Traffic Management Center Authority (TMCA) Users Guide

Version 1.2

May 14, 2019

INTEGRITY Security Services LLC
a Green Hills Software Company

TABLE OF CONTENTS

TABLE OF CONTENTS 2
1 INTRODUCTION 4
2 SCOPE 4
2.1 What's Covered 4
3 DEFINITIONS AND ACRONYMS 5
3.1 Definitions 5
3.2 Acronyms 5
4 SYSTEM OVERVIEW 6
5 INTERFACES 7
5.1 Command Shells 7
5.1.1 Itshell 7
5.1.1.1 Shutting Down an HA Pair 7
5.1.1.2 Powering Up an HA Pair 8
5.2 Web User Interface 8
5.2.1 TMCA Web User Interface Dashboard 9
5.2.1.1 TMCA Request Enrollment 9
5.2.1.2 TMCA Process Enrollment 10
5.2.1.3 TMCA Manage Users 11
5.2.1.3.1 Adding a new TMCA User 11
5.2.1.3.2 Removing a User from the TMC Authority 12
5.2.1.3.3 Updating a User on the TMC Authority 12
5.2.1.3.4 Changing User Password 12
5.2.1.4 TMCA Update SW 13
5.3 TMC Authority REST Interface 13
5.3.1 The /tmc/signmap route 14
5.3.1.1 Input 14
5.3.1.2 Response out – Success case 14
5.3.2 The /tmc/signtim route 16
5.3.2.1 Input 16
5.3.2.2 Response out – Success case 16
5.3.3 The /tmc/verifySignature route 18
5.3.3.1 Input 18
5.3.3.2 Response out – Success case 18
5.3.4 The /tmc/encryptMessage route 19
5.3.4.1 Input 19
5.3.4.2 Response out – Success case 19
5.3.5 The /tmc/decryptMessage route 20
5.3.5.1 Input 20
5.3.5.2 Response out – Success case 20
6 DEPLOYMENT CONSIDERATIONS 21
7 VERSION HISTORY 22

1 INTRODUCTION

This document provides an overview of the DLM Traffic Management Center Authority (TMCA).

2 SCOPE

2.1 What's Covered

This document will describe the following:

- An overview of what the TMC Authority is and
- A detailed overview of the interfaces for the system.
- REST service API's for signing, verifying, encrypting and decrypting messages.

3 DEFINITIONS AND ACRONYMS

3.1 Definitions

Certificate	A digitally signed data set identifying the holder of a Private Key corresponding to the Public Key contained in the Certificate
Key Pair	For asymmetric key cryptography, a Private Key and its corresponding Public Key
Private Key	A secret key known only to owner of the Key Pair. This is the key of a Key Pair that is used by the Holder to create digital signatures.
Public Key	This key is mathematically related to the owner's Private Key. It may be publicly disclosed by the owner, and is used to verify digital signatures created by the owner of the Private Key
Signature	A cryptographic data set created to demonstrate the authenticity of a digital message or documents

3.2 Acronyms

CA	Certificate Authority or Certification Authority
CMS	ISS Certificate Management Service
CSR	Certificate Signing Request
DLM	Device Lifecycle Management
HA	High Availability
HSM	Hardware Security Module
ISS	INTEGRITY Security Services LLC.
NAS	Network Attached Storage
TLS	Transport Layer Security
TMC	Traffic Management Center
PKI	Public Key Infrastructure

4 SYSTEM OVERVIEW

The Device Lifecycle Management (DLM) Traffic Management Center Authority (TMCA) is a product provided by INTEGRITY Security Services (ISS). The TMCA is intended to be installed in traffic management centers and supports signing and verification of V2X messaging between vehicles, roadside equipment, and the traffic management center itself.

The TMC Authority secures data inside and out with its FIPS 140-2 Level 3 protection of keys and TLS v1.2 tunnels to other traffic management center servers.

This manual provides an overview of the setup and administration of the TMC Authority.

5 INTERFACES

This section explains the various interfaces that are provided by the TMC Authority.

5.1 Command Shells

The command shells provide a means to configure or execute box level functionality on an individual TMCA server. These shells are not intended for general user interaction. They are intended for IT personnel when configuration or troubleshooting is required.

