".

Example:

```perl
<%h = getSessionId()%>
```
<form method="POST" enctype="multipart/form-data" action="test.hsp?_h=<%h%>&action=uploadKey">
    License Key File: <input type="file" name="file"><br>
    <input type=submit name = "Apply License Key"><br>
</form>

8.

<table>
<thead>
<tr>
<th>Name</th>
<th>licenseKey_apply</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>licenseKey_apply()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Applies an uploaded license key.</td>
</tr>
</tbody>
</table>

9.

<table>
<thead>
<tr>
<th>Name</th>
<th>licenseKey_getAddons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY licenseKey_getAddons()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns the list of license addons.</td>
</tr>
</tbody>
</table>

10.

<table>
<thead>
<tr>
<th>Name</th>
<th>geoip_apply</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>geoip_apply()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Applies an uploaded GeoIP CSV database (in MAXMIND format). action=upload should be used to upload this file.</td>
</tr>
</tbody>
</table>

11.

<table>
<thead>
<tr>
<th>Name</th>
<th>geoip_loadCountries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY geoip_loadCountries()</td>
</tr>
</tbody>
</table>
| Explanation| Returns a list of maps, parsed from the GeoIP database:
  - country: country name
  - code: country code (2 letters) |

12.

<table>
<thead>
<tr>
<th>Name</th>
<th>object_getInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP object_getInfo(STRING objectName)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns detailed info about &lt;objectName&gt; config object.</td>
</tr>
</tbody>
</table>

**Connection specific methods**

1.

<table>
<thead>
<tr>
<th>Name</th>
<th>connection_isSecure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING connection_isSecure()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns true if the current connection is secure (SSL) or false otherwise.</td>
</tr>
</tbody>
</table>

2.

<table>
<thead>
<tr>
<th>Name</th>
<th>connection_getUrlRedirect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING connection_getUrlRedirect()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns a url redirection string for the current connection</td>
</tr>
</tbody>
</table>
Service specific methods

1. Name

   service_load

   **Syntax**

   MAP service_load(STRING service [, STRING attributesType])

   **Explanation**

   Loads attributes of `<service>` from config file. It returns a map object that contains all attributes for `<service>`. `<service>` can be: "server", "dnr", "reporting", "processing", "smtpIncoming", "smtpOutgoing", "pop3", "imap", "webmail", "webadmin", "log", "cli", "ftp", "rpop", "pop3Proxy", "imapProxy", "userDb". By default `<attributesType>` is considered "values". For some services `<attributesType>` can be:
   - smtpIncoming
     - mappingData
     - accessControl
   - pop3
     - accessControl
   - imap
     - accessControl
   - webMail
     - accessControl
   - webAdmin
     - accessControl
   - cli
     - accessControl
   - ftp
     - accessControl
   - pop3Proxy
     - mappingData
     - connectionData
     - accessControl
   - imapProxy
     - mappingData
     - connectionData
     - accessControl

2. Name

   service_update

   **Syntax**

   MAP service_update(MAP values, STRING service [, STRING attributesType])

   **Explanation**

   Updates the service's `<attributesType>` specified by `<service>`. The updated properties are taken from `<values>` map object. If the map object returned by this method is not empty (update failure) then it contains all wrong properties from `<values>`.
HSP – AXIGEN Server Side Scripting Language

3. | Name | service_commit |
   | Syntax | service_commit(STRING service) |
   | Explanation | Commits changes made on <service>. |

4. | Name | service_commitFilters |
   | Syntax | service_commitFilters(STRING service) |
   | Explanation | Commits server’s filters changes. |

5. | Name | service_reset |
   | Syntax | service_reset(STRING service) |
   | Explanation | Resets changes made on <service>. |

6. | Name | service_listLoad |
   | Syntax | ARRAY service_listLoad(STRING objectName [, STRING service]) |
   | Explanation | Loads the list of <objectName> objects, contained in <service>. <objectName> and <service> can be:
   - scriptFilters
     - server
   - socketFilters
     - server
   - activeFilters
     - server
   - logRules
     - log
   - nameServers
     - dnr
   - listeners
     - smtpIncoming
     - pop3
     - imap
     - webmail
     - webadmin
     - cli
     - ftp
     - log
     - pop3Proxy
     - imapProxy
     - allowRules
       - smtpIncoming
       - pop3
       - imap
HSP – AXIGEN Server Side Scripting Language

- webmail
- webadmin
- cli
- ftp
- pop3Proxy
- imapProxy
  - denyRules
    - smtpIncoming
    - pop3
    - imap
    - webmail
    - webadmin
    - cli
    - ftp
    - pop3Proxy
    - imapProxy
  - urlRedirects
    - webmail
    - webadmin
  - hostnameResolvers
    - webmail
  - templates
    - reporting
  - databases
    - reporting
  - graphics
    - reporting
  - userMaps
    - server
  - ldapConnectors
    - userDb

It returns an ARRAY of MAP values.

<table>
<thead>
<tr>
<th>Name</th>
<th>service_listAdd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP service_listAdd(MAP objectValues, STRING objectName [, STRING service])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Adds an object specified by objectValues MAP to the list of &lt;objectName&gt; objects, contained in &lt;service&gt;.&lt;objectName&gt; can be:</td>
</tr>
<tr>
<td></td>
<td>nameservers</td>
</tr>
<tr>
<td></td>
<td>scriptFilters</td>
</tr>
<tr>
<td></td>
<td>socketFilters</td>
</tr>
<tr>
<td></td>
<td>activeFilters</td>
</tr>
<tr>
<td></td>
<td>logRules</td>
</tr>
<tr>
<td></td>
<td>urlRedirects</td>
</tr>
</tbody>
</table>
If the map object returned by this method is not empty (update failure) then it contains all wrong properties from <objectValues>.

8.

<table>
<thead>
<tr>
<th>Name</th>
<th>service_listRemove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>service_listRemove(STRING objectKey, STRING objectName [, STRING service])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Removes the object with the key &lt;objectKey&gt; from the list of objects specified by &lt;objectName&gt;, contained in &lt;service&gt;. &lt;service&gt; and &lt;objectName&gt; can have the same values as for #service_listAdd method.</td>
</tr>
</tbody>
</table>

9.

<table>
<thead>
<tr>
<th>Name</th>
<th>service_listLoadElement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP service_listLoadElement(STRING objectKey, STRING objectName [, STRING service])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads the element with the key &lt;objectKey&gt; from the list of objects specified by &lt;objectName&gt;, contained in &lt;service&gt;. &lt;service&gt; and &lt;objectName&gt; can have the same values as for #service_listAdd method.</td>
</tr>
</tbody>
</table>

10.

<table>
<thead>
<tr>
<th>Name</th>
<th>service_listUpdateElement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP service_listUpdateElement(MAP objectValues, STRING objectKey, STRING objectName [, STRING service])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Updates the object with the key &lt;objectKey&gt; from the list specified by &lt;objectName&gt;, contained in &lt;service&gt;. The updated properties are taken from objectValues map object. &lt;service&gt; and &lt;objectName&gt; can have the same values as for #service_listAdd method. If the map object returned by this method is not empty (update failure) then it contains all wrong properties from &lt;objectValues&gt;.</td>
</tr>
</tbody>
</table>

Search specific methods

<table>
<thead>
<tr>
<th>Name</th>
<th>search_applyFilter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>search_applyFilter(STRING searchListName, MAP searchInfo)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Applies a search filter on the &lt;searchListName&gt;. The search filter information is contained in a MAP object, &lt;searchInfo&gt;.</td>
</tr>
</tbody>
</table>
<searchListName> can be one of the domain objects ("domains", "users", "mlists", "groups", "folderrcpts", "mlistusers"). The MAP containing the search info must have the following properties and values:
- STRING <type> - can be:
  - "contains"
  - "begins"
  - "ends"
  - "is"
- [STRING <negation>] - optional parameter, if not specified the default value is "true"; can be:
  - "true"
  - "false"
- STRING <pattern> - the pattern to search for.

For applying such a filter, the object list must be cached first, using the corresponding caching method (e.g. #domain_listCache, #user_listCache, etc). After applying the search filter, the list should be loaded using the corresponding loading method.

