

Trustonic  
MobiCore<sup>®</sup>

## **MobiCore Validation Test Suite**

Version 1.2.0.0 Build manually



# Contents

<b>1</b>	<b>MobiCore Validation Test Suite</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>7</b>
2.1	Modules	7
<b>3</b>	<b>Data Structure Index</b>	<b>9</b>
3.1	Data Structures	9
<b>4</b>	<b>File Index</b>	<b>11</b>
4.1	File List	11
<b>5</b>	<b>Module Documentation</b>	<b>13</b>
5.1	Test Subjects	13
5.1.1	Detailed Description	13
5.2	Performance	14
5.3	Inter-world communication functions [COM].	14
5.4	Security support functions [SEC].	14
5.5	MobiCore system functions [SYS].	14
5.6	Mobicore Inter Process Call Interface	14
5.7	Precondition setup	14
5.7.1	Detailed Description	14
5.8	Standard session opened	14
5.9	MemoryManagement	14
5.9.1	Detailed Description	15
5.10	00100_mcMapBoundaryTests	15
5.10.1	Detailed Description	16
5.10.2	Function Documentation	17
5.10.2.1	TEST_SCENARIO	17
5.10.2.2	TestCase_GC00100.CheckBufferBoundariesSize1	17
5.10.2.3	TestCase_GC00101.CheckBufferBoundariesSize1Page	18
5.10.2.4	TestCase_GC00102.CheckBufferBoundariesSize2Pages	18
5.10.2.5	TestCase_GC00103.CheckBufferBoundariesSize3Pages	19
5.10.2.6	TestCase_GC00104.CheckBufferBoundariesSize4Pages	19
5.10.2.7	TestCase_GC00105.CheckBufferBoundariesSize8Pages	20
5.10.2.8	TestCase_GC00106.CheckBufferFillAndReadbackSize1	20
5.10.2.9	TestCase_GC00107.CheckBufferFillAndReadbackSize1Page	21
5.10.2.10	TestCase_GC00108.CheckBufferFillAndReadbackSize2Pages	22
5.10.2.11	TestCase_GC00109.CheckBufferFillAndReadbackSize3Pages	22
5.10.2.12	TestCase_GC00110.CheckBufferFillAndReadbackSize4Pages	23
5.10.2.13	TestCase_GC00111.CheckBufferFillAndReadbackSize8Pages	23

5.10.2.14	TestCase_GC00112_MapAllAndReadbackSize1	24
5.10.2.15	TestCase_GC00113_MapAllAndReadbackSize1Page	24
5.10.2.16	TestCase_GC00114_MapAllAndReadbackSize2Pages	25
5.10.2.17	TestCase_GC00115_MapAllAndReadbackSize3Pages	25
5.10.2.18	TestCase_GC00116_MapAllAndReadbackSize4Pages	26
5.10.2.19	TestCase_GC00117_MapAllAndReadbackSize8Pages	27
5.10.2.20	TestCase_GC00118_MapAllAndReadbackSize255Pages	27
5.10.2.21	TestCase_GC00119_MapDoesntClear	28
5.10.2.22	TestCase_BC00120_TrustletPagefaultOnWrongAddress	29
5.11	Example Sub Group - TCI Buffer Tests	29
5.12	McDrvApi	29
5.12.1	Detailed Description	30
5.13	00100.mcOpenDevice	30
5.13.1	Detailed Description	31
5.13.2	Function Documentation	32
5.13.2.1	TestCase_GC00100_IdDefault	32
5.13.2.2	TestCase_BC00101_InvalidDevice1	32
5.13.2.3	TestCase_BC00102_InvalidDevice2	33
5.13.2.4	TestCase_BC00103_InvalidDeviceMinus1	33
5.13.2.5	TestCase_BC00104_OpenRepeatedly	34
5.14	00200.mcCloseDevice	34
5.14.1	Detailed Description	34
5.14.2	Function Documentation	35
5.14.2.1	TestCase_GC00100_IdDefault	35
5.14.2.2	TestCase_BC00101_InvalidDevice1	36
5.14.2.3	TestCase_BC00102_InvalidDevice2	36
5.14.2.4	TestCase_BC00103_InvalidDeviceMinus1	37
5.14.2.5	TestCase_BC00103_CloseRepeatedly	37
5.15	00300.mcMallocWsm	38
5.15.1	Detailed Description	38
5.15.2	Function Documentation	39
5.15.2.1	TestCase_GC00100_IdDefault4k	39
5.15.2.2	TestCase_BC00101_InvalidDevice1	39
5.15.2.3	TestCase_BC00101_InvalidDevice4	40
5.15.2.4	TestCase_BC00102_InvalidDeviceMinus1	40
5.15.2.5	TestCase_BC00103_Len0	41
5.15.2.6	TestCase_BC00104_LenMinus1	41
5.15.2.7	TestCase_BC00105_WsmNullPointer	42
5.15.2.8	TestCase_BC00106_ClosedDevice	42
5.15.2.9	TestCase_GC00107_StresstestMultipleAllocations	43
5.15.2.10	TestCase_GC00108_StresstestMultipleSizes	43
5.16	00400.mcFreeWsm	44
5.16.1	Detailed Description	44
5.16.2	Function Documentation	45
5.16.2.1	TestCase_GC00100_DeviceDefault	45
5.16.2.2	TestCase_BC00101_DeviceInvalid	45
5.16.2.3	TestCase_BC00102_WsmNullPointer	46
5.16.2.4	TestCase_BC00103_MemoryNotManagedByLib	46
5.16.2.5	TestCase_BC00104_OnClosedDevice	47
5.17	00500.mcOpenSession	47