5.1.1 Itshell

For IT level configuration and troubleshooting of the servers an itshell interface is provided. Itshell can be used for establishing the server network configuration, viewing log files, management of high availability features, and restarting services on the servers.

To connect to itshell, simply ssh to the itshell user on the desired server. For example, to connect to itshell on the tmc.example.com TMCA you would use the command:

ssh itshell@tmc.example.com

At the prompt, type help to get a listing of help topics:

```
Enter a command: help

Documented commands (type help <topic>):
========================================
chkconfig              passwd    set_failover    tail_pglog          tzselect
copy_pgdata_from_peer  ping      set_ha_network  tail_syslog
delete                 poweroff  show            tail_webuicust
edit                   reboot    start_all       test_tcp_connection
net_restart            service   stop_all        tracepath

Undocumented commands:
======================
EOF  exit  help  quit

Enter a command:
```

5.1.1.1 Shutting Down an HA Pair

High availability server pairs must be shut down in a particular manner to avoid an unintended failover to happen. To shut down an HA pair via itshell, perform the following actions:

- SSH to itshell on the standby unit.
- Enter the command:
  - chkconfig heartbeat off
- Enter the command:

- service heartbeat stop
- Wait for “Stopping High-Availability services: Done.” and press enter.
- Enter a command:
  - poweroff
- SSH to itshell on the primary unit.
- Enter a command:
  - poweroff

5.1.1.2 Powering Up an HA Pair

After power is restored to the servers, HA failover mode must be re-enabled. To re-enable HA services perform the following actions:

- SSH to itshell on the primary unit
- Enter a command:
  - Show ip
  - Continue to call this command until you see that “eth0:0” is listed as a viable network interface (This is the Virtual IP and must be up and active before the standby heartbeat is turned on).
- SSH to itshell on the standby unit
- Enter a command:
  - chkconfig heartbeat on
- Enter a command:
  - service heartbeat start
  - Wait for “Starting High-Availability services: INFO: Resource is stopped Done” and press enter.

5.2 Web User Interface

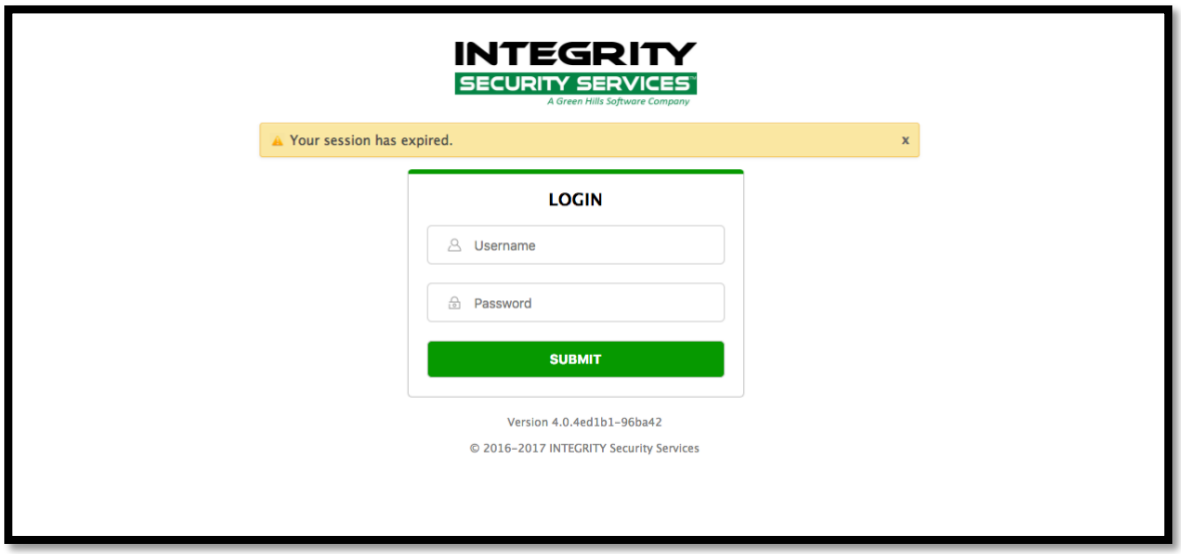
The TMCA servers have a Web user interface for manual management of their functions.