Log specific methods

1. **Name**: log_loadFileList
   **Syntax**: ARRAY log_loadFileList()
   **Explanation**: Loads the log file list; it returns an array of map objects. The map objects have the following properties:
   - "fileName"
   - "size"

2. **Name**: log_showFile
   **Syntax**: log_showFile(STRING fileName)
   **Explanation**: Shows the content of the log file <fileName>.

3. **Name**: log_removeFile
   **Syntax**: log_removeFile(STRING fileName)
   **Explanation**: Removes the log file <fileName>.

Queue specific methods

1. **Name**: queue_force
   **Syntax**: queue_force()
   **Explanation**: Force queue to try to send all mails.
2. Name | queue_loadMails  
|---|---  
Syntax | ARRAY queue_loadMails([NUMBER startIndex, NUMBER stopIndex])  
Explanation | Before calling this method you must first call queue_applySearch() method, or else the list returned will be empty. This method loads all the mails between <startIndex> and <stopIndex>. If <startIndex> and <stopIndex> are missing then all mails in the queue are loaded. It returns an ARRAY of MAP values. Each MAP object has the following properties:  
  - NUMBER id  
  - STRING hid  
  - STRING sender  
  - NUMBER size  
  - STRING receiveDate  
  - NUMBER age  
  - STRING nextSchedule  
  - NUMBER retryCount  
  - STRING status  
  - NUMBER rcptCount - total count of recipients  
  - ARRAY rcptList (limited to 5) - each element has MAP type and contains the following properties:  
    - STRING failInfo  
    - STRING mbox  
    - STRING name  
    - STRING status  
3. Name | queue_loadMail  
|---|---  
Syntax | MAP queue_loadMail(NUMBER mailID)  
Explanation | Loads the mail specified by <mailID> from the queue. It returns a MAP object with the following properties:  
  - NUMBER id  
  - STRING hid  
  - STRING sender  
  - NUMBER size  
  - STRING receiveDate  
  - NUMBER age  
  - STRING nextSchedule  
  - NUMBER retryCount  
  - STRING status  
  - NUMBER rcptCount  
  - ARRAY rcptList - each element has MAP type and contains the following properties:  
    - STRING failInfo  
    - STRING mbox
### HSP – AXIGEN Server Side Scripting Language

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>queue_rescheduleMail</td>
<td>queue_rescheduleMail(NUMBER mailId)</td>
<td>Reschedules the mail &lt;mailId&gt; from queue.</td>
</tr>
<tr>
<td>queue_removeMail</td>
<td>queue_removeMail(NUMBER mailId)</td>
<td>Deletes the mail &lt;mailId&gt; from queue.</td>
</tr>
<tr>
<td>queue_sendNDR</td>
<td>queue_sendNDR(NUMBER mailId)</td>
<td>Sends a non delivery report to the originator and remove the mail &lt;mailId&gt; from the queue. Applies for all mail recipients.</td>
</tr>
<tr>
<td>queue_getSize</td>
<td>NUMBER queue_getSize()</td>
<td>Returns the number of mails in the queue.</td>
</tr>
<tr>
<td>queue_applySearch</td>
<td>queue_applySearch(ARRAY filters)</td>
<td>Caches the list of mails in the queue and applies search &lt;filters&gt;. &lt;filters&gt; is an ARRAY with MAP elements. Each element contains the following search criteria: - name (&quot;size&quot;, &quot;age&quot;, &quot;retries&quot;, &quot;status&quot;, &quot;sender&quot;, &quot;recipient&quot; or &quot;nextSchedule&quot;) - operator (&quot;gt&quot;, &quot;lt&quot;, &quot;eq&quot;, &quot;contains&quot;, &quot;notContains&quot;) - value: can be numeric (for &quot;size&quot;, &quot;age&quot; and &quot;retries&quot;), string (for &quot;status&quot;, &quot;sender&quot; and &quot;recipient&quot;) or date (for &quot;nextSchedule&quot;); numeric and date values can be used only with &quot;gt&quot;, &quot;lt&quot; and &quot;eq&quot; operators; string values can be used only with &quot;contains&quot; and &quot;notContains&quot; operators.</td>
</tr>
<tr>
<td>queue_getSearchInfo</td>
<td>ARRAY queue_getSearchInfo()</td>
<td>Returns previously applied search &lt;filters&gt;.</td>
</tr>
<tr>
<td>queue_applySort</td>
<td>queue_applySort(STRING sortType, STRING sortOrder)</td>
<td></td>
</tr>
</tbody>
</table>
### Explanation
Sorts the list of mails from the queue (cached before) according to `<sortType>`, in `<sortOrder>`. `<sortType>` can be:
- `size`
- `age`
- `retries`
- `status`
- `sender`
- `nextSchedule`

`<sortOrder>` can be:
- `ascending`
- `descending`

### 11. Name: queue_getSortInfo

**Syntax**: `queue_getSortInfo(STRING sortType, STRING sortOrder)`

**Explanation**: Returns `<sortType>` and `<sortOrder>` of queue mails currently cached.

### 12. Name: queue_refresh

**Syntax**: `queue_refresh()`

**Explanation**: Refreshes the list of mails from the queue, applying the existing search filters.

### Listener specific methods

1. Name: listener_create

**Syntax**: `listener_create(STRING service)`

**Explanation**: Creates a new listener for `<service>`. This listener is the new current listener (cached for current session). It can be modified and committed. `<service>` can be: "smtpIncomming", "pop3", "imap", "webmail", "webadmin", "log", "cli", "ftpBackup".

   Example:
   ```
   <%listener_create("webmail")%>
   ```

2. Name: listener_load

**Syntax**: `listener_load(STRING key, STRING service)`

**Explanation**: Loads the listener identified by address `<key>`, from `<service>`. This is the new current listener.

   Example:
   ```
   <%listener_load("127.0.0.1:8000", "webmail")%>
   ```
### 3. `listener_update`

<table>
<thead>
<tr>
<th>Name</th>
<th><code>listener_update</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>listener_update()</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Updates changes of current listener.</td>
</tr>
</tbody>
</table>

### 4. `listener_add`

<table>
<thead>
<tr>
<th>Name</th>
<th><code>listener_add</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>listener_add()</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Adds the current listener to the service which was created for; in case of no failure, the listener is discarded from cache.</td>
</tr>
</tbody>
</table>

### 5. `listener_commit`

<table>
<thead>
<tr>
<th>Name</th>
<th><code>listener_commit</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>listener_commit()</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Commits the changes of the current listener. If the listener was loaded, the command updates it. If the listener is newly created, then it adds it to the service which was created for.</td>
</tr>
</tbody>
</table>

### 6. `listener_get`

<table>
<thead>
<tr>
<th>Name</th>
<th><code>MAP listener_get(STRING valuesName)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>MAP listener_get(STRING valuesName)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads the attributes <code>&lt;valuesName&gt;</code> from the current listener. <code>&lt;valuesName&gt;</code> can be &quot;values&quot; or &quot;sslControl&quot;.</td>
</tr>
</tbody>
</table>

Example:
```
<%listener = listener_get("values")%>
<%sslData = listener_get("sslData")%>
```

### 7. `listener_set`

<table>
<thead>
<tr>
<th>Name</th>
<th><code>MAP listener_set(MAP values, STRING valuesName)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>MAP listener_set(MAP values, STRING valuesName)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Sets <code>&lt;values&gt;</code> to <code>&lt;valuesName&gt;</code> object of current listener. If the map object returned by this method is not empty (set failure) then it contains all wrong properties from <code>&lt;values&gt;</code>.</td>
</tr>
</tbody>
</table>

Example:
```
<%listener_set(sslData, "sslData")%>
```

### 8. `listener_listLoad`

<table>
<thead>
<tr>
<th>Name</th>
<th><code>ARRAY listener_listLoad(STRING objectName)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td><code>ARRAY listener_listLoad(STRING objectName)</code></td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads elements of the list <code>&lt;objectName&gt;</code>, from the current listener. <code>&lt;objectName&gt;</code> can be &quot;allowRules&quot; or &quot;denyRules&quot;.</td>
</tr>
</tbody>
</table>

Example:
```
<%allowRules = listener_listLoad("allowRules")%>
```
### 9. `listener_listAdd`

**Name**  
listener_listAdd

**Syntax**  
MAP listener_listAdd(MAP objectValues, STRING objectName)

**Explanation**  
Adds `<objectValues>` to `<objectName>` list from current listener. If the map object returned by this method is not empty (add failure) then it contains all wrong properties from `<objectValues>`.