5.17.1	Detailed Description	48
5.17.2	Function Documentation	49
5.17.2.1	TestCase_GC00100_05010000000000000000000000000000	49
5.17.2.2	TestCase_BC00101_SessionNullPointer	49
5.17.2.3	TestCase_BC00102_UuidNullPointer	50
5.17.2.4	TestCase_BC00103_TciNullPointer	50
5.17.2.5	TestCase_BC00104_LenBiggerMax	51
5.17.2.6	TestCase_BC00105_LenMinus1	51
5.17.2.7	TestCase_BC00106_Len0	52
5.17.2.8	TestCase_BC00107_SessionhandleInvalid	52
5.17.2.9	TestCase_BC00108_UuidUnknown1234123412341234	53
5.17.2.10	TestCase_GC00109_MaximumSessions	53
5.17.2.11	TestCase_BC00110_CloseDeviceWithOpenSession	54
5.17.2.12	TestCase_BC00111_LeaveSession	54
5.18	00600.mcCloseSession	55
5.18.1	Detailed Description	55
5.18.2	Function Documentation	56
5.18.2.1	TestCase_GC00100	56
5.18.2.2	TestCase_BC00101_SessionNullPointer	56
5.18.2.3	TestCase_BC00102_SessionInvalidId	57
5.18.2.4	TestCase_BC00103_SessionInvalidDevice	57
5.18.2.5	TestCase_BC00104_CloseMultipleTimes	58
5.19	00700.mcNotify	58
5.19.1	Detailed Description	59
5.19.2	Function Documentation	60
5.19.2.1	TestCase_GC00100_notifyOnce	60
5.19.2.2	TestCase_BC00101_SessionhandleNullPointer	60
5.19.2.3	TestCase_BC00102_SessionhandleInvalidId	61
5.19.2.4	TestCase_BC00103_SessionhandleInvalidDevice	61
5.20	00800.mcWaitNotification	62
5.20.1	Detailed Description	62
5.20.2	Function Documentation	63
5.20.2.1	TestCase_GC00100_NotifyThenTimeoutInfinite	63
5.20.2.2	TestCase_GC00101_NotifyThenTimeoutFinite	63
5.20.2.3	TestCase_GC00102_NoNotifyThenTimeoutFinite	64
5.20.2.4	TestCase_BC00103_NoNotifyThenThenNoTimeout	65
5.20.2.5	TestCase_BC00104_SessionhandleNullPointer	65
5.20.2.6	TestCase_BC00105_SessionhandleInvalidId	66
5.20.2.7	TestCase_BC00106_SessionhandleInvalidDevice	66
5.21	00900.mcGetSessionErrorCode	67
5.21.1	Detailed Description	67
5.21.2	Function Documentation	68
5.21.2.1	TestCase_GC00050_NoNotifications	68
5.21.2.2	TestCase_GC00100_MultipleNotifications	68
5.21.2.3	TestCase_GC00101_TrustletExits	69
5.21.2.4	TestCase_GC00103_TrustletCrashes	70
5.21.2.5	TestCase_BC00104_SessionhandleNullPointer	70
5.21.2.6	TestCase_BC00105_LastErrNullPointer	71
5.21.2.7	TestCase_BC00106_SessionhandleInvalidId	71
5.21.2.8	TestCase_BC00107_SessionhandleInvalidDevice	72

5.22	01000.mcMap	72
5.22.1	Detailed Description	72
5.22.2	Function Documentation	73
5.22.2.1	TestCase_GC00100_Map4k	73
5.22.2.2	TestCase_BC00101_SessionhandleNullPointer	74
5.22.2.3	TestCase_BC00102_MapinfoNullPointer	74
5.22.2.4	TestCase_BC00103_SessionhandleInvalidId	75
5.22.2.5	TestCase_BC00104_SessionhandleInvalidDevice	75
5.22.2.6	TestCase_BC00105_MultipleMappingsOfSameBuffer	76
5.22.2.7	TestCase_BC00106_MapAReadOnlyBuffer	76
5.23	01100.mcUnmap	77
5.23.1	Detailed Description	77
5.23.2	Function Documentation	78
5.23.2.1	TestCase_GC00100_Unmap4k	78
5.23.2.2	TestCase_BC00101_SessionhandleNullPointer	78
5.23.2.3	TestCase_BC00102_BufNullPointer	79
5.23.2.4	TestCase_BC00103_MapinfoNullPointer	79
5.23.2.5	TestCase_BC00104_SessionhandleInvalidId	80
5.23.2.6	TestCase_BC00105_SessionhandleInvalidDevice	80
5.23.2.7	TestCase_BC00106_BufferPointerNotManagedByLib	81
5.23.2.8	TestCase_BC00107_MapinfoAddress0	81
5.23.2.9	TestCase_BC00108_MapinfoAddressMinus1	82
5.23.2.10	TestCase_BC00110_UnmapAnAlreadyUnmappedBulkBuffer	83
5.23.2.11	TestCase_GC00111_UnmapDoesntClear	83
5.24	01200.mcGetRndSeed	84
5.24.1	Detailed Description	84
5.24.2	Function Documentation	84
5.24.2.1	TestCase_GC01200_GetRngSeedByDriver	84
5.25	Runtime	85
5.25.1	Detailed Description	86
5.26	00100.TrustletCommands	86
5.26.1	Detailed Description	86
5.26.2	Function Documentation	86
5.26.2.1	TestCase_BC00100_TrustletCommandHasRespldSet	86
5.26.2.2	TestCase_BC00101_UnknownTestCommand	87
5.26.2.3	TestCase_GC00102_DoNothing	88
5.27	00300.Dead.Trustlet	88
5.27.1	Detailed Description	88
5.27.2	Function Documentation	89
5.27.2.1	TEST_SCENARIO	89
5.27.2.2	TestCase_BC00100_NotifyDeadTrustletWaitInfinite	89
5.27.2.3	TestCase_BC00101_MapMemoryToDeadTrustlet	89
5.28	00301.Trustlet_Endless_Loop	90
5.28.1	Detailed Description	90
5.28.2	Function Documentation	91
5.28.2.1	TestCase_BC00100_TrustletInEndlessLoop	91
5.29	00400.Communication	91
5.29.1	Detailed Description	91
5.29.2	Function Documentation	92
5.29.2.1	TestCase_GC00100_NoNotifyThenWait	92

5.29.2.2	TestCase_GC00101_NotifyT2ThenWait	93
5.29.2.3	TestCase_GC00102_NotifyT1ThenWait	93
5.29.2.4	TestCase_GC00103_NotifyT1T2ThenWait	93
5.29.2.5	TestCase_GC00104_NotifyT2T1ThenWait	93
5.29.2.6	TestCase_GC00105_SharedMemoryT1WritesT2Reads	93
5.29.2.7	TestCase_GC00106_UnmapSharedMemoryFromT1ThenCheckT2CanStillRead	94
5.30	00500_DynamicDriver	94
5.30.1	Detailed Description	94
5.30.2	Function Documentation	95
5.30.2.1	TestCase_GC00100_DriverAllocOk	95
5.30.2.2	TestCase_BC00101_DriverAllocTooMuch	95
5.30.2.3	TestCase_GC00102_DoNothing	96
5.31	00502_DynamicDriver3	96
5.31.1	Detailed Description	96
5.31.2	Function Documentation	97
5.31.2.1	TEST_SCENARIO	97
5.31.2.2	TestCase_GC00100_DriverMemoryMapping	97
5.32	00503_DynamicDriver4	97
5.32.1	Detailed Description	98
5.32.2	Function Documentation	98
5.32.2.1	TestCase_GC00100_MultipleDriverLoading_WithoutCall	98
5.32.2.2	TestCase_GC00101_MultipleDriverLoading_WithCall	98
5.32.2.3	TestCase_GC00102_MultipleDriverLoading_WithCall_2Trustlets	99
5.33	00600_CpuCoreAvailability	99
5.33.1	Detailed Description	99
5.33.2	Function Documentation	100
5.33.2.1	TestCase_GC00100_PlugOut_and_Plugin_Cpu1	100
5.34	Concurrency Tests	100
5.35	Trustlet Communication Tests	100
5.36	Trustlet API (TIApi)	100
5.36.1	Detailed Description	101
5.37	MCD_MCDIMPL_DAEMON	101
5.38	TLAPI_SEC	101
5.39	TIApi	101
5.39.1	Define Documentation	103
5.39.1.1	MAX_MAR_LIST_LENGTH	103
5.39.2	Typedef Documentation	103
5.39.2.1	void_ptr	103
5.39.3	Enumeration Type Documentation	104
5.39.3.1	kpdFuncID_t	104
5.39.3.2	cryptoFuncID_t	104
5.39.4	Function Documentation	105
5.39.4.1	callCryptoDriver	105
6	Data Structure Documentation	107
6.1	marshalingParam.t Struct Reference	107
6.1.1	Detailed Description	107
6.1.2	Field Documentation	107
6.1.2.1	functionId	107
6.2	TestUtils::BypassDaemon::notification.t Struct Reference	107