The TMCA user interface provides the following functions:

- User management, for assigning users.
- Producing enrollment request packages to enroll the TMCA with the ISS CMS.
- Processing the enrollment response from the ISS CMS.
- Applying software updates to the system.

When accessing the WebUI for an appliance, that appliance’s virtual ip should be used. This will ensure that the user is directed to the appropriate server in the HA pair.

The initial page for the appliance will be the login page. Log in with the credentials that you have been provided with to proceed.



5.2.1 TMCA Web User Interface Dashboard

After logging into the TMC Authority, the TMCA dashboard will be presented. At the top of the page is the TMCA Control Panel.

In the control panel, the following options are presented:

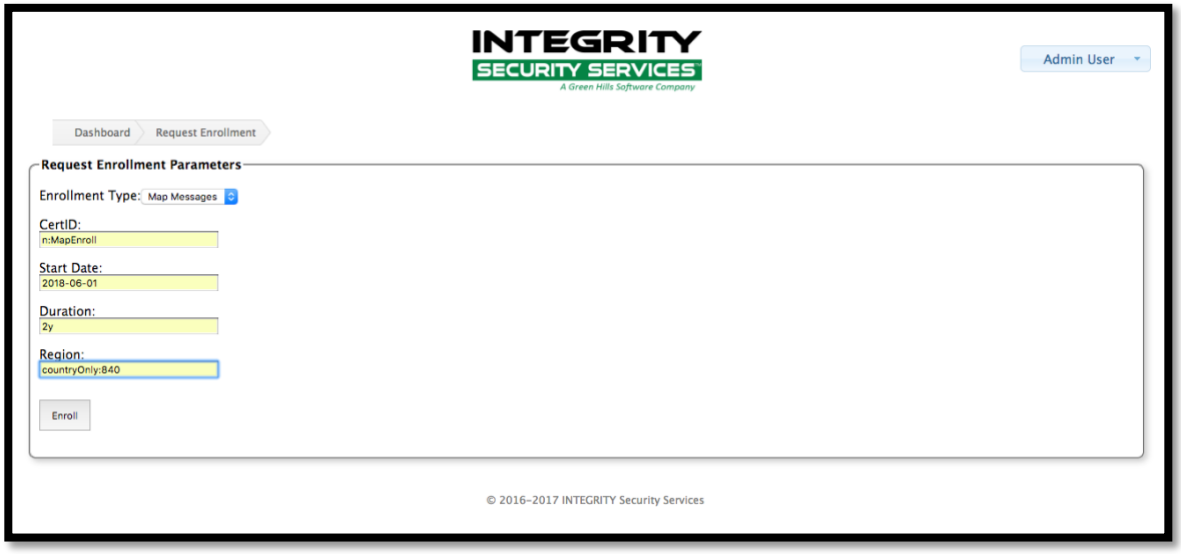
- Manage Users – For managing user accounts for the TMC appliance.
- Request Enrollment – For creating a CMS enrollment request for the TMCA.
- Process Enrollment – For processing the CMS enrolment response package.
- Update Software – For applying software updates.

Each of these operations are covered in detail below.

5.2.1.1 TMCA Request Enrollment

The "Request Enrollment" button on the TMCA control panel is where an enrollment request for enrolling with the ISS CMS can be generated. Enrollment of the TMCA with

the ISS CMS system is typically a one time event. Once a system is enrolled it will automatically contact the CMS servers to update its certificates on a periodic basis.



After filling in the enrollment request form, an SCMS enrollment request file will be generated. The generated file is automatically downloaded by your browser to your "Downloads" folder on your computer.

This file must be provided to the ISS CMS administrator to enroll the TMC Appliance. An enrollment response file will be returned to you. The enrollment response must be submitted to the TMC Authority via the "Process enrollment" button on the TMCA control panel.