### 10. `listener_listRemove`

**Name**  
listener_listRemove

**Syntax**  
listener_listRemove(STRING objectKey, STRING objectName)

**Explanation**  
Removes the element `<objectKey>` from the `<objectName>` list, from current listener.

### 11. `listener_listLoadElement`

**Name**  
listener_listLoadElement

**Syntax**  
MAP listener_listLoadElement(STRING objectKey, STRING objectName)

**Explanation**  
Loads the element identified by `<objectKey>` from `<objectName>` list, from the current listener and returns the map object.

### 12. `listener_listUpdateElement`

**Name**  
listener_listUpdateElement

**Syntax**  
MAP listener_listUpdateElement(MAP objectValues, STRING objectKey, STRING objectName)

**Explanation**  
Updates the object identified by `<objectKey>` (from `<objectName>` list) with the attributes from `<objectValues>`. If the map object returned by this method is not empty (update failure) then it contains all wrong properties from `<objectValues>`.

### 13. `listener_getService`

**Name**  
listener_getService

**Syntax**  
STRING listener_getService()

**Explanation**  
Returns the service for the current listener.

### 14. `listener_isNew`

**Name**  
listener_isNew

**Syntax**  
listener_isNew()

**Explanation**  
It returns "true" if the listener is new, or "false" otherwise.

---

**Domain specific methods**

### 1. `domain_create`

**Name**  
domain_create

**Syntax**  
domain_create(STRING domainName, STRING password,
<table>
<thead>
<tr>
<th><strong>String</strong></th>
<th><strong>Syntax</strong></th>
<th><strong>Explanation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>STRING dSLocation, STRING maclSupport, STRING dOSLocation, ARRAY mSLocations</td>
<td></td>
<td>Creates a new domain with name <code>&lt;domainName&gt;</code>, with the postmaster password <code>&lt;password&gt;</code>, into domain location specified by <code>&lt;dSLocation&gt;</code> (domainStorageLocation) and optionally <code>&lt;maclSupport&gt;</code> (&quot;yes&quot; or &quot;no&quot;), <code>&lt;dOSLocation&gt;</code> (domainObjectsStorageLocation) and <code>&lt;mSLocations&gt;</code> (messageStorageLocations). This domain is the new current domain.</td>
</tr>
</tbody>
</table>

2. **Name**
   - `domain_load`

   **Syntax**
   - `domain_load(STRING domainName)`

   **Explanation**
   - Loads the domain `<domainName>`. This is the new current domain.

3. **Name**
   - `domain_getMacl`

   **Syntax**
   - `STRING domain_getMacl()`

   **Explanation**
   - Returns "enable" if MACL is enabled for the current domain or "disable" otherwise.

4. **Name**
   - `domain_setMacl`

   **Syntax**
   - `domain_setMacl(STRING enable)`

   **Explanation**
   - If `<enable>` is "enable" then MACL is enabled for the current domain or disabled if `<enable>` is "disable".

5. **Name**
   - `MAP domain_get`

   **Syntax**
   - `MAP domain_get(STRING valuesName)`

   **Explanation**
   - Loads `<valueName>` object from the current domain. `<valueName>` can be:
     - "values"
     - "defaultQuota"
     - "defaultLimits"
     - "adminLimits"
     - "webmailData"
     - "migrationData"
     - "registryInfo"
     - "dsParams"
     - "storageInfo": returns a MAP:
       - MAP domainStorage
         - STRING location
         - NUMBER maxFileSize
         - NUMBER maxFileCount
         - NUMBER allTotalSize
         - NUMBER allUsedSize
### 6. domain_set

**Syntax**

```map
domain_set(MAP values, STRING valueName)
```

**Explanation**

Sets `<values>` map object to `<valueName>` object of current domain. If the map object returned by this method is not empty (set failure) then it contains all wrong properties from `<values>`. `<valueName>` can be "values", "defaultQuota", "defaultLimits", "adminLimits", "webmailData" or "migrationData".

### 7. domain_commit

**Syntax**

```function
domain_commit()
```

**Explanation**

Commits changes of current domain.
8. **domain_commitFilters**
   - **Syntax**: `domain_commitFilters()`
   - **Explanation**: Commits filters changes for the current domain.

9. **domain_register**
   - **Syntax**: `domain_register(STRING dSLocation[, STRING dOSLocation, STRING mSLocation])`
   - **Explanation**: Registers the domain found in the location specified by `<dSLocation>` and optionally by `<dOSLocation>` and `<mSLocation>`.

10. **domain_unregister**
    - **Syntax**: `domain_unregister(STRING domainName)`
    - **Explanation**: Unregisters the domain specified by `<domainName>`.

11. **domain_listLoad**
    - **Syntax**: `ARRAY domain_listLoad(STRING objectName [, NUMBER startIndex, NUMBER stopIndex])`
    - **Explanation**: Loads elements of the `<objectName>` list from the current domain. The list contains elements between `<startIndex>` and `<stopIndex>`. If `<startIndex>` and `<stopIndex>` are missing then all elements are loaded. `<objectName>` can be:
      - "domains": returns an ARRAY of MAPs:
        - STRING name
        - STRING totalSize
        - STRING usedSize
        - STRING expandableSize
      - "users"
      - "mailLists"
      - "groups"
      - "folderRCPTs"
      - "userClasses"
      - "aliases"
      - "scriptFilters"
      - "socketFilters"
      - "activeFilters"
      - "defaultScriptFilters"
      - "defaultSocketFilters"
      - "defaultActiveFilters"
### 12. domain_listAdd

**Syntax**

```
MAP domain_listAdd(MAP objectValues, STRING objectName)
```

**Explanation**

Adds `<objectValues>` into `<objectName>` list of the current domain. If the map object returned by this method is not empty (add failure) then it contains all wrong properties from `<objectValues>`. `<objectName>` can be: "aliases", "scriptFilters", "socketFilters", "activeFilters", "defaultScriptFilters", "defaultSocketFilters" or "defaultActiveFilters".

### 13. domain_listRemove

**Syntax**

```
domain_listRemove(STRING objectKey, STRING objectName)
```

**Explanation**

Removes element `<objectKey>` from `<objectName>` list, from current domain. `<objectName>` can be: "users", "mailLists", "groups", "folderRcpts", "userClasses", "aliases", "scriptFilters", "socketFilters", "activeFilters", "defaultScriptFilters", "defaultSocketFilters" or "defaultActiveFilters".

### 14. domain_listLoadElement

**Syntax**

```
MAP domain_listLoadElement(STRING objectKey, STRING objectName)
```

**Explanation**

Loads the element identified by `<objectKey>` from `<objectName>` list, from current domain. `<objectName>` can be: "scriptFilters", "socketFilters", "activeFilters", "defaultScriptFilters", "defaultSocketFilters" or "defaultActiveFilters".

### 15. domain_listUpdateElement

**Syntax**

```
MAP domain_listUpdateElement (MAP objectValues, STRING objectKey, STRING objectName)
```

**Explanation**

Updates the object identified by `<objectKey>`, from `<objectName>` list, with attributes contained in `<objectValues>`. If the map object returned by this method is not empty (update failure) then it contains all wrong properties from `<objectValues>`. `<objectName>` can be: "scriptFilters", "socketFilters", "activeFilters", "defaultScriptFilters", "defaultSocketFilters" or "defaultActiveFilters".

### 16. domain_listGetSize

**Syntax**

```
NUMBER domain_listGetSize(STRING objectName)
```

**Explanation**

Returns number of elements from `<objectName>` list. `<objectName>` can be "domains", "users", "mailLists", "groups", "folderRcpts" or "userClasses".

---

Copyright © 2008 Gecad Technologies S.A.

