

Documentation for OpenEMM Script Actions (for EMM replace value 1 in method signature with your CompanyID)		Version 1.6			
HTTP Request	Methods	Method Description	Method Signature	Remarks	Response
requestParameters	<key>	Read HTTP request parameter from hashmap	\$requestParameters.<key>	get value of (HTML form) field (key in upper case)	String
requestParameters	put(<key>, <value>)	Write HTTP request parameter to hashmap	\$requestParameters.put(„String<key>“, „String <value>“)	manipulate value of (HTML form) field (key in upper case)	void
Object	Useful Methods of Object	Method Description	Method Signature	Remarks	Response
Customer	get<key>()	Get parameter of current customer	\$Customer.get<key>()	Replace <key> with CustomerID, Email, Firstname, Lastname or Gender*	int (CustomerID + Gender) or String
Customer	findByKeyColumn(Key, Value)	find CustomerID	\$Customer.findByKeyColumn(„String column-name“, „String value“)	find CustomerID for customer identified by value for field key	int CustomerID (0 = no matching record found)
Customer	getCustomerDataFromDb()	load all data of current customer from database	\$Customer.getCustomerDataFromDb()	set CustomerID first before calling this method	java.util.Map
Customer	getNewCustomerID(int)	creates new CustomerID	\$Customer.getNewCustomerID(1)	gets new customerID from DB sequence and stores it in customerID	int
Customer	importRequestParameters(requestParameters, null)	updates customer data with request parameters	\$Customer.importRequestParameters(\$requestParameters, null)	imports all customer data available in HTTP request	True / False
Customer	insertNewCust()	insert new customer in database	\$Customer.insertNewCust()	saves current customer data as new customer in the database	int CustomerID
Customer	loadAllListBindings()	get all list bindings of current customer	\$Customer.loadAllBindings()	gets all list binding information of current customer from the database	java.util.Hash table
Customer	loadCustDBStructure()	load customer database structure	\$Customer.loadCustDBStructure()	gets customer data structure (like customer-specific fields) from the database before working with this customer	True / False
Customer	resetCustParameters()	reset internal map for customer parameter	\$Customer.resetCustParameters()	clear all customer data	void
Customer	setCompanyID(int)	set CompanyID	\$Customer.setCompanyID(1)	EMM only	void
Customer	setCustomerID(int)	set CustomerID	\$Customer.setCustomerID(int)		void
Customer	updateInDB()	Write internal map for customer data to database	\$Customer.updateInDB()	Set the following parameters before calling this method: CustomerID, CompanyID (EMM only) and import \$requestParameters	True / False
BindingEntry	get<key>()	Get parameter of current binding entry	\$BindingEntry.get<key>()	Replace <key> with MailinglistID, CustomerID, MediaType**, ExitMailingID***, UserType****, UserStatus***** or UserRemark	int or String (UserStatus + UserRemark)
BindingEntry	set<key>(<value>)	Set parameter of current binding entry	\$BindingEntry.set<key>(int value or “String <value>“)	Replace <key> with MailinglistID, CustomerID, MediaType, ExitMailingID, UserType, UserStatus or UserRemark and replace <value> with integer value or String value (UserType + UserRemark)	void
BindingEntry	getChangeDate()	Get change date of binding entry	\$BindingEntry.getChangeDate()	date when binding entry was changed (i.e. subscription or unsubscription)	java.util.Date
BindingEntry	getUserBindingFromDB(1)	Check, if a customer has a binding to a certain mailinglist	\$BindingEntry.getUserBindingFromDB(1)	Set the following parameters before calling this method: MailinglistID, CustomerID, MediaType, this methods also loads all binding parameters of the customer	True / False
BindingEntry	insertNewBindingInDB(1)	Write a new mailinglist binding for one recipient to database	\$BindingEntry.insertNewBindingInDB(1)	Set the following parameters before calling this method: MailinglistID, CustomerID, UserType, UserStatus, UserRemark, ExitMailingID, MediaType	True / False
BindingEntry	optOutEmailAdr(Email-address, 1)	Set binding of customer to „opt out by admin“	\$BindingEntry.optOutEmailAdr(“String Email-address“, 1)	unsubscribes customer with given email address from current mailinglist	True / False (true: UserStatus = active)
BindingEntry	setChangeDate(date)	Set change date of binding entry	\$BindingEntry.setChangeDate(java.util.Date date)		void