---

6.2.1	Detailed Description	108
6.2.2	Field Documentation	108
6.2.2.1	sessionId	108
6.2.2.2	payload	108
6.3	tlApiCrAbort_t Struct Reference	108
6.3.1	Detailed Description	108
<b>7</b>	<b>File Documentation</b>	<b>109</b>
7.1	MobiCoreDriverCmd.h File Reference	109
7.1.1	Detailed Description	109
7.2	TIApiCrypto.c File Reference	110
7.2.1	Detailed Description	110
7.3	TIApiMarshal.h File Reference	110
7.3.1	Detailed Description	112



# Chapter 1

## MobiCore Validation Test Suite

### Disclaimer

The validation suite does not perform any system or performance or stability test. The focus of the validation suite is to verify the MobiCore functionality.

### Introduction

This documentation contains the test specification of the MobiCore Validation Test Suite. The MobiCore Validation Test Suite can be used by OEMs to perform testing with MobiCore binaries. The document describes all the test cases that are implemented in the Trustlet Connector (TLC) 'ValidationTestSuite'.

These tests are implemented in one binary: -> *ValidationTestSuite*

A Trustlet is needed to interact with the Validation Test Suite.

-> Its binary file is called *05010000000000000000000000000000.tlbin*

Also a Driver is needed to interact with the Validation Test Suite.

-> Its binary file is called *02040000000000000000000000000000.tlbin*

To execute the tests please follow the instruction in chapter 'MobiCore Validation Test Suite' of the file 'Readme.pdf'.

### Glossary

#### A

ADB - Android Debug Bridge

Activation - Before a Service Provider can install and run Trustlets, MobiCore needs to be activated on the device by the System Owner. The use case Root Registration is equivalent to Activation. Activation changes the life cycle state of MobiCore to a fully

functional state. Note: before activation MobiCore may be available to support certain device security features, but will not be able to support Service Provider's trusted services.

API - Application Programming Interface

ARM - Advanced RISC Machines

## B

BIT - Binary Digit

## C

Container - Containers are hierarchical data structures holding content management relevant data and are associated to the various roles in the MobiCore ecosystem. MobiCore containers are the Root Container, Service Provider Containers and Trustlet Containers.

CM - Content Management

CM Trustlet (CMTL) - MobiCore System Trustlet and is responsible for content management operations in cooperation with the NWd Provisioning Agent.

CPU - Central Processing Unit

## D

DCI - Driver Control Interface

DDK - Device Driver Kit, SW development kit for creating device drivers for specific operating system.

DS-5 - ARM Development Studio 5

## F

FC - FastCall from the SWd to the MobiCore

FCI - FastCall interface of the MobiCore

FIQ - Fast Interrupt Request, concept of the ARM CPU for a high priority interrupt that can even interrupt the IRS of an IRQ. In a TZ system FIQs are used as secure interrupts handled in the SWd while IRQs are handled in the NWd.

## H

HAL - Hardware Abstraction Layer

---

**I**

IEC - International Electrotechnical Commission

IMEI - International Mobile Station Equipment Identity

IPC - Inter-Process Communication, communication between two tasks running in different processes.

IRQ - Interrupt Request

ISR - Interrupt Service Routine

IWC - Inter-World Communication, communication between two tasks running in different worlds.

**J**

JNI - Java Native Interface

**K**

KiB - Kibibyte, IEC term for 1024 Byte. The commonly used KB actually refers to 1.000 Byte.

**M**

MC - MobiCore

MCI - MobiCore Control Interface

MCP - MobiCore Control Protocol

MiB - Mebibyte, IEC term for 1024 KiB. The commonly used MB actually refers to 1.000 KB.

MU - Mapping Unit. A Trustlet has an address space of 8 MiB, where each MiB is a mapping unit. TCI/DCI is in the 2nd MU, 1 MiB WSM Blocks can be mapped in the 3rd to 8th MU

MMU - Memory Mapping Unit

MPU - Memory Protection Unit

MobiCore - G&D MobiCore

MobiCore System - The MobiCore System is an instantiation of a TEE as defined in OMTP. It is the software system consisting of a MobiCore enabled device and invoked by external components.

**N**

NQ - Notification Queue

NQ-IRQ - Notification Queue Interrupt raised by the Mobicore

S-SIQ - Secure Simulated Interrupt (becomes a IRQ for NWd)

N-SIQ - Non-secure Simulated Interrupt triggered by parameterized SMC call to Monitor from NWd

NWd - Normal World (the term "non-secure world" should be avoided). The software system running on an ARM TrustZone enabled device in non secure mode.

## O

OTA - Over-the-air

OS - Operating System

## P

PID - Process ID

PA - Provisioning Agent, NWd counterpart of the SWd Content Management Trustlet and cooperates with that System Trustlet for Content Management operations. The Provisioning Agent communicates with the Sytem Owner / Root and Service Provider backends. Furthermore the Provisioning Agent is responsible to store Content Management Secure Objects in NWd memory space.

## R

RFU - Reserved for future use

Rich OS - The platform OS like Symbian, Android, iOS, ... running in the NWd.

RISC - Reduced Instruction Set Computer

Root Container - The Root Container is associated to the role System Owner / Root and is available after MobiCore Root Registration. It organizes the Service Provider Containers of the Service Providers registered at the device.

RPC - Remote Procedure Call

RVCT - ARM RealView Compilation Tools, now called ARM Compiler

RVDS - ARM RealView Development Suite, discontinued and now replaced by DS-5

## S

SE - Secure Element

SO - Secure Object, used for the persistent storage of Authentication Token, Root, Service Provider, Trustlet Container and Trustlet Personalization data. A Secure Object is an encrypted and integrity protected data package of that data. A Secure Object is transferred from SWd to the Provisioning Agent Client application to be persisted in a NWd storage space.

Service - A Service consists of an Application, at least one Trustlet and its TLC.

Service Provider Container - A Service Provider Container is associated to a specific Service Provider holding content management relevant Service Provider data. A Service Provider Container is available after the registration of the Service Provider to the device. The Service Provider Container organizes the Trustlet Containers of its Trustlets.

SiP - Silicon Provider

SMC - Secure Monitor Call, instruction of the ARM CPU

SP - Service Provider

SP\_ID - Service Provider ID used to identify a registered Service Provider in MobiCore. SP\_ID is part of the Service Provider Container.

SUID - SoC Unique Identifier. The SUID is an identifier of the SoC and is unique within all SoCs. As the SoC is build in a device (handset) the SUID is used to identify the device in the MobiCore ecosystem.

SoC - System on a Chip

SWd - Secure World, software system running in secure mode in the ARM TrustZone. MobiCore is running in the SWd.

## T

TCI - Trustlet Control Interface.