[http://www.axigen.com](http://www.axigen.com)
Name | domain_listCache  
---|---  
Syntax | domain_listCache(STRING objectName[, ARRAY permissions])  
Explanation | It caches the list of <objectName> items. <objectName> can be "domains", "users", "mailLists", "groups", "folderRcpts" or "userClasses". In case of "domains" the list is filtered considering the list of <permissions>. If size of <permissions> list is greater than 0 no domain storage info will be available.

**User specific methods**

1. Name | user_create  
---|---  
Syntax | user_create(STRING userName, STRING domainName)  
Explanation | Creates a new account, <userName>, for <domainName>. This account is the new current account.

2. Name | user_load  
---|---  
Syntax | user_load(STRING userName, STRING domainName)  
Explanation | Loads account <userName> from <domainName>. This is the new current account.

3. Name | user_get  
---|---  
Syntax | MAP user_get(STRING valuesName)  
Explanation | Loads <valuesName> object from the current account. <valuesName> can be "values", "webmailData", "contactInfo", "filterData", "limits", "inheritedLimits", "quota", "inheritedQuota" or "registryInfo".

4. Name | user_set  
---|---  
Syntax | MAP user_set(MAP values, STRING valuesName)  
Explanation | Sets <values> map object to <valuesName> object of current account. If the map object returned by this method is not empty (set failure) then it contains all wrong properties from <attributes>. <valuesName> can be "values", "webmailData", "contactInfo", "filterData", "limits" or "quota".

5. Name | user_commit  
---|---  
Syntax | user_commit()  
Explanation | Commits changes of current account.

6. Name | user_listLoad  
---|---  
Syntax | ARRAY user_listLoad(STRING objectName)
Explanation | Loads elements of the <objectName> list from the current account. <objectName> can be: "aliases", "scriptFilters", "socketFilters", "activeFilters".

7. Name | user_listAdd
Syntax | MAP user_listAdd(MAP objectValues, STRING objectName)
Explanation | Adds <objectValues> into <objectName> list of the current account. If the map object returned by this method is not empty (add failure) then it contains all wrong properties from <objectValues>. <objectName> can be: "aliases", "scriptFilters", "socketFilters", "activeFilters".

8. Name | user_listRemove
Syntax | user_listRemove(STRING objectKey, STRING objectName)
Explanation | Removes element <objectKey> from <objectName> list, from current account. <objectName> can be: "aliases", "scriptFilters", "socketFilters", "activeFilters".

9. Name | user_listLoadElement
Syntax | MAP user_listLoadElement(STRING objectKey, STRING objectName)
Explanation | Loads the element identified by <objectKey> from <objectName> list, from current account. <objectName> can be: "scriptFilters", "socketFilters", "activeFilters".

10. Name | user_listUpdateElement
Syntax | MAP user_listUpdateElement(MAP objectValues, STRING objectKey, STRING objectName)
Explanation | Updates the object identified by <objectKey>, from <objectName> list, with attributes <objectValues>. If the map object returned by this method is not empty (update failure) then it contains all wrong properties from <objectValues>. <objectName> can be: "scriptFilters", "socketFilters", "activeFilters".

Maillist specific methods

1. Name | mlist_create
Syntax | mlist_create(STRING mlistName, STRING domainName)
Explanation | Creates a new mail list,<mlistName> for <domainName>. This mail list is the new current mail list. Setting the parameter <adminEmail> from "values" must be done until committing the mail list.
2. **mlist_isNew**

<table>
<thead>
<tr>
<th>Name</th>
<th>mlist_isNew</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>mlist_isNew()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns <em>true</em> if the current list is newly created and <em>false</em> otherwise.</td>
</tr>
</tbody>
</table>

Example for using this method:

```html
<%IF mlist_isNew()%>
    The current list is new.<br>
<%ELSE%>
    The current list is old.<br>
<%ENDIF%>
```

3. **mlist_load**

<table>
<thead>
<tr>
<th>Name</th>
<th>mlist_load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>mlist_load(STRING mlistName, STRING domainName)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads mail list <code>&lt;mlistName&gt;</code> from <code>&lt;domainName&gt;</code>. This is the new current mail list.</td>
</tr>
</tbody>
</table>

4. **mlist_get**

<table>
<thead>
<tr>
<th>Name</th>
<th>mlist_get</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP mlist_get(STRING valuesName)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads <code>&lt;valuesName&gt;</code> object from the current mail list. <code>&lt;valuesName&gt;</code> can be &quot;values&quot;, &quot;webmailData&quot;, &quot;limits&quot;, &quot;mboxContainerQuota&quot; or &quot;registryInfo&quot;.</td>
</tr>
</tbody>
</table>

5. **mlist_set**

<table>
<thead>
<tr>
<th>Name</th>
<th>mlist_set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP mlist_set(MAP values, STRING valuesName)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Sets <code>&lt;values&gt;</code> map object to <code>&lt;valuesName&gt;</code> object of current mail list. If the map object returned by this method is not empty (set failure) then it contains all wrong properties from <code>&lt;values&gt;</code>. <code>&lt;valuesName&gt;</code> can be &quot;values&quot;, &quot;webmailData&quot;, &quot;limits&quot; or &quot;mboxContainerQuota&quot;.</td>
</tr>
</tbody>
</table>

6. **mlist_commit**

<table>
<thead>
<tr>
<th>Name</th>
<th>mlist_commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>mlist_commit()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Commits changes of current mail list. For a successful commit, &quot;adminEmil&quot; from mlist's <code>&lt;values&gt;</code> is required.</td>
</tr>
</tbody>
</table>

7. **mlist_listLoad**

<table>
<thead>
<tr>
<th>Name</th>
<th>mlist_listLoad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY mlist_listLoad(STRING objectName [, NUMBER startIndex, NUMBER stopIndex])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads elements of the <code>&lt;objectName&gt;</code> list from the current mail list. <code>&lt;objectName&gt;</code> can be &quot;removeHeaders&quot;, &quot;scriptFilters&quot;, &quot;socketFilters&quot;, &quot;activeFilters&quot; or &quot;mailListUsers&quot;.</td>
</tr>
<tr>
<td>Name</td>
<td>Syntax</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>mlist_listAdd</td>
<td>MAP mlist_listAdd(MAP objectValues, STRING objectName)</td>
</tr>
<tr>
<td>mlist_listRemove</td>
<td>mlist_listRemove(STRING objectKey, STRING objectName)</td>
</tr>
<tr>
<td>mlist_listLoadElement</td>
<td>MAP mlist_listLoadElement(STRING objectKey, STRING objectName)</td>
</tr>
<tr>
<td>mlist_listUpdateElement</td>
<td>MAP mlist_listUpdateElement(MAP objectValues, STRING objectKey, STRING objectName)</td>
</tr>
<tr>
<td>mlist_listGetSize</td>
<td>NUMBER mlist_listGetSize(STRING objectName)</td>
</tr>
<tr>
<td>mlist_listCache</td>
<td>mlist_listCache(STRING objectName)</td>
</tr>
</tbody>
</table>

Note: `<startIndex>` and `<stopIndex>` can be used only for "mailListUsers". These arguments are ignored for others `<objectName>` lists.

8. Name: mlist_listAdd
   Syntax: MAP mlist_listAdd(MAP objectValues, STRING objectName)
   Explanation: Adds `<objectValues>` into `<objectName>` list of the current mail list. If the map object returned by this method is not empty (add failure) then it contains all wrong properties from `<objectValues>`. `<objectName>` can be "removeHeaders", "scriptFilters", "socketFilters", "activeFilters" or "mailListUsers".

9. Name: mlist_listRemove
   Syntax: mlist_listRemove(STRING objectKey, STRING objectName)
   Explanation: Removes element `<objectKey>` from `<objectName>` list, from current mail list. `<objectName>` can be "removeHeaders", "scriptFilters", "socketFilters", "activeFilters" or "mailListUsers".

10. Name: mlist_listLoadElement
    Syntax: MAP mlist_listLoadElement(STRING objectKey, STRING objectName)
    Explanation: Loads the element identified by `<objectKey>` from `<objectName>` list, from current mail list. `<objectName>` can be "scriptFilters", "socketFilters", "activeFilters" or "mailListUsers".