5.2.1.2 TMCA Process Enrollment

The "Process Enrollment" button on the TMCA control panel is where an enrollment response from the ISS CMS is processed. The enrollment response is provided as a result of an enrollment request.

The enrollment response is provided as a zip package that is uploaded and processed by the TMC Authority. Once enrollment has completed successfully, the TMC Authority can be used to sign and verify messages.

To process the enrollment response package, first select the enrollment response file with the "Choose File" button. After the file has been selected, press the "Process Enrollment" button to complete the enrollment. A success or error status will be displayed to indicate if the enrollment response package was processed successfully.

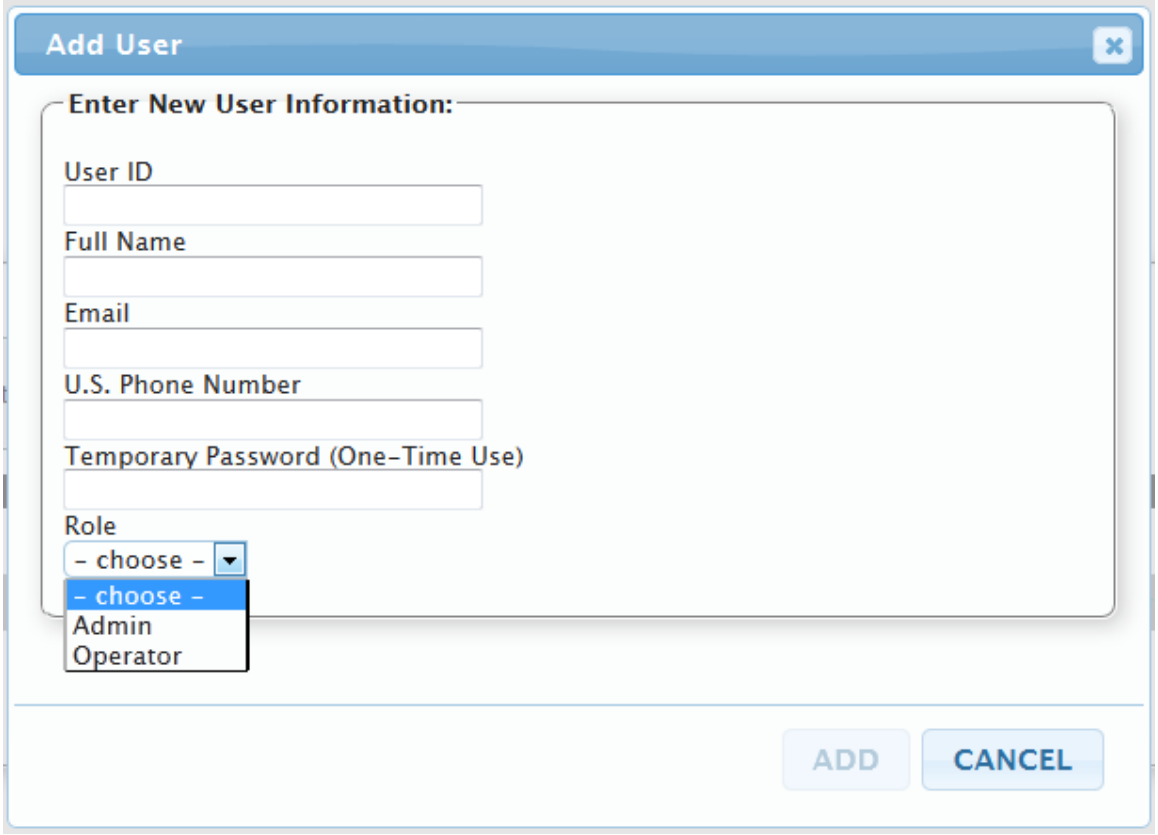
NOTE: If the enrollment package is processed successfully, the user will also be automatically logged out while internal services re-start. After the internal services re-start, the TMCA will automatically contact the ISS CMS to obtain application certificates for the enrolled signing services.

5.2.1.3 TMCA Manage Users

The "Manage Users" button on the TMCA control panel opens the user management page. This is where users of the TMCA can be created, updated, and deactivated. The user management page is presented as follows:

5.2.1.3.1 Adding a new TMCA User

The "Add User" button allows a new TMCA user to be created.