TEE - Trusted Execution Environment. MobiCore is an instantiation of a TEE. The term TEE is defined in the OMTP specifications.

TL - Trustlet, Application running on MobiCore

TLAPI - Trustlet API

TLC - Trustlet Connector. Software running in the NWd providing a high level convenience interface to access a Trustlet in the SWd by the NWd client.

Trustlet - Trusted Application on the SWd side executing specific security services in the MobiCore runtime environment. The security services provided by a Trustlet are called by NWd client applications. MobiCore differentiates between System Trustlets like CM Trustlet and Trustlets of Service Providers.

Trustlet Container - A Trustlet Container is a container holding content management relevant data of a Trustlet (e.g. UUID and the Service Provider's key to encrypt Trustlet code).

TZ - ARM TrustZone

TZPC - TrustZone Protection Controller. The TZPC controls the configuration of memory regions and peripherals to be secure or non-secure.

**U**

UUID - Universal Unique Identifier. The UUID is an identifier standard used in software construction, standardized by the Open Software Foundation (OSF) as part of the Distributed Computing Environment (DCE). The UUID in the MobiCore ecosystem is used to uniquely identify Trustlets.

**W**

WSM - World Shared Memory. Shared memory which will be used for IPC between NWd and SWd.

## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

Test Subjects . . . . .	13
Performance . . . . .	14
Mobicore Inter Process Call Interface . . . . .	14
MemoryManagement . . . . .	14
00100_mcMapBoundaryTests . . . . .	15
Example Sub Group - TCI Buffer Tests . . . . .	29
McDrvApi . . . . .	29
00100_mcOpenDevice . . . . .	30
00200_mcCloseDevice . . . . .	34
00300_mcMallocWsm . . . . .	38
00400_mcFreeWsm . . . . .	44
00500_mcOpenSession . . . . .	47
00600_mcCloseSession . . . . .	55
00700_mcNotify . . . . .	58
00800_mcWaitNotification . . . . .	62
00900_mcGetSessionErrorCode . . . . .	67
01000_mcMap . . . . .	72
01100_mcUnmap . . . . .	77
01200_mcGetRndSeed . . . . .	84
Runtime . . . . .	85
00100_TrustletCommands . . . . .	86
00300_Dead_Trustlet . . . . .	88
00301_Trustlet_Endless_Loop . . . . .	90
00400_Communication . . . . .	91
00500_DynamicDriver . . . . .	94
00502_DynamicDriver3 . . . . .	96
00503_DynamicDriver4 . . . . .	97
00600_CpuCoreAvailability . . . . .	99
Concurrency Tests . . . . .	100

Trustlet Communication Tests . . . . .	100
Trustlet API (TIApi) . . . . .	100
Inter-world communication functions [COM]. . . . .	14
Security support functions [SEC]. . . . .	14
MobiCore system functions [SYS]. . . . .	14
Precondition setup . . . . .	14
Standard session opened . . . . .	14
MCD_MCDIMPL_DAEMON . . . . .	101
TLAPI_SEC . . . . .	101
TIApi . . . . .	101



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">marshalingParam_t</a> (Marshaled union ) . . . . .	107
<a href="#">TestUtils::BypassDaemon::notification_t</a> (Notification data structure ) . . . . .	107
<a href="#">tlApiCrAbort_t</a> (Marshaled function parameters ) . . . . .	108



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

<a href="#">MobiCoreDriverCmd.h</a> (Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1 ) . . . . .	109
<a href="#">TIApiCrypto.c</a> (TIApi security functions ) . . . . .	110
<a href="#">TIApiMarshal.h</a> (Marshaling types and declarations ) . . . . .	110



## Chapter 5

# Module Documentation

### 5.1 Test Subjects

The following test subjects are part of the MobiCore Integration Tests.

#### Modules

- [Performance](#)  
*This subject contains the test cases for performance testing.*
- [Mobicore Inter Process Call Interface](#)  
*These tests bypass the Trustlet API and issue IPCs directly.*
- [MemoryManagement](#)  
*These tests verify the correct working of allocations and deallocations in NWd components,  
as well as correct setup of shared memory with SWd.*
- [McDrvApi](#)  
*The MobiCore Driver API functions are tested here.*
- [Runtime](#)  
*Test of the runtime behavior of SWd components.*
- [Trustlet API \(TIApi\)](#)  
*These tests contain good cases and bad cases for the API functions.*

#### 5.1.1 Detailed Description

The following test subjects are part of the MobiCore Integration Tests.

## 5.2 Performance

This subject contains the test cases for performance testing.

This subject contains the test cases for performance testing.

## 5.3 Inter-world communication functions [COM].

## 5.4 Security support functions [SEC].

## 5.5 MobiCore system functions [SYS].

## 5.6 Mobicore Inter Process Call Interface

These tests bypass the Trustlet API and issue IPCs directly.

These tests bypass the Trustlet API and issue IPCs directly. All existing MobiCore threads are contacted. Because Trustlets are isolated, only IPCH should respond. Drivers and other Trustlets should not be reachable. RTM threads other than IPCH should not interact with Trustlet, e.g. ignore message, or send error message.

## 5.7 Precondition setup

Test groups defined by target set up and and tear down.

### Modules

- [Standard session opened](#)

### 5.7.1 Detailed Description

Test groups defined by target set up and and tear down.

## 5.8 Standard session opened

## 5.9 MemoryManagement

These tests verify the correct working of allocations and deallocations in NWd components,

as well as correct setup of shared memory with SWd.

## Modules

- [00100\\_mcMapBoundaryTests](#)  
*Involved Components and their required Functionality*
- [Example Sub Group - TCI Buffer Tests](#)

### 5.9.1 Detailed Description

These tests verify the correct working of allocations and deallocations in NWd components,

as well as correct setup of shared memory with SWd.

## 5.10 00100\_mcMapBoundaryTests

### Involved Components and their required Functionality

### Functions

- [TEST\\_SCENARIO](#) (00100\_mcMapBoundaryTests)
- [TestCase\\_GC00100\\_CheckBufferBoundariesSize1](#) ()  
*Verifies that the bidirectional NWd, SWd read write operations are successful on the buffer and page boundaries for a buffer size which is smaller than 1 page.*
- [TestCase\\_GC00101\\_CheckBufferBoundariesSize1Page](#) ()
- [TestCase\\_GC00102\\_CheckBufferBoundariesSize2Pages](#) ()
- [TestCase\\_GC00103\\_CheckBufferBoundariesSize3Pages](#) ()
- [TestCase\\_GC00104\\_CheckBufferBoundariesSize4Pages](#) ()
- [TestCase\\_GC00105\\_CheckBufferBoundariesSize8Pages](#) ()
- [TestCase\\_GC00106\\_CheckBufferFillAndReadbackSize1](#) ()
- [TestCase\\_GC00107\\_CheckBufferFillAndReadbackSize1Page](#) ()
- [TestCase\\_GC00108\\_CheckBufferFillAndReadbackSize2Pages](#) ()
- [TestCase\\_GC00109\\_CheckBufferFillAndReadbackSize3Pages](#) ()
- [TestCase\\_GC00110\\_CheckBufferFillAndReadbackSize4Pages](#) ()
- [TestCase\\_GC00111\\_CheckBufferFillAndReadbackSize8Pages](#) ()
- [TestCase\\_GC00112\\_MapAllAndReadbackSize1](#) ()
- [TestCase\\_GC00113\\_MapAllAndReadbackSize1Page](#) ()
- [TestCase\\_GC00114\\_MapAllAndReadbackSize2Pages](#) ()
- [TestCase\\_GC00115\\_MapAllAndReadbackSize3Pages](#) ()
- [TestCase\\_GC00116\\_MapAllAndReadbackSize4Pages](#) ()
- [TestCase\\_GC00117\\_MapAllAndReadbackSize8Pages](#) ()
- [TestCase\\_GC00118\\_MapAllAndReadbackSize255Pages](#) ()
- [TestCase\\_GC00119\\_MapDoesntClear](#) ()
- [TestCase\\_BC00120\\_TrustletPagefaultOnWrongAddress](#) ()

