AXIGEN® Server-Side Scripting Language

Product version: beta2

Last update on:
6/29/2005 7:40:16 PM
Document version: 1.0

Copyright & trademark notices

This edition/ notices applies to version beta2 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 Ltd. products, programs, or services do not imply that GECAD TECHNOLOGIES Ltd. intends to make these available in all countries in which GECAD TECHNOLOGIES Ltd. operates. Evaluation and verification of operation in conjunction with other products, except those expressly designated by GECAD TECHNOLOGIES Ltd., are the user's responsibility. GECAD TECHNOLOGIES Ltd. 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 Ltd. marketing department, sales@AXIGEN.com .

Copyright Acknowledgement

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 Ltd. will not be responsible for any loss, costs or damages incurred due to the use of this documentation.

GECAD TECHNOLOGIES™ and **AXIGEN®** are trademarks of GECAD TECHNOLOGIES Ltd. Other company, product or service names may be trademarks or service marks of others.

AXIGEN® is a SOFTWARE PRODUCT of GECAD TECHNOLOGIES Ltd.

If you brake any of this copyright & trademark notices, GECAD TECHNOLOGIES Ltd. reserves its right to take legal action against you for damages.

GECAD TECHNOLOGIES Ltd.

10A Dimitrie Pombei blvd, Connect Business Center, 2nd fl., Bucharest 2, ROMANIA; tel: +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

Table of contents

Part I: Introduction

Welcome

Congratulations on your decision to evaluate AXIGEN®**.** Currently this product is in beta phase, so it only includes some of the features it will finally include in the commercial version once released. For a complete description of the functions currently available in AXIGEN®, please refer to the Description section in this document. The commercial version of AXIGEN® will include SMTP, POP, IMAP and Webmail offering numerous other capabilities.

Structure and purpose of this document

This document is divided into 2 main sections and 1 appendixes:

- Part 1: Introduction contains general information about AXIGEN® and its producer, GECAD TECHNOLOGIES;
- Part II: HSP language description describes HSP, the proprietary server side scripting language used by AXIGEN Webmail module to generate HTML code
- Appendix A: Bug Report Form provides a form you can use to report any bugs in the product.

Scope

This document serves as a guide to administrators of the mail servers in companies where the Webmail module of AXIGEN version beta2 is installed and evaluated. The purpose of this document is to provide a description of the server-side scripting language implemented in the Webmail module.

Important: This document relates to a **beta** version of **AXIGEN®.** We would really appreciate your feedback on this beta version.

Audience and knowledge requirements

The intended audience for this document is represented by administrators of the mail servers in companies where the version beta2 of AXIGEN® application is installed and evaluated, administrators willing to expand the abilities of the Webmail module.

In order to build, extract and acquire the correct information from this manual, a regular audience should have:

- A detailed knowledge of general mail server abilities and functions.

- Good comprehension of the RFC requirements.
- Scripting skills, in order to use correctly AXIGEN® Webmail extension capabilities.

Related documentation

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

- AXIGEN User Manual - contains in-depth description of AXIGEN server configuration; and
- AXIGEN Online documentation.

About GECAD TECHNOLOGIES

Founded in 2001, GECAD TECHNOLOGIES is a company involved in technology research and project management, offering services initially for antivirus industry (GECAD has sold the RAV Antivirus IPR to Microsoft Inc. in 2003). Since then, we have concentrated on an innovative technology: messaging solutions.

GECAD TECHNOLOGIES is a member of GECAD Group.

Development Strategy

We create messaging technology incorporated in scalable solutions: from small companies to large ISPs. Our products are developed for demanding system administrators, people in a continuous quest for new ways to improve their solution.

Our policy is to continuously develop our product in a pace that follows closely our customers' needs. Based on platform independent core and independent running modules and accommodating the newest hardware architectures, our product is built for reliability and security, supplying ease for our customers.