11. Name: mlist_listUpdateElement
    Syntax: MAP mlist_listUpdateElement(MAP objectValues, STRING objectKey, STRING objectName)
    Explanation: Updates the object identified by `<objectKey>`, from `<objectName>` list, with attributes `<objectValues>`. If the map object returned by this method is not empty (update failure) then it contains all wrong properties from `<objectValues>`. `<objectName>` can be "scriptFilters", "socketFilters", "activeFilters" or "mailListUsers".

12. Name: mlist_listGetSize
    Syntax: NUMBER mlist_listGetSize(STRING objectName)
    Explanation: Returns the number of elements contained by the list specified by `<objectName>`. `<objectName>` can only be "mailListUsers".

13. Name: mlist_listCache
    Syntax: mlist_listCache(STRING objectName)
### Explanation
Caches the list of `<objectName>` elements. `<objectName>` can only be "mailListUsers".

### Group specific methods

1. **Name**
   - group_create

   **Syntax**
   - group_create(STRING groupName, STRING domainName)

   **Explanation**
   Creates a new group for `<domainName>`. This group is the new current group.

2. **Name**
   - group_load

   **Syntax**
   - group_load(STRING groupName, STRING domainName)

   **Explanation**
   Loads group `<groupName>` from `<domainName>`. This is the new current group.

3. **Name**
   - group_get

   **Syntax**
   - MAP group_get(STRING valuesName)

   **Explanation**
   Loads `<valuesName>` object from the current group. `<valuesName>` can be "values" or "registryInfo".

4. **Name**
   - group_set

   **Syntax**
   - MAP group_set(MAP values, STRING valuesName)

   **Explanation**
   Sets `<values>` map object to `<valuesName>` object of current group. If the map object returned by this method is not empty (set failure) then it contains all wrong properties from `<values>`. `<valuesName>` can be only "values".

5. **Name**
   - group_commit

   **Syntax**
   - group_commit()

   **Explanation**
   Commits changes of current group.

6. **Name**
   - group_listLoad

   **Syntax**
   - ARRAY group_listLoad(STRING objectName)

   **Explanation**
   Loads elements of the `<objectName>` list from the current group. `<objectName>` can be "addresses", "scriptFilters", "socketFilters" or "activeFilters".

7. **Name**
   - group_listAdd

   **Syntax**
   - MAP group_listAdd(MAP objectValues, STRING objectName)

   **Explanation**
   Adds `<objectValues>` into `<objectName>` list of the current group. If the map object returned by this method is not empty
### HSP – AXIGEN Server Side Scripting Language

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>group_listRemove</td>
<td>group_listRemove(STRING objectKey, STRING objectName)</td>
<td>Removes element &lt;objectKey&gt; from &lt;objectName&gt; list, from current group. &lt;objectName&gt; can be &quot;addresses&quot;, &quot;scriptFilters&quot;, &quot;socketFilters&quot; or &quot;activeFilters&quot;.</td>
</tr>
</tbody>
</table>

8.

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>group_listLoadElement</td>
<td>MAP group_listLoadElement(STRING objectKey, STRING objectName)</td>
<td>Loads the element identified by &lt;objectKey&gt; from &lt;objectName&gt; list, from current group. &lt;objectName&gt; can be &quot;scriptFilters&quot;, &quot;socketFilters&quot; or &quot;activeFilters&quot;.</td>
</tr>
</tbody>
</table>

9.

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>group_listUpdateElement</td>
<td>MAP group_listUpdateElement(MAP objectValues, STRING objectKey, STRING objectName)</td>
<td>Updates the object identified by &lt;objectKey&gt;, from &lt;objectName&gt; list, with attributes &lt;objectValues&gt;. If the map object returned by this method is not empty (update failure) then it contains all wrong properties from &lt;objectValues&gt;. &lt;objectName&gt; can be &quot;scriptFilters&quot;, &quot;socketFilters&quot; or &quot;activeFilters&quot;.</td>
</tr>
</tbody>
</table>

10.

FolderRCPT specific methods

1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>folderRcpt_create</td>
<td>folderRcpt_create(STRING fRcptName, STRING domainName)</td>
<td>Creates a new folderRCPT for &lt;domainName&gt;. This folderRCPT is the new current folderRCPT.</td>
</tr>
</tbody>
</table>

2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>folderRcpt_load</td>
<td>folderRcpt_load(STRING fRcptName, STRING domainName)</td>
<td>Loads folderRCPT &lt;fRcptName&gt; from &lt;domainName&gt;. This is the new current fRcpt.</td>
</tr>
</tbody>
</table>

3.

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>folderRcpt_get</td>
<td>MAP folderRcpt_get(STRING valuesName)</td>
<td>Loads &lt;valuesName&gt; object from the current folderRCPT. &lt;valuesName&gt; can be &quot;values&quot; or &quot;registryInfo&quot;.</td>
</tr>
</tbody>
</table>
4. **Name** | folderRcpt_set  
**Syntax** | MAP folderRcpt_set(MAP values, STRING valuesName)  
**Explanation** | Sets `<values>` map object to `<valuesName>` object of current folderRCPT. If the map object returned by this method is not empty (set failure) then it contains all wrong properties from `<values>`. `<valuesName>` can be only "values".

5. **Name** | folderRcpt_commit  
**Syntax** | folderRcpt_commit()  
**Explanation** | Commits changes of current folderRCPT.

6. **Name** | folderRcpt_listLoad  
**Syntax** | ARRAY folderRcpt_listLoad(STRING objectName)  
**Explanation** | Loads elements of the `<objectName>` list from the current folderRCPT. `<objectName>` can be "scriptFilters", "socketFilters" or "activeFilters".

7. **Name** | folderRcpt_listAdd  
**Syntax** | MAP folderRcpt_listAdd(MAP objectValues, STRING objectName)  
**Explanation** | Adds `<objectValues>` into `<objectName>` list of the current folderRCPT. If the map object returned by this method is not empty (add failure) then it contains all wrong properties from `<objectValues>`.

8. **Name** | folderRcpt_listRemove  
**Syntax** | folderRcpt_listRemove(STRING objectKey, STRING objectName)  
**Explanation** | Removes element `<objectKey>` from `<objectName>` list, from current folderRCPT.

9. **Name** | folderRcpt_listLoadElement  
**Syntax** | MAP folderRcpt_listLoadElement(STRING objectKey, STRING objectName)  
**Explanation** | Loads the element identified by `<objectKey>` from `<objectName>` list, from current folderRCPT.

10. **Name** | folderRcpt_listUpdateElement  
**Syntax** | MAP folderRcpt_listUpdateElement(MAP objectValues, STRING objectKey, STRING objectName)  
**Explanation** | Updates the object identified by `<objectKey>`, from `<objectName>` list, with attributes `<objectValues>`. If the map
Public folder specific methods

1. **Name**  
   **publicFolder_load**  
   **Syntax**  
   publicFolder_load(STRING domainName)  
   **Explanation**  
   Loads the public folder from <domainName>.

2. **Name**  
   **publicFolder_get**  
   **Syntax**  
   MAP publicFolder_get(STRING valuesName)  
   **Explanation**  
   Loads <valuesName> object from the current public folder. <valuesName> can be "values" or "mboxContainerQuota".

3. **Name**  
   **publicFolder_set**  
   **Syntax**  
   MAP publicFolder_set(MAP values, STRING valuesName)  
   **Explanation**  
   Sets <values> map object to <valuesName> object of current public folder. If the map object returned by this method is not empty (set failure) then it contains all wrong properties from <values>.

4. **Name**  
   **publicFolder_commit**  
   **Syntax**  
   publicFolder_commit()  
   **Explanation**  
   Commits changes of current public folder.

5. **Name**  
   **publicFolder_listLoad**  
   **Syntax**  
   ARRAY publicFolder_listLoad(STRING objectName)  
   **Explanation**  
   Loads elements of the <objectName> list from the current public folder. <objectName> can be only "mboxes".

6. **Name**  
   **publicFolder_listAdd**  
   **Syntax**  
   MAP publicFolder_listAdd(MAP objectValues, STRING objectName)  
   **Explanation**  
   Adds <objectValues> into <objectName> list of the current public folder. If the map object returned by this method is not empty (add failure) then it contains all wrong properties from <objectValues>.