### Involved Components and their required Functionality

### Involved Components and their required Functionality

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) mcOpenDevice()
3. McDriverModule
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection
4. Trustlet TITestMemoryManagement
  - (a) Trustlet copied to Android on Test device in /data/app/ as 05010000000000000000000000000000.trst
  - (b) Trustlet runnable and bug-free

### Generic preconditions (setup)

Generic Preconditions are the same for every test case. They are executed always in front of every single test case.

- open device using the default ID by calling `mcOpenDevice()`
- allocate a WSM buffer (4k)
- open a session to the Trustlet

### Generic post processing (teardown)

Generic post processing is the same for every test case. It is executed always after every single test case.

- close session to the Trustlet
- free allocated memory of WSM buffer
- close Device by calling `mcCloseDevice()`



## 5.10.2 Function Documentation

### 5.10.2.1 TEST\_SCENARIO ( 00100\_mcMapBoundaryTests )

**Description:**

TLC maps additional Bulk memory to Trustlet and tests the access via TLC and Trustlet

Testsequence:

1. TLC fills Bulk buffer with a number and notifies the Trustlet
2. TL reads number from the Bulk buffer and increments it
3. TLC reads back number after notification from the TL

**Precondition**

Involved components running and functioning

**Verification:**

returns MC\_DRV\_OK.

Test output is OK.

### 5.10.2.2 TestCase\_GC00100.CheckBufferBoundariesSize1 ( )

Verifies that the bidirectional NWd, SWd read write operations are successful on the buffer and page boundaries for a buffer size which is smaller than 1 page.

**Description:**

Verifies that the bidirectional NWd, SWd read write operations are successful on the buffer and page boundaries for a buffer size which is smaller than 1 page.

**Precondition**

Involved components running and functioning

**Focus:**

CMD\_TEST\_FILL\_MEM command is executable for single byte memory insertion

**Verification:**

Page and buffer boundaries are written from SWd to NWd and verified  
Data is written from NWd to SWd and verified. CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00100\_CheckBufferBoundariesSize1

### 5.10.2.3 TestCase\_GC00101\_CheckBufferBoundariesSize1Page ( )

**Description:**

Verifies that the bidirectional NWd, SWd read write operations are successful on the buffer and page boundaries for a buffer sized smaller than 2 pages. Proves that memory allocation does not cause a boundary related problems with 2 pages affected.

**Precondition**

Involved components running and functioning

**Focus:**

CMD\_TEST\_FILL\_MEM command is executable for over 1 page size memory insertion

**Verification:**

Page and buffer boundaries are written from SWd to NWd and verified  
Data is written from NWd to SWd and verified.  
CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00101\_CheckBufferBoundariesSize1Page

### 5.10.2.4 TestCase\_GC00102\_CheckBufferBoundariesSize2Pages ( )

**Description:**

Verifies that the bidirectional NWd, SWd read write operations are successful on the buffer and page boundaries for a buffer sized smaller than 3 pages.

**Precondition**

Involved components running and functioning

**Focus:**

CMD\_TEST\_FILL\_MEM command is executable for over 2 page size memory insertion

**Verification:**

Page and buffer boundaries are written from SWd to NWd and verified  
Data is written from NWd to SWd and verified. CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00102\_CheckBufferBoundariesSize2Pages

**5.10.2.5 TestCase\_GC00103\_CheckBufferBoundariesSize3Pages ( )****Description:**

Verifies that the bidirectional NWd, SWd read write operations are successful on the buffer and page boundaries for a buffer sized smaller than 4 pages.

**Precondition**

Involved components running and functioning

**Focus:**

CMD\_TEST\_FILL\_MEM command is executable for over 3 page size memory insertion

**Verification:**

Page and buffer boundaries are written from SWd to NWd and verified  
Data is written from NWd to SWd and verified. CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00103\_CheckBufferBoundariesSize3Pages

**5.10.2.6 TestCase\_GC00104\_CheckBufferBoundariesSize4Pages ( )****Description:**

Verifies that the bidirectional NWd, SWd read write operations are successful on the buffer and page boundaries for a buffer sized smaller than 5 pages.

**Precondition**

Involved components running and functioning

**Focus:**

CMD\_TEST\_FILL\_MEM command is executable for over 4 page size memory insertion

**Verification:**

Page and buffer boundaries are written from SWd to NWd and verified  
Data is written from NWd to SWd and verified.  
CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00104\_CheckBufferBoundariesSize4Pages

**5.10.2.7 TestCase\_GC00105\_CheckBufferBoundariesSize8Pages ( )****Description:**

Verifies that the bidirectional NWd, SWd read write operations are successful on the buffer and page boundaries for a buffer sized smaller than 9 pages.

**Precondition**

Involved components running and functioning

**Focus:**

CMD\_TEST\_FILL\_MEM command is executable for over 8 page size memory insertion

**Verification:**

Page and buffer boundaries are written from SWd to NWd and verified  
Data is written from NWd to SWd and verified. CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00105\_CheckBufferBoundariesSize8Pages

**5.10.2.8 TestCase\_GC00106\_CheckBufferFillAndReadbackSize1 ( )****Description:****Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

Data block is written to the NWd via SWd.

Data is written from NWd to SWd and verified.

CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Bindings are unMap'ped and allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00106\_CheckBufferFillAndReadbackSize1

**5.10.2.9 TestCase\_GC00107\_CheckBufferFillAndReadbackSize1Page ( )****Description:**

Verifies that the integrity of randomly generated data written into the NWd via SWd is kept totally. Data consistency check is tested for data larger than a page size.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

Data block which is larger than a page size is written to the NWd via SWd.

Data is written from NWd to SWd and verified.

CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00107\_CheckBufferFillAndReadbackSize1Page

#### 5.10.2.10 TestCase\_GC00108\_CheckBufferFillAndReadbackSize2Pages ( )

**Description:****Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

Data block which is larger than 2 pages' size is written to the NWd via SWd.  
Data is written from NWd to SWd and verified.  
CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00108\_CheckBufferFillAndReadbackSize2Pages

#### 5.10.2.11 TestCase\_GC00109\_CheckBufferFillAndReadbackSize3Pages ( )

**Description:****Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

Data block which is larger than 3 pages' size is written to the NWd via SWd.  
Data is written from NWd to SWd and verified.  
CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00109\_CheckBufferFillAndReadbackSize3Pages

**5.10.2.12 TestCase\_GC00110\_CheckBufferFillAndReadbackSize4Pages ( )****Description:****Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

Data block which is larger than 4 pages' size is written to the NWd via SWd.