BindingEntry	updateBindingInDB(1)	Update this binding in the database	\$BindingEntry.updateBindingInDB(1)	Set at minimum the following parameters before calling this method: MailinglistID, CustomerID and UserStatus	True / False
BindingEntry	updateStatusInDB(1)	Update the status of a binding in the database	\$BindingEntry.updateStatusInDB(1)	Set the following parameters before calling this method: MailinglistID, CustomerID, UserStatus, UserRemark and ExitMailingID	True / False
MailingDao	getMailing(MailingID, 1)	Get a certain mailing	\$MailingDao.getMailing(int mailingID, 1)		object Mailing
MailingDao	saveMailing(Mailing)	Save a mailing	\$MailingDao.saveMailing(Object mailing)	Request parameter is object Mailing, not mailingID	int MailingID
Mailing	get<key>()	Get parameter of current mailing	\$Mailing.get<key>()	Replace <key> with Id (=MailingID), MailinglistID, MailingType***** or Shortname	int or String (Shortname)
Mailing	set<key>(<value>)	set parameter of current mailing	\$Mailing.set<key>(int <value> or „String <value>“)	Replace <key> with Id (=MailingID), MailinglistID, MailingType or Shortname and replace <value> with integer value or String value (Shortname)	void
Mailing	clone(Context)	Copy current mailing	\$Mailing.clone(ApplicationContext Context)	MailinglistID, CustomerID, UserType, UserStatus, UserRemark, ExitMailingID or MediaType and replace <value> with integer value or String value (UserType + UserRemark)	object Mailing
Mailing	getDynamicTagByld(DynNameID)	Read certain content block of current mailing	\$Mailing.getDynamicTagByld(int DynNameID)	Get mailing with \$MailingDao.getMailing() first	object DynamicTag (can be used like String)
Mailing	getDynTags()	Read all content blocks of current mailing	\$Mailing.getDynTags()	Get mailing with \$MailingDao.getMailing() first	java.util.Map
Mailing	sendEventMailing(CustomerID, DelayMinutes, UserStatus, Overwrite, Context)	Send event mailing	\$Mailing.sendEventMailing(int CustomerID, int DelayMinutes, String UserStatus, [java.util.Map Overwrite null], ApplicationContext Context)	Get mailing with \$MailingDao.getMailing() first, get ApplicationContext with \$ScriptHelper.getApplicationContext()	True / False
ScriptHelper	findLastNewsletter(CustomerID, 1, MailinglistID)	Find latest newsletter that would have been sent to customer	\$ScriptHelper.findLastNewsletter(int CustomerID, 1, int MailinglistID)	The newsletter found by findLastNewsletter can be selected by \$MailingDao.getMailing() and sent by \$Mailing.sendEventMailing (the newsletter also gets a new entry in maildrop_status_tbl so that it can be sent as event-based mail)	MailingID
ScriptHelper	getApplicationContext()	Load object ApplicationContext	\$ScriptHelper.getApplicationContext()		object ApplicationCo ntext
ScriptHelper	newHashMap()	Create new hashmap	\$ScriptHelper.newHashMap()	Does not permit NULL values for keys!	java.util.Map
ScriptHelper	parseTransactionMailXml(XML-input)	Parse XML document into LinkedList object	\$ScriptHelper.parseTransactionMailXml(“String XML-input“)		java.util.Linke dList
ScriptHelper	println(output)	Print output	\$ScriptHelper.println(„String output“)	Writes information to Velocity log	void
ScriptHelper	sendEmail(From-address, To-address, Subject, Body-text, Body-html, Mailtype, Charset)	Send single email	\$ScriptHelper.sendEmail(“String From-address“, “String To-address“, “String Subject“, “String Body-text“, “String Body-html“, int Mailtype, “String Charset“)		True / False
	*Gender: 0=male, 1=female, 2=unknown				
	**MediaType: 0=email				
	***ExitMailingID: set to 0 or to ID of mailing which the recipient unsubscribed from				
	****UserType: W=ordinary recipient, T=test recipient, A=admin recipient				
	*****UserStatus: 1=active, 2=bounced, 3=admin opt-out, 4=user opt-out, 5=waiting for confirmation (double opt-in), 6=blacklisted, 7=status pending				
	*****MailingType: 0=normal mailing, 1=event-driven mailing, 2=time-driven mailing				
	Since we use Velocity as basis for script actions, please see http://velocity.apache.org/engine/releases/velocity-1.4/user-guide.html for further information on syntax and all available Velocity statements like #set(), #if(), #elseif(), #else, #foreach and #end.				
	Note: To initialize the scripting functionality you have to add an action of type „Script Action“ to the form containing Velocity expressions first.				
	The content of this script action must include at least this code at the end to return a „success“ value:				
	#set(\$scriptResult="1")				
	Example to show the name of a mailing in a form:				
Script Action					

#set(\$mail=\$MailingDao.getMailing(815, 1))				
#set(\$scriptResult="1")				
Corresponding Form:				
\$mail.getShortname()				
Example to load all customer data for given email address:				
\$Customer.loadCustDBStructure()				
\$Customer.resetCustParameters()				
\$Customer.setCustomerID(0)				
#set(customerID=\$Customer.findByKeyColumn(„EMAIL“, \$email))				
#set(\$requestParameters = \$ScriptHelper.newHashMap().putAll(\$Customer.getCustomerDataFromDb()))				
#set(\$scriptResult="1")				
Example to send a single mail:				
#set(\$result=\$ScriptHelper.sendEmail(\$requestParameters.SENDER, \$requestParameters.RECIPIENT, \$requestParameters.SUBJECT, \$requestParameters.TEXT, "", 0, "ISO-8859-1"))				
#if(\$result)				
#set(\$scriptResult="1")				
\$ScriptHelper.println("Mail with subject \$requestParameters.SUBJECT was sent to \$requestParameters.RECIPIENT")				
#else				
#set(\$scriptResult="0")				
\$ScriptHelper.println("Mail with subject \$requestParameters.SUBJECT could NOT be sent to \$requestParameters.RECIPIENT")				
#end				
#set(\$scriptResult="1")				
A link triggering the sending of a single mail could look like this (if form „SendMail“ triggers action „SendMail“ consisting of the action script				
http://www.domain.com:8080/form.do?agnCI=1&agnFN=SendMail&SENDER=news@domain.com&RECIPIENT=paul@yahoo.com&SUBJECT=Notification&TEXT=Dear%20Paul..				
Example to send an event-based mailing:				
#set(\$mail=\$MailingDao.getMailing(815,1))				
#if(\$mail)				
\$mail.sendEventMailing(\$customerID.intValue(),0,1, null, \$ScriptHelper.getApplicationContext())				
#end				
#set(\$scriptResult="1")				
Example to send out latest available newsletter:				
#set(\$last=\$ScriptHelper.findLastNewsletter(\$customerID, 1))				
#set(\$mail=\$MailingDao.getMailing(\$last, 1))				
#if(\$mail)				
\$mail.sendEventMailing(\$customerID.intValue(), 0, "1", null, \$ScriptHelper.getApplicationContext())				
#end				
#set(\$scriptResult="1")				