7. **Name**  
   **publicFolder_listRemove**  
   **Syntax**  
   publicFolder_listRemove(STRING objectKey, STRING objectName)
### 8. publicFolder_listLoadElement

**Syntax**

```plaintext
MAP publicFolder_listLoadElement(STRING objectKey, STRING objectName)
```

**Explanation**

Removes element `<objectKey>` from `<objectName>` list, from current public folder.

### 9. publicFolder_listUpdateElement

**Syntax**

```plaintext
MAP publicFolder_listUpdateElement(MAP objectValues, STRING objectKey, STRING objectName)
```

**Explanation**

Loads the element identified by `<objectKey>` from `<objectName>` list, from current public folder.

### Graphic specific methods

**Name**

`graphic_show`

**Syntax**

```plaintext
graphic_show(STRING graphicName)
```

**Explanation**

Sends the content of the graphic identified by `<graphicName>`. WARNING: this method MUST be called alone, without any other HTML or HSP code, because it sends binary data (image files) to the client and resets HTTP headers.

### smtpFilter Specific Methods

#### 1. smtpFilter_isInited

**Syntax**

```plaintext
smtpFilter_isInited()
```

**Explanation**

Returns true if the smtp filters have been inited (parseFile has been succesfull or overwriteFile was called).

#### 2. smtpFilter_loadUsage

**Syntax**

```plaintext
MAP smtpFilter_loadUsage()
```

**Explanation**

Loads a vector of events where the smtp filters and conditions can be used.

#### 3. smtpFilter_loadKeywords

**Syntax**

```plaintext
MAP smtpFilter_loadKeywords()
```

**Explanation**

Retrieves the list of sfl keywords that cannot be used as filter names.
## 4. smtpFilter_parseFile

<table>
<thead>
<tr>
<th>Name</th>
<th>smtpFilter_parseFile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>MAP smtpFilter_parseFile()</td>
</tr>
</tbody>
</table>
| Explanation   | Parses the contents of the SMTP filter script file, `<workdir>/filters/smtpFilters.scripts`. If the file is successfully parsed, the returned MAP contains the following property:  
  - STRING success  
  If the smtp filters script file cannot be successfully parsed, the method returns a MAP with two properties:  
  - STRING error  
  - NUMBER line - the number of the line where parsing failed |

## 5. smtpFilter_overwriteFile

<table>
<thead>
<tr>
<th>Name</th>
<th>smtpFilter_overwriteFile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>smtpFilter_overwriteFile()</td>
</tr>
<tr>
<td>Explanation</td>
<td>If the existing SMTP filter script file (<code>&lt;workdir&gt;/filters/smtpFilters.scripts</code>) content cannot be successfully parsed, this method can be used in order to overwrite the file with a new one edited by the user with the smtp filter wizard.</td>
</tr>
</tbody>
</table>

## 6. smtpFilter_loadFile

<table>
<thead>
<tr>
<th>Name</th>
<th>smtpFilter_loadFile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING smtpFilter_loadFile()</td>
</tr>
<tr>
<td>Explanation</td>
<td>It returns the content of the SMTP filter script file, <code>&lt;workdir&gt;/filters/smtpFilters.scripts</code>.</td>
</tr>
</tbody>
</table>

## 7. smtpFilter_saveFile

<table>
<thead>
<tr>
<th>Name</th>
<th>smtpFilter_saveFile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>smtpFilter_saveFile(STRING content)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Updates the SMTP filter script file with <code>&lt;content&gt;</code> as text.</td>
</tr>
</tbody>
</table>
### Name: smtpFilter_load

<table>
<thead>
<tr>
<th>Syntax</th>
<th>MAP smtpFilter_load(NUMBER filterId)</th>
</tr>
</thead>
</table>

**Explanation:** This method loads the filter with the specified id in the returned MAP. The MAP element has the same properties as above (smtpFilter_loadList method).

### Name: smtpFilter_enable

<table>
<thead>
<tr>
<th>Syntax</th>
<th>smtpFilter_enable(NUMBER filterId)</th>
</tr>
</thead>
</table>

**Explanation:** Enables the filter with the specified id.

### Name: smtpFilter_disable

<table>
<thead>
<tr>
<th>Syntax</th>
<th>smtpFilter_disable(NUMBER filterId)</th>
</tr>
</thead>
</table>

**Explanation:** Disables the filter with the specified id.

### Name: smtpFilter_add

<table>
<thead>
<tr>
<th>Syntax</th>
<th>smtpFilter_add(MAP filter)</th>
</tr>
</thead>
</table>

**Explanation:** Adds the specified filter to the list of SMTP filters. The MAP must contain the following properties:
- ARRAY actions - each element has MAP type and has these properties:
  - STRING type
  - STRING value
- ARRAY conditions - each element has MAP type and has these properties:
  - STRING type
  - STRING value
- STRING enabled - can be "true" or "false"
- STRING match - can be "all" or "any"
- STRING name

### Name: smtpFilter_update

<table>
<thead>
<tr>
<th>Syntax</th>
<th>smtpFilter_update(NUMBER filterId, MAP filter)</th>
</tr>
</thead>
</table>

**Explanation:** Updates the filter with the specified id with the values from the MAP variable. The MAP variable must contain the same properties as above.
### HSP – AXIGEN Server Side Scripting Language

#### 14.

<table>
<thead>
<tr>
<th>Name</th>
<th>smtpFilter_remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>smtpFilter_remove(NUMBER filterId)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Removes the filter with the specified id.</td>
</tr>
</tbody>
</table>

#### 15.

<table>
<thead>
<tr>
<th>Name</th>
<th>smtpFilter_commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>smtpFilter_commit()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Commits the changes made to the SMTP filters.</td>
</tr>
</tbody>
</table>

#### 16.

<table>
<thead>
<tr>
<th>Name</th>
<th>smtpFilter_reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>smtpFilter_reset()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Resets the changes made (but not committed) to the current SMTP filter.</td>
</tr>
</tbody>
</table>

#### 17.

<table>
<thead>
<tr>
<th>Name</th>
<th>smtpFilter_setOrder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>smtpFilter_setOrder(STRING order)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Sets the filters order. The &lt;order&gt; parameter must contain the id's of the filters in the desired order, comma separated.</td>
</tr>
</tbody>
</table>

Example for 3 filters:

```
smtpFilter_setOrder("2, 3, 1")
```

### Sieve filters specific methods

#### 1.

<table>
<thead>
<tr>
<th>Name</th>
<th>avFilter_listLoad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY avFilter_listLoad()</td>
</tr>
</tbody>
</table>
| Explanation      | Returns a list of detected antivirus filters. If <reset> is "true" then the list of antivirus filters is detected again on server. Each object contains the following properties:  
  - STRING name: the name of the antivirus filter  
  - STRING status  
  - STRING path  
  - STRING address |

#### 2.

<table>
<thead>
<tr>
<th>Name</th>
<th>sieveFilter_init</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sieveFilter_init(STRING path)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Inits the sieve filter wizard by setting the filename that contains the sieve scripts.</td>
</tr>
</tbody>
</table>
### 3. sieveFilter_isInited

<table>
<thead>
<tr>
<th>Name</th>
<th>sieveFilter_isInited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sieveFilter_isInited()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns true if the sieve filter wizard has been inited.</td>
</tr>
</tbody>
</table>

### 4. sieveFilter_loadBlackList

<table>
<thead>
<tr>
<th>Name</th>
<th>sieveFilter_loadBlackList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY sieveFilter_loadBlackList()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads the list of email addresses that have been blacklisted. The ARRAY contains STRINGS.</td>
</tr>
</tbody>
</table>

### 5. sieveFilter_addToBlackList

<table>
<thead>
<tr>
<th>Name</th>
<th>sieveFilter_addToBlackList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sieveFilter_addToBlackList(STRING email)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Adds an email address to the black list</td>
</tr>
</tbody>
</table>

### 6. sieveFilter_removeFromBlackList

<table>
<thead>
<tr>
<th>Name</th>
<th>sieveFilter_removeFromBlackList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sieveFilter_removeFromBlackList(NUMBER email_index)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Removes the entry with the email_index position from the black list</td>
</tr>
</tbody>
</table>