Data is written from NWd to SWd and verified.

CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00110\_CheckBufferFillAndReadbackSize4Pages

**5.10.2.13 TestCase\_GC00111\_CheckBufferFillAndReadbackSize8Pages ( )****Description:****Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

Data block which is larger than 8 pages' size is written to the NWd via SWd.

Data is written from NWd to SWd and verified.

CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00111\_CheckBufferFillAndReadbackSize8Pages

**5.10.2.14 TestCase\_GC00112\_MapAllAndReadbackSize1 ( )****Description:**

Verifies the SWd to NWd memory access with multiple buffers allocated, read and written.  
allocates 4 buffers with 1 byte, writes the randomly generated info to NWd memory via SWd and verifies the data.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

4 identical data blocks which sized 1 byte is written to the NWd via SWd.  
Data is written from NWd to SWd and verified.  
CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00112\_MapAllAndReadbackSize1

**5.10.2.15 TestCase\_GC00113\_MapAllAndReadbackSize1Page ( )****Description:**

Verifies the SWd to NWd memory access with multiple buffers allocated, read and written.  
allocates 4 buffers with data sized larger than 1 page size, writes the randomly generated info to  
NWd memory via SWd and verifies the data.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

4 identical data blocks which are sized larger than 1 pages is written to the NWd via SWd.



Data is written from NWd to SWd and verified.  
CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00113\_MapAllAndReadbackSize1Page

**5.10.2.16 TestCase\_GC00114\_MapAllAndReadbackSize2Pages ( )****Description:**

Verifies the SWd to NWd memory access with multiple buffers allocated, read and written.  
allocates 4 buffers with data sized larger than 2 pages' size, writes the randomly generated info to  
NWd memory via SWd and verifies the data.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

4 identical data blocks which are sized larger than 2 pages is written to the NWd via SWd.  
Data is written from NWd to SWd and verified.  
CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00114\_MapAllAndReadbackSize2Pages

**5.10.2.17 TestCase\_GC00115\_MapAllAndReadbackSize3Pages ( )****Description:**

Verifies the SWd to NWd memory access with multiple buffers allocated, read and written.

allocates 4 buffers with data sized larger than 3 pages' size, writes the randomly generated info to NWd memory via SWd and verifies the data.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

4 identical data blocks which are sized larger than 3 pages is written to the NWd via SWd.

Data is written from NWd to SWd and verified.

CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00115\_MapAllAndReadbackSize3Pages

**5.10.2.18 TestCase\_GC00116\_MapAllAndReadbackSize4Pages ( )****Description:**

Verifies the SWd to NWd memory access with multiple buffers allocated, read and written.

allocates 4 buffers with data sized larger than 4 pages' size, writes the randomly generated info to

NWd memory via SWd and verifies the data.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

4 identical data blocks which are sized larger than 4 pages is written to the NWd via SWd.

Data is written from NWd to SWd and verified.

CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00116\_MapAllAndReadbackSize4Pages

**5.10.2.19 TestCase\_GC00117\_MapAllAndReadbackSize8Pages ( )****Description:**

Verifies the SWd to NWd memory access with multiple buffers allocated, read and written.

allocates 4 buffers with data sized larger than 8 pages' size, writes the randomly generated info to NWd memory via SWd and verifies the data.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

4 identical data blocks which are sized larger than 8 pages is written to the NWd via SWd.

Data is written from NWd to SWd and verified.

CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00117\_MapAllAndReadbackSize8Pages

**5.10.2.20 TestCase\_GC00118\_MapAllAndReadbackSize255Pages ( )****Description:**

Verifies the SWd to NWd memory access with multiple buffers allocated, read and written.

allocates 4 buffers with data sized 255 pages' size, writes the randomly generated info to NWd memory via SWd and verifies the data.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap call maps the SWd memory synchronizes with NWd buffer.

**Verification:**

4 identical data blocks which are sized as big as 255 pages is written to the NWd via SWd.  
Data is written from NWd to SWd and verified.  
CMD\_TEST\_FILL\_MEM and CMD\_TEST\_COMPARE\_MEM commands are successfully executed.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00118\_MapAllAndReadbackSize255Pages

**5.10.2.21 TestCase\_GC00119\_MapDoesntClear ( )****Description:**

TLC writes something in buffer and then maps the buffer to Trustlet right after the moment of mapping buffer is verified to be cleared totally.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap clears the buffer when mapping takes place

**Verification:**

Mapping does not clear the buffer  
Test output is OK.

**Postcondition**

Allocated fields are freed.

**Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_GC00119\_MapDoesntClear

#### 5.10.2.22 TestCase\_BC00120\_TrustletPagefaultOnWrongAddress ( )

**Description:**

TLC maps a buffer to Trustlet and then orders Trustlet to write next to the buffer.

**Note**

This is more a test of mcUnmap(), bad we keep it here because it resembles the other tests here. TODO: move test in separate file or to unmap scenario

**Precondition**

Involved components running and functioning

**Focus:**

mcNotify informs TL to switch the access offset in the buffer.

**Verification:**

Trustlet pagefaults

mcUnmap() returns MC\_DRV\_OK, even if trustlet is dead. Test output is OK.

**Postcondition****Test ID:**

MemoryManagement\_00100\_mcMapBoundaryTests\_BC00120\_TrustletPagefaultOnWrongAddress

## 5.11 Example Sub Group - TCI Buffer Tests

## 5.12 McDrvApi

The MobiCore Driver API functions are tested here.

**Modules**

- [00100\\_mcOpenDevice](#)  
*Test Scenario for McDriverAPI function mcOpenDevice()*
- [00200\\_mcCloseDevice](#)  
*Test Scenario for McDriverAPI function mcCloseDevice()*
- [00300\\_mcMallocWsm](#)  
*Test Scenario for McDriverAPI function mcMallocWsm()*

- [00400\\_mcFreeWsm](#)  
*Involved Components and their required Functionality*
- [00500\\_mcOpenSession](#)  
*Involved Components and their required Functionality*
- [00600\\_mcCloseSession](#)  
*Involved Components and their required Functionality*
- [00700\\_mcNotify](#)  
*Involved Components and their required Functionality*
- [00800\\_mcWaitNotification](#)  
*Involved Components and their required Functionality*
- [00900\\_mcGetSessionErrorCode](#)  
*Involved Components and their required Functionality*
- [01000\\_mcMap](#)  
*Involved Components and their required Functionality*
- [01100\\_mcUnmap](#)  
*Involved Components and their required Functionality*
- [01200\\_mcGetRndSeed](#)  
*Involved Components and their required Functionality*

### 5.12.1 Detailed Description

The MobiCore Driver API functions are tested here. Testcases in MobiCore Driver API also test the communication API [COM] of the Trustlet API by using `tlNotify` and `tlWaitNotification`.