To add a new user to the TMC Authority, provide a unique user ID, and fill out the information for the new user. A role for the new user must also be selected.

Once the new user has been configured press the “Add” button.

5.2.1.3.2 Removing a User from the TMC Authority

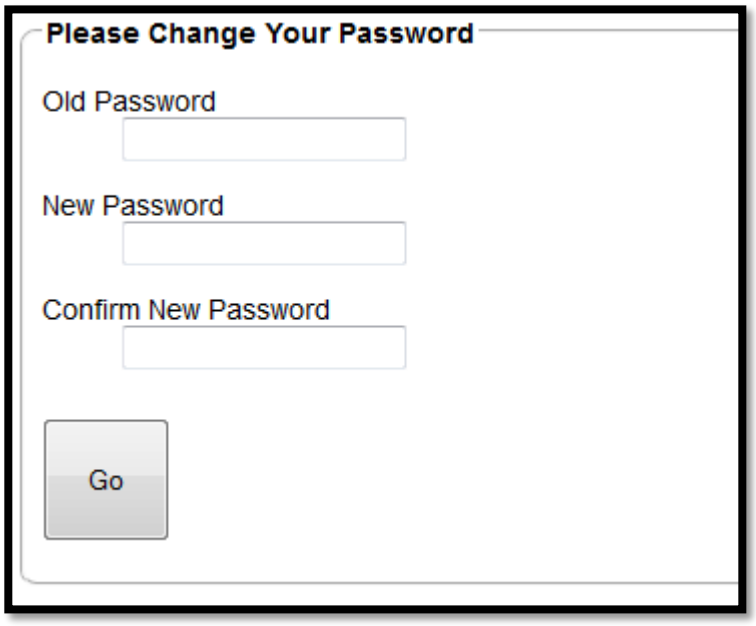
To remove a user from the TMCA, simply select the user and press the “Delete User” button. Upon confirming the deletion, the user will be removed from the system.

5.2.1.3.3 Updating a User on the TMC Authority

To update a user's information, lock the user's account, or reset a user's password; select the user from the list and press the “Update User” button. Simply update the user information and press the “Update” button.

5.2.1.3.4 Changing User Password

A user logged into the TMCA can change their own password. This action is performed by pressing the “Change My Password” button. The change password page is presented as follows:



Once the fields have been filled in press the "Go" button to proceed.

5.2.1.4 TMCA Update SW

The "Update SW" button on the TMCA control panel opens the software update page. This page typically allows software updates to be applied by the customer.

5.3 TMC Authority REST Interface

The TMC Authority has a REST interface on port 55443 that provides message signing and verification services for TMC-created MAP and TIM messages. Note that the Authority must be enrolled with the ISS CMS for the corresponding signing method prior to signing.

The following routes are provided by the REST service:

Route	Description
/tmc/signmap	For signing "map" messages.
/tmc/signtim	For signing "tim" messages.
/tmc/verifySignature	For verifying signed messages.
/tmc/encryptMessage	For encrypting messages

/tmc/decryptMessage	For decrypting messages

5.3.1 The /tmc/signmap route

This service signs “map” messages.

```
POST /tmc/signmap
```

5.3.1.1 Input

Input field	Required	Parameter name	Parameter Type (max length)	Encoding	Note
Unsigned message	Yes	message	Binary (< 1998 bytes)	Base64	The unsigned message must be Base64 encoded.
Validity override	No	sigValidityOverride	Number		If a custom validity period is desired for the signature it can be specified with this optional parameter. If the parameter is present and > 0 the custom validity time is in milliseconds. If the parameter is 0 or omitted, the default value is used.

Request Example

```
{
    “message”:“ BQUFBQUFBQUFBQUFBQUFBQUFBQU=”
}
```

Or with a custom validity period specified:

```
{
    “message”:“ BQUFBQUFBQUFBQUFBQUFBQUFBQU=”,
    “sigValidityOverride”: 30000
}
```

5.3.1.2 Response out – Success case

Upon success, the signed message is returned in Base64 encoded format.