### 7. sieveFilter_loadList

<table>
<thead>
<tr>
<th>Name</th>
<th>sieveFilter_loadList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY sieveFilter_loadList()</td>
</tr>
</tbody>
</table>
| Explanation  | Loads the list of WebMail filters for the current account. Every element has MAP type and contain the following properties:  
- STRING name: the name of the filter  
- NUMBER id: the ID of the filter  
- STRING enabled: can be "yes" or "no". |

### 8. sieveFilter_add

<table>
<thead>
<tr>
<th>Name</th>
<th>sieveFilter_add</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sieveFilter_add(MAP filter)</td>
</tr>
</tbody>
</table>
| Explanation  | Adds the filter with the properties specified by <filter>. The MAP can contain the following properties:  
- STRING name: name of the filter  
- STRING match: can be "all" or "any"  
- ARRAY expressions: expressions to be matched; each element has MAP type and can contain the following properties:  
  - STRING header: header to be matched; can be "no header", "subject", "to", "cc", "to or cc", "from", "size", "custom"  
  - STRING customHeader: custom header to be matched; exists only if <header> is "custom"  
  - STRING value: value to be matched on header |
STRING condition: can be "is greater than", "is lower than" if <header> is "size" or the <customHeader> is defined and is considered to have a numerical value, or it can be "contains", "is", "begins with", "ends with", "does not contain", "is not", "does not begin with", "does not end with" for any other <header> value

- ARRAY actions: a list of actions to be taken when matching occurs; each element has MAP type and can contain the following properties:
  - STRING type: can be "no action", "move", "copy", "delete", "forward" or "vacation"

If <type> is "vacation" then the following properties must exist for an action element:
- STRING subject: subject of the message
- STRING text: body of the message
- STRING days: number of days after another message will be sent to the same address

If <type> is not "vacation", then the following property must be defined:
- STRING destination: if <type> is "forward", then <destination> property should contain the e-mail address where the mail will be forwarded; in case <type> is "move" or "copy", then it must contain the name of the folder where the message will be moved or copied.

For more clarity, examples will follow for each type of filter that can be created. The example below adds a simple filter, which forwards any message that contains the string "[Jokes]" to "jokes@localedomain" email address.

```xml
<%MAP filter%>
<%filter.name="Jokes"%>
<%filter.match="all"%>
<%ARRAY expressions%>
  <%MAP expression%>
    <%expression.header="subject"%>
    <%expression.value="[Jokes]"%>
    <%expression.condition="contains"%>
    <%push(expressions,expression)%>
  <%filter.expressions=expressions%>
<%ARRAY actions%>
  <%MAP action%>
    <%action.type="forward"%>
    <%action.destination="jokes@localedomain"%>
    <%push(actions,action)%>
  <%filter.actions=actions%>
<%IF filter_add(filter)%>
  Filter added
```

Copyright © 2008 Gecad Technologies S.A.
http://www.axigen.com
The following example illustrates filtering on a custom header. It checks the custom header "X-Spam" and if it is "***" then it moves the message into "Spam" folder.

```%
<%MAP filter%
<%filter.name="Spam"%>
<%filter.match="all"%>
<%ARRAY expressions%>
    <%MAP expression%>
        <%expression.header="custom"%>
        <%expression.customHeader="X-Spam"%>
        <%expression.value="***"%>
        <%expression.condition="is"%>
        <%push(expressions,expression)%>
    <%filter.expressions=expressions%>
<%ARRAY actions%>
    <%MAP action%>
        <%action.type="move"%>
        <%action.destination="Spam"%>
        <%push(actions,action)%>
<%filter.actions=actions%>
<%IF filter_add(filter)%>
    Filter added
<%ELSE%>
    Adding filter failed
<%ENDIF%>
```

The last example illustrates the creation of a responder (<type> is "vacation").

```%
<%MAP filter%
<%filter.name="Out of Office"%>
<%filter.match="any"%>
<%ARRAY expressions%>
<%filter.expressions=expressions%>
<%ARRAY actions%>
    <%MAP action%>
        <%action.type="vacation"%>
        <%action.subject="Out of office reply"%>
        <%action.text="I will be out of office until the 20th October."%>
        <%action.days=7%>
        <%push(actions,action)%>
```
9. Name | sieveFilter_load  
| Syntax | MAP sieveFilter_load(NUMBER filterId)  
| Explanation | Loads the filter with the ID <filterId>. This is the current active filter. The returned MAP element contains the same properties as for filter_add, with the same meanings, and two more properties:
- STRING enabled: it can be "true" or "false"
- NUMBER id: the ID of the filter.

10. Name | sieveFilter_enable  
| Syntax | sieveFilter_enable(NUMBER filterId)  
| Explanation | Enables the filter with the ID <filterId>.

11. Name | sieveFilter_disable  
| Syntax | sieveFilter_disable(NUMBER filterId)  
| Explanation | Disables the filter with the ID <filterId>.

12. Name | sieveFilter_update  
| Syntax | sieveFilter_update(NUMBER filterId, MAP filter)  
| Explanation | Updates the properties contained in <filter> for the filter specified by <filterId> (MAP element has the same properties as for #filter_load method).

13. Name | sieveFilter_remove  
| Syntax | sieveFilter_remove(NUMBER filterId)  
| Explanation | Removes the filter identified by <filterId> from the list of filters.

14. Name | sieveFilter_commit  
| Syntax | sieveFilter_commit()  
| Explanation | Commits the changes made on the current active filter.
15. | Name          | sieveFilter_reset |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sieveFilter_reset()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Resets the changes made on the current active filter.</td>
</tr>
</tbody>
</table>

16. | Name          | sieveFilter_setOrder |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>sieveFilter_setOrder(STRING order)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Sets the filters order. The &lt;order&gt; parameter must contain the id's of the filters in the desired order, comma separated.</td>
</tr>
</tbody>
</table>

Example for 3 filters:
```
sieveFilter_setOrder("2, 3, 1")%>
```

**AAACL specific methods**

1. | Name          | auser_getLoginName |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING auser_getLoginName()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns the current administrative user’s name logged in.</td>
</tr>
</tbody>
</table>

2. | Name          | aacl_listLoad |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>ARRAY aacl_listLoad(STRING listType)</td>
</tr>
<tr>
<td>Explanation</td>
<td>Loads the list of &quot;agroups&quot;(name and description), &quot;ausers&quot;(name and description), &quot;resources&quot;(name and type), &quot;serverPermissions&quot;, &quot;domainPermissions&quot;.</td>
</tr>
</tbody>
</table>

3. | Name          | principal_create |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>principal_create(STRING prType, STRING prName[, STRING password])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Creates a new principal with the name &lt;prName&gt;. His &lt;prType&gt; can be &quot;auser&quot;, or &quot;agroup&quot;. &lt;password&gt; is required for &quot;auser&quot;.</td>
</tr>
</tbody>
</table>

4. | Name          | principal_remove |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>principal_remove(STRING prType, STRING prName[, STRING forced])</td>
</tr>
<tr>
<td>Explanation</td>
<td>Removes a principal with the name &lt;prName&gt;. His &lt;prType&gt; can be &quot;auser&quot; or &quot;agroup&quot;. The &quot;forced&quot; parameter is optionally used for &quot;agroup&quot;; an administrative group with children can be removed only if &lt;forced&gt; is &quot;true&quot;; &lt;forced&gt; is by default considered to be &quot;false&quot;.</td>
</tr>
</tbody>
</table>
### 5. `principal_load`

**Syntax**

MAP principal_load(STRING prType, STRING prName)

**Explanation**

Returns a map filled with principal's ("auser" or "agroup") properties:
- STRING name (principal's name)
- STRING descriptions (principal's description)
- ARRAY(STRING) parents (principal's parents)
- ARRAY(MAP) resources (principal's resources)
  - STRING name (resource name)
  - STRING type (resource type)
  - ARRAY(STRING) grant (list of permissions granted on this resource)
  - ARRAY(STRING) deny (list of permissions denied on this resource)

### 6. `principal_loadEffectivePermissions`

**Syntax**

MAP principal_loadEffectivePermissions(STRING prType, STRING prName, STRING resType, STRING resName)

**Explanation**

Returns a map filled with principal's ("auser" or "agroup") <prName> effective permissions on the given resource("server" or "domain") <resName>:
- ARRAY(STRING) grant (list of granted permissions)
- ARRAY(STRING) deny (list of denied permissions)

### 7. `principal_update`

**Syntax**

principal_update(STRING prType, STRING prName, MAP props)

**Explanation**

Updates principal's properties with <props> ("name", "description" and "password" for auser; "name" and "description" for a group).