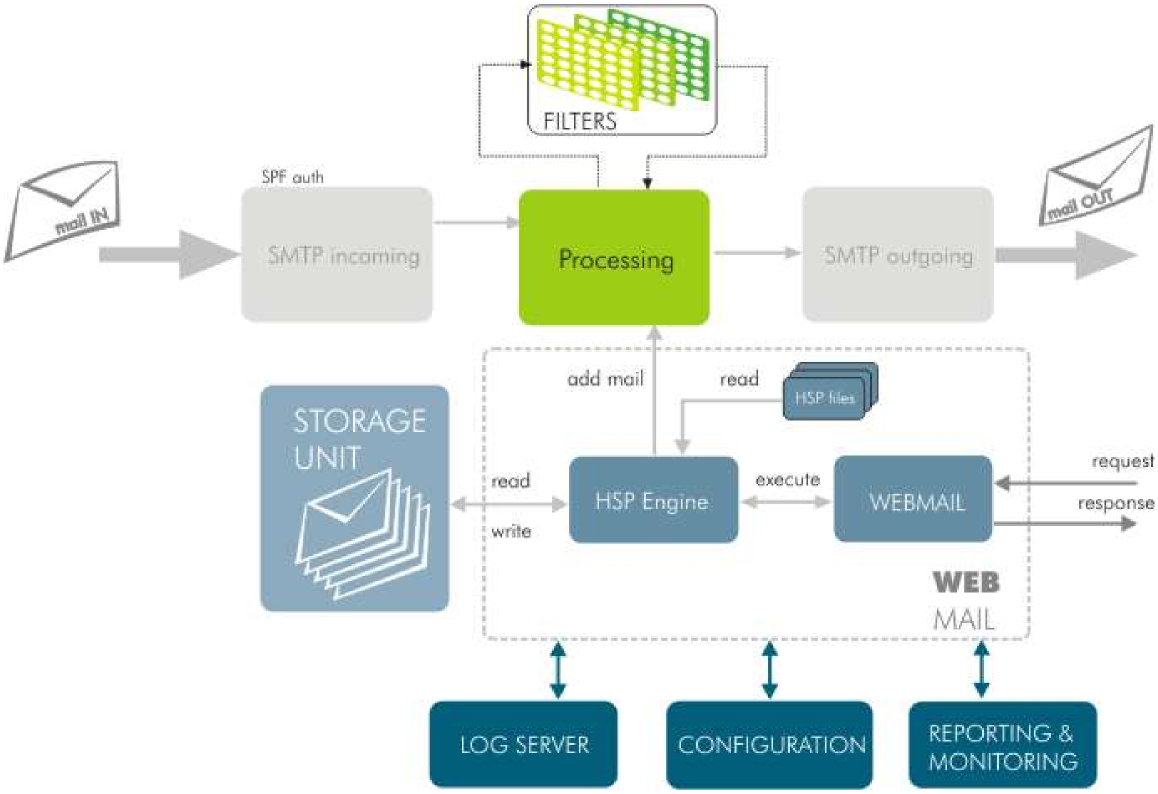
Technical support

For any details regarding the installation and the functionality of this product, please contact the local dealer you have bought the product from. If he does not offer you the adequate technical support, please feel free to contact AXIGEN Team directly and you will be offered technical support.

Part II: HSP Language description

About HSP

HSP is a proprietary server side scripting language used by AXIGEN Webmail module to generate HTML code.



HSP delimiters

HSP delimiters are "<%" and "%>", and you can use HSP syntax only between these 2 separators.

Example:

```
<html>
<% FOR i = 1 TO 10 %>
<b> <%i%> </b>
<br>
<% ENDFOR %>
</html>
```

Language structure

Variables

A variable can have one of the following types:

- NUMBER
- STRING
- MAP (associative array)
- ARRAY

Variable declarations

NUMBER and STRING variables are declared automatically, when they are used for the first time.

Example:

```
<% a = "some string" %>
<% n = 10 %>
```

Only MAP and ARRAY variables must be declared explicitly.

Example:

```
<%MAP person%>
<%ARRAY obj%>
```

Declaration of NUMBER-type variables

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

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

Declaration of STRING-type variables

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

Example:

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

Declaration of ARRAY-type variables

Elements of an array can have any of the HSP types (NUMBER, STRING, MAP or ARRAY).

You can access, modify, push or pop an element of an array.

