HASHING AND ENCRYPTING EMAIL MANAGER CONTENT

VERSION MANAGEMENT

This document can be retrieved from the author.

VERSION HISTORY

Version	Date	Author	Reason for issue
1.0	02/08/2017	B Clark	First Version
2.0	15/08/2017	B Clark	Final Version
3.0	08/11/2017	B Clark	Added MD5 Hash Functionality

TABLE OF CONTENTS

1 SUMMARY

As we move more deeply into adaptive emails and one to one bespoke content for our customers our clients have a requirement to allow external functions to understand securely who they need to customise the content for. This requirement simply put is to be able to pass data securely from within Email Manager through a link to another source.

This data may be required for tracking activity or passing through an identifier that can allow bespoke content to be displayed once a user clicks through.

2 SIMPLE MD5 HASHING

At times it is useful to simply MD5 hash a variable in email manager.

The MD5 hashing modifier can be applied to any variable used within an Email Manager link or as part of a creative. The example below shows it being used to encode an Email Manager variable. You simply add the below to encode a string

```
{variable}.Base64Encode()
```

This will decode a string

```
{variable}.Base64Decode()
```

3 MORE SECURE SALTED HASHING

We do not recommend that hashing is used alone for security purposes as many of the hash algorithms can relatively easily be guessed with modern processing speeds. It does however provide a data verification method that can add value to automations or as part of a broader security process so has been included. The SHA-2 algorithms are still considered safe from a security point of view particularly if a SALT is used, but if security is the main focus the encryption method mentioned later in the document should be the preferred method.

3.1 SUPPORTED HASHING ALGORITHMS AND ENCODINGS

Hashing Algorithm	MD5	SHA-1	SHA-256	SHA-384	SHA-512
Required Format	MD5	SHA1	SHA256	SHA384	SHA512

Hashing encoding	HEX	BASE64
Required Format	HEX	BASE64

Note: The Hashing encoding determines the encoding for both the output and the salt value if supplied. The output will be an encoded string of the hash bytes created by the chosen algorithm. No URL or

HTML encoding is automatically applied to this encoding on output. The existing modifiers can be used if needed. These are:

URLENCODE - {var}.urlencode

URLDECODE - {var}.urldecode

HTMLENCODE - {var}.htmlencode

HTMLDECODE - {var}.htmldecode

4 HOW TO HASH TO A VARIABLE

The hashing modifier can be applied to any variable used within an Email Manager link or as part of a creative. The example below shows it being used to encode an Email Manager variable. You simply add the

.hash

and the chosen algorithm and encoding type and salt after. If the Salt is not required it can be left out but the empty brackets are still required.

```
{variable}.hash([algorithm][encoding][salt])
```

4.1.1 {VARIABLE}

This is the actual content to be hashed, it has to be driven by a variable. In the example below the variable “recipientID” will be used and hashed for each recipient.

```
{recipientID}.hash([algorithm][encoding][salt])
```

4.1.2 [ALGORITHM]

This is your selected algorithm; the table in 2.1 shows you the options available. The higher values provide higher levels of encryption but will require more performance to create the hashes and the hash values are longer. If added to a URL you should be certain that it does not go beyond the required standards for a URL.

```
{recipientID}.hash([SHA512][encoding][salt])
```

4.1.3 [ENCODING]

This provides the encoding type for the salt value being passed. Two are supported HEX and Base64

```
{recipientID}.hash([SHA512][HEX][salt])
```

4.1.4 [SALT]

We can further complicate the hashes by appending a string, called a salt, to the value before hashing. This could make the same hash into a completely different string every time. The Salt can be a variable different for each recipient or a single salt added in send a message which would be less secure. It should be noted that the salt value passed in must be pre-encoded into the appropriate format. In this case Hex was used.

```
{recipientID}.hash([SHA512][HEX][{SALT}])
```

4.1.5 URL ENCODING THE RESULT

```
{recipientID}.hash([SHA512][HEX][{SALT}]).urlencode
```

Note: Using a SALT, increasing the length of the SALT and using a unique SALT for each recipient makes the chances of a Hashed value being discovered much less likely. It also significantly increases the effort required to crack the other hashes and therefore would put off further cracking attempts. We would suggest a SALT as many characters long as the output is used where possible.