#### Introduction

Tests checking the basic functionality of the MobiCore driver interface.

## 5.13 00100\_mcOpenDevice

Test Scenario for McDriverAPI function `mcOpenDevice()`

## Functions

- [TestCase\\_GC00100\\_IdDefault \(\)](#)
- [TestCase\\_BC00101\\_InvalidDevice1 \(\)](#)
- [TestCase\\_BC00102\\_InvalidDevice2 \(\)](#)
- [TestCase\\_BC00103\\_InvalidDeviceMinus1 \(\)](#)
- [TestCase\\_BC00104\\_OpenRepeatedly \(\)](#)

### 5.13.1 Detailed Description

Test Scenario for McDriverAPI function mcOpenDevice()

#### Involved Components and their required Functionality

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
3. McDriverModule
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection

#### Generic preconditions (setup)

Generic Preconditions are the same for every test case. They are executed always in front of every single test case.

- none

#### Generic post processing (teardown)

Generic post processing is the same for every test case. It is executed always after every single test case.

- close Device by calling mcCloseDevice()

### 5.13.2 Function Documentation

#### 5.13.2.1 TestCase\_GC00100\_IdDefault ( )

**Description:**

TLC opens a new default device (MC\_DEVICE\_ID\_DEFAULT)

**Precondition**

-

**Focus:**

mcOpenDevice(MC\_DEVICE\_ID\_DEFAULT)

**Verification:**

MC\_DRV\_OK is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00100\_mcOpenDevice\_GC00100\_IdDefault

#### 5.13.2.2 TestCase\_BC00101\_InvalidDevice1 ( )

**Description:**

TLC opens an invalid device (1)

**Precondition**

-

**Focus:**

mcOpenDevice(1)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00100\_mcOpenDevice\_BC00101\_InvalidDevice1



**5.13.2.3 TestCase\_BC00102\_InvalidDevice2 ( )****Description:**

TLC opens an invalid device (2)

**Precondition**

-

**Focus:**

mcOpenDevice(2)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00100\_mcOpenDevice\_BC00102\_InvalidDevice2

**5.13.2.4 TestCase\_BC00103\_InvalidDeviceMinus1 ( )****Description:**

TLC opens an invalid device (-1)

**Precondition**

-

**Focus:**

mcOpenDevice(-1)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00100\_mcOpenDevice\_BC00103\_InvalidDeviceMinus1

#### 5.13.2.5 TestCase\_BC00104\_OpenRepeatedly ( )

**Description:**

TLC opens same device a second time

**Precondition**

1. mcOpenDevice(MC\_DEVICE\_ID\_DEFAULT) => returns MC\_DRV\_OK

**Focus:**

2. mcOpenDevice(MC\_DEVICE\_ID\_DEFAULT)

**Verification:**

MC\_DRV\_ERR\_INVALID\_OPERATION is returned for second time

**Postcondition**

-

**Test ID:**

McDrvApi\_00100\_mcOpenDevice\_BC00104\_OpenRepeatedly

## 5.14 00200\_mcCloseDevice

Test Scenario for McDriverAPI function mcCloseDevice()

**Functions**

- [TestCase\\_GC00100\\_IdDefault \(\)](#)
- [TestCase\\_BC00101\\_InvalidDevice1 \(\)](#)
- [TestCase\\_BC00102\\_InvalidDevice2 \(\)](#)
- [TestCase\\_BC00103\\_InvalidDeviceMinus1 \(\)](#)
- [TestCase\\_BC00103\\_CloseRepeatedly \(\)](#)

### 5.14.1 Detailed Description

Test Scenario for McDriverAPI function mcCloseDevice()

**Involved Components and their required Functionality**

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon

- (a) Daemon running
- (b) Daemon socket accessible by TLC
- (c) mcOpenDevice()

### 3. McDriverModule

- (a) Module loaded
- (b) Module socket accessible by daemon
- (c) Module listens for connection

#### Generic preconditions (setup)

Generic Preconditions are the same for every test case. They are executed always in front of every single test case.

- open device using the default ID by calling mcOpenDevice()

#### Generic post processing (teardown)

Generic post processing is the same for every test case. It is executed always after every single test case.

- close Device by calling mcCloseDevice() if it is not already closed

## 5.14.2 Function Documentation

### 5.14.2.1 TestCase\_GC00100\_IdDefault ( )

#### Description:

TLC closes a device session using the default device ID (MC\_DEVICE\_ID\_DEFAULT)

#### Precondition

A device has been opened using the default ID in Generic preconditions (setup)

#### Focus:

mcCloseDevice(MC\_DEVICE\_ID\_DEFAULT)

#### Verification:

MC\_DRV\_OK is returned

#### Postcondition

-

#### Test ID:

McDrvApi\_00200\_mcCloseDevice\_GC00100\_IdDefault

#### 5.14.2.2 TestCase\_BC00101\_InvalidDevice1 ( )

**Description:**

TLC closes a device with an invalid device ID (1)

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcCloseDevice(1)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00200\_mcCloseDevice\_BC00101\_InvalidDevice1

#### 5.14.2.3 TestCase\_BC00102\_InvalidDevice2 ( )

**Description:**

TLC closes a device with an invalid device ID (2)

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcCloseDevice(2)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00200\_mcCloseDevice\_BC00102\_InvalidDevice2

#### 5.14.2.4 TestCase\_BC00103\_InvalidDeviceMinus1 ( )

**Description:**

TLC closes a device with an invalid device ID (-1)

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcOpenDevice(-1)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00200\_mcCloseDevice\_BC00103\_InvalidDeviceMinus1

#### 5.14.2.5 TestCase\_BC00103\_CloseRepeatedly ( )

**Description:**

TLC closes a device which is already closed

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

1. mcCloseDevice(MC\_DEVICE\_ID\_DEFAULT) => returns MC\_DRV\_OK

**Focus:**

2. mcCloseDevice(MC\_DEVICE\_ID\_DEFAULT)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned for second time

**Postcondition**

-

**Test ID:**

McDrvApi\_00200\_mcCloseDevice\_BC00103\_CloseRepeatedly

## 5.15 00300\_mcMallocWsm

Test Scenario for McDriverAPI function mcMallocWsm()

### Functions

- [TestCase\\_GC00100\\_IdDefault4k \(\)](#)
- [TestCase\\_BC00101\\_InvalidDevice1 \(\)](#)
- [TestCase\\_BC00101\\_InvalidDevice4 \(\)](#)
- [TestCase\\_BC00102\\_InvalidDeviceMinus1 \(\)](#)
- [TestCase\\_BC00103\\_Len0 \(\)](#)
- [TestCase\\_BC00104\\_LenMinus1 \(\)](#)
- [TestCase\\_BC00105\\_WsmNullPointer \(\)](#)
- [TestCase\\_BC00106\\_ClosedDevice \(\)](#)
- [TestCase\\_GC00107\\_StresstestMultipleAllocations \(\)](#)
- [TestCase\\_GC00108\\_StresstestMultipleSizes \(\)](#)

### 5.15.1 Detailed Description

Test Scenario for McDriverAPI function mcMallocWsm()

#### Involved Components and their required Functionality

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) mcOpenDevice()
3. McDriverModule
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection

#### Generic preconditions (setup)

Generic Preconditions are the same for every test case. They are executed always in front of every single test case.