Example:

```
<!-- 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 accessto an ARRAY's element must have the expression:

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

Declaration of MAP-type variables

Elements of a map can have any of the HSP types.

You can access, modify, or insert an element of a map.

Example:

```
<%s = "some string"%>

<!-- array declaration -->
<%MAP person%>

<!-- insert elements of any type -->
<%person.name = s%>
<%person.age = 24)%>

<!-- modify element -->
<%person.age = 23%>
```

```
<!-- get element value -->
<%a = person.age%>
```

The access to a MAP's element must have the expression:

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

Instructions

IF - ELSE statement

The IF-ELSE statement has the following classical structure:

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

FOR statement

The FOR statement has the following classical structure:

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

Example:

```
<%FOR el = 1 TO 10 INDEXED BY i%>
Element(<%i%>) = <%el%> <br>
<%ENDFOR%>
```

FOREACH statement

The FOREACH statement has the following classical structure:

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

Example:

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

Example:

```
<%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 structure:

```
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 in a HSP file another HSP source. It has the following structure:

```
INCLUDE_EXPR = "<%" "INCLUDE" PARAM "%>"
```

where PARAM represents the path of a HSP file that will be inserted in the current HSP.

The path is relative to webmail "path" (configured).

Only in private zone, you can insert a HSP file from another folder.

HTTP server rejects paths like "*/../*".

Inclusion depth is limited to 16.

Note: Any HSP file MUST have a .hsp extension.
Example:
<%INCLUDE "hello.hsp"%>

Operators

HSP Language uses only "+" operator to concatenate values.

Example:

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

Call Methods

The HSP call methods have the following syntax:

```
METHOD = VARIABLE "(" PARAM_LIST ")"

PARAM_LIST = PARAM | (PARAM "," PARAM)

PARAM = (NUMBER | STRING | VARIABLE | ARRAY_ACCESS |
MAP_ACCESS | CONCAT)

CONCAT = (PARAM "+" CONCAT) | (PARAM "+" PARAM)


VAR = VARIABLE | ARRAY_ACCESS | MAP_ACCESS
```

The HSP module has 3 sets of methods.

- the first set is shared by Webmail and Webadmin modules and contains general language methods;

- the second one contains Webmail specific methods;
- the third one contains Webadmin related methods.

List of general HSP Language methods:

INC method

```
INC(VAR param1 [, PARAM step])
```

Description:

Increments parameter's value by one step. If the next step is missing, param1 will be incremented by 1.

Example:

```
<%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 -->
```

DEC method

```
DEC(VAR param [, PARAM step])
```

Description:

Decrements parameter by step. If step is missing, or if it has a NULL value, param1 will be decremented by 1.

EQ method

Usage:

```
EQ(PARAM param1, PARAM param2)
```

Description:

Returns true if param1 is equal to param2 (case INSENSITIVE), returns false otherwise.

Example:

```
<%a = 1%>
```

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

EQCASE method

Usage:

```
EQCASE(PARAM param1, PARAM param2)
```

Description:

Returns true if param1 is equal to param2 (case SENSITIVE), returns false otherwise.

GT method

Usage:

```
GT(PARAM param1, PARAM param2)
```

Description:

Returns true if param1 > param2.

GTE method

Usage:

```
GTE(PARAM param1, PARAM param2)
```

Description:

Returns true if param1 >= param2.

LT method

Usage:

```
LT(PARAM param1, PARAM param2)
```

Description:

Returns true if param1 < param2.

LTE method

Usage:

```
LTE(PARAM param1, PARAM param2)
```

Description:

Returns True if param1 <= param2.

Note: param1 and param2 are casted to NUMBER

EVAL method

Usage:

```
EVAL(PARAM string)
```

Description:

Returns the value of the variable that has the name <string>.

Example:

```
<%i2 = 4%>
<%a = EVAL("i"+2)%>
<%a%> <!-- 4 -->
```

EXISTS method

Usage:

```
EXISTS(VAR param)
```

Description:

Returns true if <param> exists.

ISEMPTY method

Usage:

```
ISEMPTY(VAR param)
```

Description:

Returns true if value of <param> is empty; if <param> is an ARRAY or a MAP returns true if it has no element.

POP method

Usage:

```
POP(VAR el, VAR A)
```

Description:

Pops an element of array A and stores his value into <el>.

PUSH method

Usage:

```
PUSH(VAR A, VAR el)
```

Description:

Pushes element <el> into ARRAY <A>.

GETSIZE

Usage:

```
GETSIZE(VAR param)
```

Description:

Returns size of <param> (<param> can be an ARRAY or a MAP).

STR_ESCAPE

Usage 1:

```
STR_ESCAPE(PARAM arg1)
```

Description:

Escapes string <arg1> and prints its value.

Usage 2:

```
STR_ESCAPE(VAR arg1, PARAM arg2])
```

Escapes string <arg2>; arg1 stores the value.

Example:

```
<%a = "test\"test"%> <!-- test"test -->

<%STR_ESCAPE(a)%>
```

is equivalent to:

```
<%str = STR_ESCAPE(a)%>
```

```
<%str%><!-- test\"test -->
```

ENCODE_URL

Description:

Transforms non-alphanumeric characters in %{hex}{hex} and spaces in "+"

Example: "." becomes "%2E".

Usage 1:

```
ENCODE_URL(PARAM arg1])
```

Description:

Encodes string <arg1> in url format and prints its value.

Usage 2:

```
ENCODE_URL(VAR arg1, PARAM arg2)
```

Encodes string <arg2> in url format. <arg1> stores the value.

Example:

```
<%str = "a.b c"%>
<%encode_url(str)%> <!--  a%2E+c -->
```

ENCODE_HTML

Action: '<' becomes "&lt;", '>' becomes "&gt;" and "&" becomes "&amp;".

Usage 1:

```
ENCODE_HTML(PARAM arg1)
```

Description:

Encodes string <arg1> in html format and prints its value.

Usage 2:

```
ENCODE_HTML(VAR arg1, PARAM arg2)
```

Description:

Encodes string <arg2> in html formatr; <arg1> stores the value.

Example:

```
<%str = "<html>"%>
<%ENCODE_HTML(str)%> <!--  &lt;html&gt; -->
```

STR_REPLACE

Usage 1:

```
STR_REPLACE(PARAM haystack, PARAM needle, PARAM newsub)\
```

Description:

Replaces all <needle> occurrences, from <haystack>, with <newsub> and prints the new value.

Usage 2:

```
STR_REPLACE(VAR  newstr,  PARAM  haystack,  PARAM  needle,
PARAM newsub)
```

Description:

Replaces all <needle> occurrences, from <haystack>, with <newsub> and stores the new value into <newstr>.

Example:

```
<%str = "abcabc"%>
<%STR_REPLACE(str, "ab", "a")%> <!-- acac  -->
```

NUMBER STR_LEN method

Usage:

```
NUMBER STR_LEN(PARAM str)
```

Returns size of string <str>.

Notes:	The syntax <%METHOD(el, PARAM_LIST)%> is the same with: <%el = METHOD(PARAM_LIST)%>. <el> MUST be VAR.
Important:	Method names are case insensitive.

List of Webmail specific methods

folder_create

Usage:

```
folder_create(PARAM folderName, PARAM parentPath)
```

Description:

Creates a folder with name <folderName>. The parent's folder will be <parentPath>. The root parent is "/".

folder_delete

Usage:

```
folder_delete(PARAM folderPath)
```

Description:

Deletes the folder with absolute path <folderPath>.

folder_move

Usage:

```
folder_move(oldFolderPath, newFolderParent)
```

Description:

Moves a folder with absolute path <oldFolderPath>. The new location will be <newFolderParent>.

folder_rename

Usage:

```
folder_rename(PARAM oldFolderPath, PARAM newFolderName)
```

Description:

Renames a folder with absolute path <oldFolderPath>. The new name will be <newFolderName>.

ARRAY folder_loadLocalList()

Usage:

```
ARRAY folder_loadLocalList()
```

Description:

Returns current user's local folders.

Every element has MAP type and the following properties:

- name: folder name
- path: absolute path
- level: folder level (needed for tree view)
- fid: folder ID
- count: number of mails from folder
- news: number of unseen mails from folder
- isSelectable: doesn't have a value and it's optional.
- hasChildren: no value, optional.

Example:

```
<%folder_list = folder_loadLocalList()%>
<%FOREACH folder IN folder_list%>
<%IF EXISTS(folder.isSelectable)%>
<%folder.name%>(<%folder.news%>/<%folder.count%>)
<%ELSE%>
<%folder.name%>
<%endif%>
<%ENDFOR%>
```

MAP folder_loadById

Usage:

```
MAP folder_loadById(PARAM folderId)
```

Description:

Returns the following properties for <folderId>.

- name: folder name
- path: absolute path
- count: number of mails from folder
- news: number of unseen mails from folder
- fid: folder id

- sortType: "subject", "from", "size", "date" or "incoming"
- sortOrder: "ascending" or "descending"

MAP folder_loadByPath

Usage:

```
MAP folder_loadByPath(PARAM folderPath)
```

Description:

Returns the following properties for <folderPath>.

- name: folder name
- path: folder absolute path
- count: number of mails from folder
- news: number of unseen mails from folder
- fid: folder id
- sortType: "subject", "from", "size", "date" or "incoming"
- sortOrder: "ascending" or "descending"

mail_compose

Usage:

```
mail_compose(MAP fields[, VAR action[, VAR fid, VAR
mid]])
```

Description:

Tries to send a composed mail. First it verifies its correctness. If mail_compose returns true the syntax is OK. If <fields> 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> is optional and can have one of the following values:

- "save": the mail is saved to "Drafts" folder
- "reply": the mail identified by <fid> and <mid> will be sent as "Reply"
- "replyall": the mail identified by <fid> and <mid> will be sent as "Reply to all"

- "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".

mail_copy

Usage:

```
mail_copy(PARAM folderId, PARAM mailId, PARAM newFolderId)
```

Description:

Copy the mail with id <mailId>, from the folder with id <folderId>, to the folder with id <newFolderId>.

mail_delete

Usage:

```
mail_delete(PARAM folderId, PARAM mailId)
```

Description:

Deletes the mail identified by <mailId> and <folderId>.

mail_move

Usage:

```
mail_move(PARAM folderId, PARAM mailId, PARAM newFolderId)
```

Description:

Moves the mail identified by <mailId> and <folderId>, to the folder with id <newFolderId>.

mail_setFlag

Usage:

```
mail_setFlag(PARAM folderId, PARAM mailId, PARAM flag)
```

Description:

Sets the <flag> for the mail identified by <mailId> and <folderId>; <flag> can be: "seen", "flagged" or "deleted". Any other values are ignored.

mail_unsetFlag

Usage:

```
mail_unsetFlag(PARAM folderId, PARAM mailId, PARAM flag)
```

Description:

Unsets the <flag> for the mail identified by <mailId> and <folderId>; <flag> can be: "seen", "flagged" or "deleted". Any other values are ignored.

MAP mail_load

Usage:

```
MAP mail_load(PARAM folderId, PARAM mailId)
```

Description:

Returns a MAP object that contains the headers of the mail identified by <mailId> and <folderId>.

MAP object properties:

- from
- subject
- to
- cc
- date
- seen: no value, optional
- flagged: no value, optional
- deleted: no value, optional
- next: next mail ID
- prev: previous mail ID
- hasNext: exists if current mail has a successor
- hasPrev: exists if current mail has a predecessor
- mid: mail Id
- att: exists only if mail has attachments
- replyto: used to fill "To" field in case of a "Reply" action
- replytoall: used to fill "Cc" field in case of a "Reply To All" action

ARRAY mail_loadList

Usage:

```
ARRAY mail_loadList(PARAM folderId, PARAM jumpIdx)
```

Description:

Returns the list of mails form the folder with id <folderId>. The first mail from the list has <jumpIdx> index (1, default).

Every element has MAP type and properties:

- subject
- from
- date
- size: mail size
- id: mail index
- mid: mail ID
- seen: no value, optional
- flagged: no value, optional
- deleted: no value, optional

mail_sendBinary

Usage:

```
mail_sendBinary(PARAM folderId, PARAM mailId)
```

Description:

Sends original mail (headers + body) in text/plain (RAW) format.

Warning:	Do not write any other HTML or HSP code before this method.

mail_sendHeaders

Usage:

```
mail_sendHeaders(PARAM folderId, PARAM mailId)
```

Description:

Sends only the headers of mail identified by <mailId> and <folderId>.

Warning: Do not write any other HTML or HSP code before this method.

ARRAY body_loadParts

Usage:

```
ARRAY body_loadParts(PARAM folderId, PARAM mailed)
```

Description:

Returns the body list of the mail identified by <mailId> and <folderId>.

Every element has MAP type and properties:

- bid: body id
- type: mime type
- subtype: mime subtype

The following properties are available only for message bodies:

- from
- subject
- to
- date

body_show

Usage:

```
body_show(PARAM folderId, PARAM mailId, PARAM bodyId,
PARAM typeShow)
```

Description:

This method will display the content of the body identified by <bodyId>, <mailId> and <folderId>. <bodyId> can be obtained with body_loadParts method. Only text, html or multipart/related bodies are displayed.

<typeShow> can have one of the following types:

- "text": the body will be displayed only if it is TEXT/nonHTML
- "text_indented": used for reply action; every line of text will be indented with "> " string
- "html": displays a body if it is of TEXT/HTML type. It's parsed and filtered for external links, scripts, etc.
- "html_graphics": enables graphics display for HTML bodies

ARRAY att_loadList

Usage:

```
ARRAY att_loadList(PARAM folderId, PARAM mailId)
```

Description:

Returns attachment list of the mail identified by <mailId> and <folderId>.

Every element has MAP type and the following properties:

- attid

- name: filename

- type: mime type

- subtype: mime subtype

att_send

Usage:

```
att_send(PARAM folderId, PARAM mailId, PARAM attId)
```

Description:

Sends 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.

att_sendCid

Usage:

```
att_sendCid(PARAM folderId, PARAM mailId, PARAM bodyId,
PARAM cid
```

Description:

Sends 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.

setHeaders

Usage:

```
setHeaders([PARM header1 [, PARAM header2[, ...]]])
```

Description:

Resets http headers (<header[i]> must have STRING values). It is used for a personalized response to the client.

Warning:	Do not write any other HTML or HSP code before this method

STRING getSessionId()

Usage:

```
STRING getSessionId()
```

Description:

Returns the current session key.

MAP upatt_loadList()

Usage:

```
MAP upatt_loadList()
```

Description:

Loads uploaded file list.

Properties:

- filename
- size
- id

upatt_delete

Usage:

```
upatt_delete(PARAM attUpId)
```

Description:

Deletes uploaded file with id <attUpId>.

upatt_emptyList()

Usage:

```
upatt_emptyList()
```

Description:

Empties list of uploaded attachments.

upatt_storeFwAtt

Usage:

```
upatt_storeFwAtt(PARAM folderId, PARAM mailId, PARAM disposition)
```

Description:

Appends the attachment list (from mail identified by <mailId> and <folderId>), to the mail that is forwarded. If <disposition> is equal to "attachment", than the whole mail will be attached

STRING config_getAccountName()

Usage:

```
STRING config_getAccountName()
```

Description:

Returns account name.

MAP config_loadPersonals()

Usage:

```
MAP config_loadPersonals()
```

Description:

Returns account's personals:

- firstname

- lastname

MAP config_loadSettings()

Usage:

```
MAP config_loadSettings()
```

Description:

Returns account's settings:

- pageSize: number of mails displayed on a page
- skin: user's Webmail skin
- confirmMaildelete: option for confirmation before mail deletion
- confirmFolderEmpty: option for confirmation before empty a folder

- saveToSent: option for saving a mail to "Sent" folder after send action
- deleteToTrash: option for moving mail to "Trash" folder after delete action

config_setPersonals method

Usage:

```
config_setPersonals(PARAM personalField, PARAM value)
```

Description:

Stores in config : <personalField> = <value>.

config_setSettings method

Usage:

```
config_setSettings(PARAM setField, PARAM value)
```

Description:

Stores in config : <setField> = <value>.

Language constraints

Single level indexing

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

Example:

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

Printing values

The only way you can print a value is:

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

Example:

```
<%i = 10%>
<%INC(i)%>
<%i%>
```

Errors

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.

Misspelling a method's names is not reported as an error!

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

A language error is generated when an instruction is not properly implemented.

Example:

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

HSP comments

```
"<%" [HSP_CODE] "/*" COMMENT "*/" [HS_PCODE] "%>"
```

HSP_CODE = any HSP syntax

COMMENT = set of accepted chars

Other specifications

A numeric value has a "signed int" type (4 bytes). Overflow operations are allowed.

GET, POST and COOKIE are global MAP variables and contain values sent via a GET, POST command or a Cookie HTTP header.

HSP limitations

VAR name size: 256

STRING size: 4096

Appendix A: Bug Report Form

For each bug you report for AXIGEN®, please use the following format to provide relevant information for our technical support department:

Software:

- OS, distribution, version, kernel version:
- glibc:

Hardware:

- Processor (freq/type/platform):
- Memory (total/free):
- Physical
- Virtual
- HDD (total/free):

Problem description:

Other Info:

- location of queue, storages and log dir