5 ENCRYPTION

As mentioned under the hashing section the security of a hashed value is not always considered acceptable for sensitive data. We have includes an encryption funtion that works with RSA private and public keys that will meet this requirement. The private key is only known to the customer. They add the public key to the function. This means that only the customer can decrypt the data. As with the hashing this is a function that can be used for Variables.

6 HOW TO ENCRYPT A VARIABLE

The encryption modifier can be applied to any variable used within an Email Manager link or as part of a creative. The example below shows it being used to encode an Email Manager variable. You simply add the

.encrypt

and the chosen Public Key.

```
{variable}.encrypt([publicKey])
```

6.1.1 {VARIABLE}

This is the actual content to be encrypted, it should be driven by a variable. In the example below the variable "recipientID" will be used and encrypted for each recipient.

```
{recipientID}.encrypt([publicKey])
```

6.1.2 [PUBLICKEY]

This is where you add the PEM encoded public key to be used to encrypt the data. The public key can be added directly into the code in creative builder “source view”. “Design view” may automatically add characters that cause the encoding to fail so should not be used.

As it is a public key it does not matter if it is freely available. You can add the full key including the Begin and End as this is stripped out automatically and including it reduces the chances of cut and paste errors.

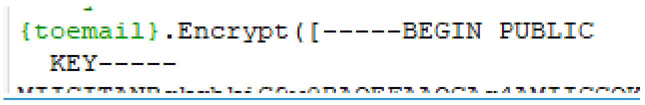
```
{recipientID}.encrypt([-----BEGIN PUBLIC KEY-----
MIGeMA0GCSqGSIb3DQEBAQUAA4GMADCBiAKBgGQ5N6VY7/NauDp5Ro5R8VbU/ele
QJA7SsWeg7iBYft/NHEilmo5fkXF7HbgI/XYzHaa82+CbjnCKiHvJBuecFFq7Un+
JRTMNhVTSF4cxg/QcnX1s0mIkV4RGyCPIlIBeAplKAMy7aI0KVTC5ktMkkOCUyIU
4GZVWzwEyUhN8WcJAgMBAAE=
-----END PUBLIC KEY-----])
```

As you can see adding the Public key directly into the code increases the length of the modifier significantly and if this is used in multiple locations may make it difficult to maintain the code.

It may be more convenient to create a shared variable with the public key. This can be shared across creatives, made default and hidden from users to reduce the risk of a typo changing the key.

```
{recipientID}.encrypt([{publicKey}])
```

Please note that the length of the 4096 bit key can cause issues when pasted into source view. We would recommend checking the first header line following first save to ensure the line has not broken at within the header line. This may cause the encoding to fail. Creating a new key and using this may resolve the issue. Example below.



```
{toemail}.Encrypt([-----BEGIN PUBLIC
  KEY-----
```

Note: It may be that customers would like to use more than one public key for different campaigns. You could create more than one shared variable and simply map the required one each time. {publicKey1}, {publicKey2} etc

Alterian will not have the private key so anything encrypted by our customers cannot be decrypted by Alterian.

7 VALIDATION AND FEEDBACK

Please note that the initial release of the hashing and encryption functionality does not include validation on the setup of the modifier or the values used to drive it. Please follow the syntax as accurately as possible and test thoroughly before going live to be sure your setup is valid.

If the syntax is not added correctly or you attempt to add values in the incorrect format the deployment will still be attempted. In most cases the deployment will fail with mailer errors for all emails visible in Deployment manager or quick totals.

8 PERFORMANCE

Each time email manager hashes or encrypts a variable it requires greater effort from our servers. This is particularly true of the higher bit rate hashes and encryption. When our servers are to do more, they will work through your emails more slowly. This may mean a decrease in overall email output from Email Manager.

We would suggest that you put hashing or encryption in place only where it is required from a security perspective. We would also suggest significant testing before going live with any project including this functionality to ensure performance still meets your business requirements.

Testing is also vital to ensure the changing nature of your links due to hashing or encryption does not cause issues.