Output field	Sent by the service	Parameter name	Parameter Type (max length)	Encoding	Note
Signed message	Yes	message-signed	Binary (2048 bytes)	Base64	The signed Base64 encoded message

Success Response Example

```
{
```

"message-signed":"
A4EAQAOAFAUFBQUFBQUFBQUFBQUFBQUFYAMgQJcAAbBpT2eQ5gABsGol+zTmgQEBAAMBgBY
xr7X8JV0PUIIILyGNk71VDhpeb1sAAxxSnNCEAKmDAQGAA0gBAQADIECXgYP/EuTPatGhcBTmGf
rO0pNohD1foK/0bdndtyytgzi2iYCCeT+MZjQxVJfxRQvAThdHPjPR+uioAMkKRZfjG2LIZh7Z6
TtH8pzOkzX6Doj2e0aC8+6btFPYkQ/d2DhY9EMwAw=="
}

5.3.2 The /tmc/signtim route

This service signs "tim" messages.

```
POST /tmc/signtim
```

5.3.2.1 Input

Input field	Required	Parameter name	Parameter Type (max length)	Encoding	Note
Unsigned message	Yes	message	Binary (< 1998 bytes)	Base64	The unsigned message must be Base64 encoded.
Validity override	No	sigValidityOverride	Number		If a custom validity period is desired for the signature it can be specified with this optional parameter. If the parameter is present and > 0 the custom validity time is in milliseconds. If the parameter is 0 or omitted, the default value is used.

Request Example

```
{
    "message":"  BQUFBQUFBQUFBQUFBQUFBQUFBQU="
}
```

Or with a custom validity period specified:

```
{
    "message":"  BQUFBQUFBQUFBQUFBQUFBQUFBQU=",
    "sigValidityOverride": 30000
}
```

5.3.2.2 Response out – Success case

Upon success, the signed message is returned in Base64 encoded format.

Output field	Sent by the service	Parameter name	Parameter Type (max length)	Encoding	Note
Signed message	Yes	message-signed	Binary (2048 bytes)	Base64	The signed Base64 encoded message

Success Response Example

```
{
    "message-signed":"
A4EAQAOAFAUFBQUFBQUFBQUFBQUFBQUFYAMgQJcAAbBpT2eQ5gABsGol+zTmgQEBAAMBgBY
xr7X8JV0PUIIILyGNk71VDhpeb1sAAxxSnNCEAKmDAQGAA0gBAQADIECXgYP/EuTPatGhcBTmGf
```

rO0pNohD1foK/0bdndtyytgzi2iYCCeT+MZjQxVJfxRQvAThdHPjPR+uioAMkKRZfjG2LIZh7Z6
TtH8pzOkzX6Doj2e0aC8+6btFPYkQ/d2DhY9EMwAw=="
}

5.3.3 The /tmc/verifySignature route

This service verifies signed messages.

```
POST /tmc/verifySignature
```

5.3.3.1 Input

Input field	Required	Parameter name	Parameter Type (max length)	Encoding	Note
Unsigned message	Yes	Message	Binary (< 1998 bytes)	Base64	The unsigned message must be Base64 encoded.

Request Example

```
{
    “message”:“
A4EAQAOAFAUFBQUFBQUFBQUFBQUFBQUFYAMgQJcAAbBpT2eQ5gABsGol+zTmgQEBAAMBgBY
xr7X8JV0PUIIILyGNk71VDhpeb1sAAxxSnNCEAKmDAQGAA0gBAQADIECXgYP/EuTPatGhcBTmGf
rO0pNohD1foK/0bdndtyytgzi2iYCCeT+MZjQxVJfxRQvAThdHPjPR+uioAMkKRZfjG2LIZh7Z6
TtH8pzOkzX6Doj2e0aC8+6btFPYkQ/d2DhY9EMwAw==”
}
```

5.3.3.2 Response out – Success case

Upon success, the signed message is returned in Base64 encoded format.

Output field	Sent by the service	Parameter name	Parameter Type (max length)	Encoding	Note
Verification status	Yes	message-verified	Boolean		True if the message was verified, false otherwise.