### 8. `principal_addParent`

**Syntax**

principal_addParent(STRING prType, STRING prName, STRING parent)

**Explanation**

Adds <parent> in <prName> principal's parents list.

### 9. `principal_removeParent`

**Syntax**

principal_removeParent(STRING prType, STRING prName, STRING parent)

**Explanation**

Removes <parent> from <prName> principal's parents list.
### 10. `principal_listLoadAllParents`

**Syntax**

```
ARRAY principal_listLoadAllParents(STRING prType, STRING prName)
```

**Explanation**

Loads all parents (not only direct parents) of the `<prName>` principal. The result contains:

- `STRING name`
- `NUMBER level`

### 11. `agroup_listLoadMembers`

**Syntax**

```
ARRAY agroup_listLoadMembers(STRING groupName)
```

**Explanation**

Loads the members of the `<groupName>` administrative group. The result contains:

- `STRING name`
- `NUMBER level`
- `STRING type` ("auser" or "agroup")

### 12. `principal_setPermissions`

**Syntax**

```
principal_setPermissions(STRING prType, STRING prName, ARRAY permissions, STRING resType, STRING resName)
```

**Explanation**

Sets new `<permissions>` for principal `<prName>` on resource `<resName>`. `<permissions>` contains maps with properties:

- `STRING name` (permission name)
- `STRING type` (permission type: "grant", "deny", or "revoke")

### 13. `resource_listLoadPrincipals`

**Syntax**

```
MAP resource_listLoadPrincipals(STRING resType)
```

**Explanation**

Returns the principals list for the resource with type `<resType>`. `<resType>` can be "server" or "domain". If the resource is a domain, `domain_load` method must be called firstly. The result contains:

- `STRING name` (principal name)
- `STRING type` (principal type: "auser" or "agroup")
- `ARRAY(STRING)` grant (list of grant permissions)
- `ARRAY(STRING)` deny (list of deny permissions)

### 14. `resource_checkPermission`

**Syntax**

```
resource_checkPermission(STRING permission, STRING resType)
```

**Explanation**

Returns true if the administrative user logged in has the `<permission>` on the current resource of type `<resType>`.
<resType> can be "server" or "domain". In case of "domain", domain_load method must be called before.

User Classes specific methods

1. **Name**: userClass_create  
   **Syntax**: userClass_create(STRING userClassName, STRING domainName)  
   **Explanation**: Creates a new user class, <userClassName>, for <domainName>. This user class is the new current user class.

2. **Name**: userClass_load  
   **Syntax**: userClass_load(STRING userClassName, STRING domainName)  
   **Explanation**: Loads <userClassName> from <domainName>. This is the new current user class.

3. **Name**: userClass_get  
   **Syntax**: MAP userClass_get(STRING valuesName)  
   **Explanation**: Loads <valuesName> object from the current user class. <valuesName> can be "values", "limits", "inheritedLimits", "quota", "inheritedQuota" or "filterData".

4. **Name**: userClass_set  
   **Syntax**: MAP userClass_set(MAP values, STRING valuesName)  
   **Explanation**: Sets <values> map object to <valuesName> object of current user class. If the map object returned by this method is not empty (set failure) then it contains all wrong properties from <attributes>. <valuesName> can be "values", "limits", "quota" or "filterData".

5. **Name**: userClass_commit  
   **Syntax**: userClass_commit()  
   **Explanation**: Commits changes of current user class.

6. **Name**: userClass_listLoad  
   **Syntax**: ARRAY userClass_listLoad(STRING objectName)  
   **Explanation**: Loads elements of the <objectName> list from the current user class. <objectName> can be: "scriptFilters", "socketFilters", "activeFilters".
### 7. userClass_listAdd

**Syntax**

```plaintext
MAP userClass_listAdd(MAP objectValues, STRING objectName)
```

**Explanation**

Adds `<objectValues>` into `<objectName>` list of the current user class. If the map object returned by this method is not empty (add failure) then it contains all wrong properties from `<objectValues>`. `<objectName>` can be: "scriptFilters", "socketFilters", "activeFilters".

### 8. userClass_listRemove

**Syntax**

```plaintext
userClass_listRemove(STRING objectKey, STRING objectName)
```

**Explanation**

Removes element `<objectKey>` from `<objectName>` list, from current user class. `<objectName>` can be: "scriptFilters", "socketFilters", "activeFilters".

### 9. userClass_listLoadElement

**Syntax**

```plaintext
MAP userClass_listLoadElement(STRING objectKey, STRING objectName)
```

**Explanation**

Loads the element identified by `<objectKey>` from `<objectName>` list, from current user class. `<objectName>` can be: "scriptFilters", "socketFilters", "activeFilters".

### 10. userClass_listUpdateElement

**Syntax**

```plaintext
MAP userClass_listUpdateElement(MAP objectValues, STRING objectKey, STRING objectName)
```

**Explanation**

Updates the object identified by `<objectKey>`, from `<objectName>` list, with attributes `<objectValues>`. If the map object returned by this method is not empty (update failure) then it contains all wrong properties from `<objectValues>`. `<objectName>` can be: "scriptFilters", "socketFilters", "activeFilters".

#### Date functions

### timezone_loadList

**Syntax**

```plaintext
ARRAY timezone_loadList()
```

**Explanation**

Returns a list of strings. Each string represents a timezone.
2.4.5 WebmailProxy Specific Methods

1.

<table>
<thead>
<tr>
<th>Name</th>
<th>isMobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>isMobile()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns true if the client was detected as a mobile browser.</td>
</tr>
</tbody>
</table>

2.

<table>
<thead>
<tr>
<th>Name</th>
<th>connection_isSecure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING connection_isSecure()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns true if the current connection is secure (SSL) or false otherwise.</td>
</tr>
</tbody>
</table>

3.

<table>
<thead>
<tr>
<th>Name</th>
<th>connection_getHostNameResolver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING connection_getHostNameResolver()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns the domain name of the host. &quot;Host&quot; HTTP header MUST be sent by the client in order to resolve host name to domain name.</td>
</tr>
</tbody>
</table>

4.

<table>
<thead>
<tr>
<th>Name</th>
<th>connection_getUrlRedirect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>STRING connection_getUrlRedirect()</td>
</tr>
<tr>
<td>Explanation</td>
<td>Returns a url redirection string for the current connection which is used to redirect from a plain to a secure connection.</td>
</tr>
</tbody>
</table>

2.5 Language Constraints

1. HSP Language supports a single level for indexed elements. The solution to avoid this constraint is illustrated in the following example:

```hsp
<%FOR i = 1 to n%>
    <% a_i = a[i] %>
    <%FOR j = 1 to m%>
        <% a_i[j] %>
    <%ENDFOR%>
<%ENDFOR%>
```

2. The only way you can print a value is:

```hsp
PRINT = "<%" PARAM "">"
```

Example:

```hsp
<%i = 10%>
<%INC(i)%>
<%i%>
```
2.6 HSP comments
If you need to insert comments in your HSP code, use the following characters: "<%" "/*" COMMENT "*/" "%>"", where COMMENT is a set of accepted chars.

Example:
<%i = 10%>
<%/*This is a comment*/%>
<%INC(i)%>
<%i%>

2.7 Other specifications
Besides all the definitions given so far, two other rules apply:

1. A numeric value has a "signed int" type (4 bytes). Overflow operations are allowed.
2. GET, POST and COOKIE are global MAP variables and contain values sent via a GET, POST command or a Cookie HTTP header.

2.8 HSP limitations
In HSP, the two following limitations apply:

1. VAR name size: 256
2. STRING size: 4096 (on declaration)