Success Response Example

```
{
    "message-verified":true
}
```

5.3.4 The /tmc/encryptMessage route

This service encrypts messages. The message can be either an arbitrary message to be encrypted or an SPDU to be encrypted. For an arbitrary message the isSPDU parameter must be set to 0. If the message is an SPDU then the isSPDU parameter must be set to 1.

```
POST /tmc/encryptMessage
```

5.3.4.1 Input

Input field	Required	Parameter name	Parameter Type (max length)	Encoding	Note
Is SPDU	Yes	isSPDU	Numeric (0 or 1)	Number	Set as 1 if the message is an SPDU, or 0 to encrypt an arbitrary message.
Plaintext message	Yes	message	Binary (< 1998 bytes)	Base64	The plaintext message must be Base64 encoded.

Request Example

```
{
    “isSPDU”: 0,
    “message”:“ BQUFBQUFBQUFBQUFBQUFBQUFBQU=”
}
```

5.3.4.2 Response out – Success case

Upon success, the encrypted message is returned in Base64 encoded format.

Output field	Sent by the service	Parameter name	Parameter Type (max length)	Encoding	Note
Encrypted message	Yes	message-encrypted	Binary (2048 bytes)	Base64	The encrypted Base64 encoded message

Success Response Example

```
{
    "message-encrypted":"
A4EAQAOAFAUFBQUFBQUFBQUFBQUFBQUFYAMgQJcAAbBpT2eQ5gABsGol+zTmgQEBAAMBgBY
xr7X8JV0PUIIILyGNk71VDhpeb1sAAxxSnNCEAKmDAQGAA0gBAQADIECXgYP/EuTPatGhcBTmGf
rO0pNohD1foK/0bdndtyytgzi2iYCCeT+MZjQxVJfxRQvAThdHPjPR+uioAMkKRZfjG2LIZh7Z6
TtH8pzOkzX6Doj2e0aC8+6btFPYkQ/d2DhY9EMwAw=="
}
```

5.3.5 The /tmc/decryptMessage route

This service decrypts messages.

`POST /tmc/decryptMessage`

5.3.5.1 Input

Input field	Required	Parameter name	Parameter Type (max length)	Encoding	Note
Encrypted message	Yes	message	Binary (< 1998 bytes)	Base64	The encrypted message must be Base64 encoded.

Request Example

```
{
    “message”:“ BQUFBQUFBQUFBQUFBQUFBQU=”
}
```

5.3.5.2 Response out – Success case

Upon success, the decrypted message is returned in Base64 encoded format.

Output field	Sent by the service	Parameter name	Parameter Type (max length)	Encoding	Note
Decrypted message	Yes	message-decrypted	Binary (2048 bytes)	Base64	The decrypted Base64 encoded message

Success Response Example

```
{
    "message-decrypted":"
A4EAQAOAFAUFBQUFBQUFBQUFBQUFBQUFYAMgQJcAAbBpT2eQ5gABsGol+zTmgQEBAAMBgBY
xr7X8JV0PUIIILyGNk71VDhpeb1sAAxxSnNCEAKmDAQGAA0gBAQADIECXgYP/EuTPatGhcBTmGf
rO0pNohD1foK/0bdndtyytgzi2iYCCeT+MZjQxVJfxRQvAThdHPjPR+uioAMkKRZfjG2LIZh7Z6
TtH8pzOkzX6Doj2e0aC8+6btFPYkQ/d2DhY9EMwAw=="
}
```

6 DEPLOYMENT CONSIDERATIONS

This section provides a brief overview of service ports that are required for the TMCA's operation.

Source	Destination	Ports	Protocol
External	DC-master, DC-standby	55443	TCP
DC-Master	DC-Standby	1095, 5432, 55443, 19090, 9997, 22	TCP
DC-Master	DC-Standby	694	UDP
Jump server and internal admin	DC-master, DC-Standby	22, 443, 8000	TCP

7 VERSION HISTORY

Version	Date	Comments
1.0	15-Apr-2018	Initial release of this guide
1.0a	16-Jan-2019	Added firewall information
1.1	24-Jan-2019	Added REST service details, and deployment considerations.