- open device using the default ID by calling mcOpenDevice()

**Generic post processing (teardown)**

Generic post processing is the same for every test case. It is executed always after every single test case.

- free allocated memory by calling mcFreeWsm() if it has been allocated
- close Device by calling mcCloseDevice()

**5.15.2 Function Documentation****5.15.2.1 TestCase\_GC00100\_IdDefault4k ( )****Description:**

TLC allocates 4kB of WSM

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcMallocWsm(MC\_DEVICE\_ID\_DEFAULT, 0, 4096, &tc, 0)

**Verification:**

MC\_DRV\_OK is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_GC00100\_IdDefault4k

**5.15.2.2 TestCase\_BC00101\_InvalidDevice1 ( )****Description:**

TLC calls mcMallocWsm with an invalid device id (1)

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcMallocWsm(1, 0, 4096, &tc, 0)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_BC00101\_InvalidDevice1

**5.15.2.3 TestCase\_BC00101\_InvalidDevice4 ( )****Description:**

TLC calls mcMallocWsm with an invalid device id (4)

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcMallocWsm(4, 0, 4096, &tc, 0)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_BC00101\_InvalidDevice4

**5.15.2.4 TestCase\_BC00102\_InvalidDeviceMinus1 ( )****Description:**

TLC calls mcMallocWsm with an invalid device id (-1)

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcMallocWsm(-1, 0, 4096, &tc, 0)

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

-



**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_BC00102\_InvalidDeviceMinus1

**5.15.2.5 TestCase\_BC00103\_Len0 ( )****Description:**

TLC calls mcMallocWsm with an unsupported len parameter (0)

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcMallocWsm(MC\_DEVICE\_ID\_DEFAULT, 0, 0, &tc, 0)

**Verification:**

MC\_DRV\_ERR\_NO\_FREE\_MEMORY is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_BC00103\_Len0

**5.15.2.6 TestCase\_BC00104\_LenMinus1 ( )****Description:**

TLC calls mcMallocWsm with an unsupported len parameter (-1)

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcMallocWsm(MC\_DEVICE\_ID\_DEFAULT, 0, -1, &tc, 0)

**Verification:**

MC\_DRV\_ERR\_NO\_FREE\_MEMORY is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_BC00104\_LenMinus1

#### 5.15.2.7 TestCase\_BC00105\_WsmNullPointer ( )

**Description:**

TLC calls mcMallocWsm with a NULL pointer

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

mcMallocWsm(MC\_DEVICE\_ID\_DEFAULT, 0, 4096, 0, 0)

**Verification:**

MC\_DRV\_ERR\_INVALID\_PARAMETER is returned

**Postcondition**

-

**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_BC00105\_WsmNullPointer

#### 5.15.2.8 TestCase\_BC00106\_ClosedDevice ( )

**Description:**

TLC calls mcMallocWsm with a not yet opened device

**Precondition**

Close device using the default ID

**Focus:**

mcMallocWsm

**Verification:**

MC\_DRV\_ERR\_UNKNOWN\_DEVICE is returned

**Postcondition**

Open device again using the default ID

**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_BC00106\_ClosedDevice

### 5.15.2.9 TestCase\_GC00107\_StresstestMultipleAllocations ( )

**Description:**

TLC calls mcMallocWsm 16 times  
After that, the memory (available to the TLC) is exhausted.  
After that, the TLC frees the memory again.  
Several runs of this test should not leak any other memory

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

call mcMallocWsm several times allocating blocks of 4kB

**Verification:**

MC\_DRV\_OK is returned until no more memory is available then  
MC\_DRV\_ERR\_NO\_FREE\_MEMORY is returned.

**Postcondition**

All memory is freed again by calling mcFreeWsm(), no memory leaks occur

**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_GC00107\_StresstestMultipleAllocations

### 5.15.2.10 TestCase\_GC00108\_StresstestMultipleSizes ( )

**Description:**

TLC calls mcMallocWsm and iteratively increases the size.  
The TLC frees the memory after each allocation.  
Several runs of this test should not leak any other memory

**Precondition**

A device has been opened using the default ID in Generic preconditions (setup)

**Focus:**

call mcMallocWsm several times allocating blocks with increasing block size

**Verification:**

MC\_DRV\_OK is returned until no more memory is available then  
MC\_DRV\_ERR\_NO\_FREE\_MEMORY is returned.

**Postcondition**

memory is not freed explicitly, but device is closed by generic post processing

**Test ID:**

McDrvApi\_00300\_mcMallocWsm\_GC00108\_StresstestMultipleSizes

## 5.16 00400\_mcFreeWsm

### Involved Components and their required Functionality

#### Functions

- [TestCase\\_GC00100\\_DeviceDefault](#) () 1)
- [TestCase\\_BC00101\\_DeviceInvalid](#) () 1)
- [TestCase\\_BC00102\\_WsmNullPointer](#) () 1)
- [TestCase\\_BC00103\\_MemoryNotManagedByLib](#) () 1)
- [TestCase\\_BC00104\\_OnClosedDevice](#) () 2)

### 5.16.1 Detailed Description

#### Involved Components and their required Functionality

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) mcOpenDevice()
3. McDriverModule
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection **Generic preconditions (setup)**

Generic Preconditions are the same for every test case. They are executed always in front of every single test case.
  - open device using the default ID by calling mcOpenDevice()
  - allocate a WSM buffer (4k)

#### Generic post processing (teardown)

Generic post processing is the same for every test case. It is executed always after every single test case.

- close session to the Trustlet
- free allocated memory of WSM buffer
- close Device by calling mcCloseDevice()

## 5.16.2 Function Documentation

### 5.16.2.1 TestCase\_GC00100\_DeviceDefault ( )

**Description:**

1. TLC allocates a block of WSM.
2. TLC frees the block again

**Precondition**

Involved components running and functioning

**Focus:**

mcFreeWsm works correctly.

**Verification:**

All checks pass, Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00400\_mcFreeWsm\_GC00100\_DeviceDefault

### 5.16.2.2 TestCase\_BC00101\_DeviceInvalid ( )

**Description:**

TLC calls mcFreeWsm with an unknown device id (1)

**Precondition**

Involved components running and functioning

**Focus:**

mcFreeWsm returns an error for an unknown device id.

**Verification:**

mcFreeWsm() returns MC\_DRV\_ERR\_UNKNOWN\_DEVICE, Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00400\_mcFreeWsm\_BC00101\_DeviceInvalid

### 5.16.2.3 TestCase\_BC00102\_WsmNullPointer ( )

**Description:**

TLC calls mcFreeWsm with a NULL pointer

**Precondition**

Involved components running and functioning

**Focus:**

mcFreeWsm returns an error for a null device id parameter.

**Verification:**

mcFreeWsm() returns MC\_DRV\_ERR\_INVALID\_PARAMETER, Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00400\_mcFreeWsm\_BC00102\_WsmNullPointer

### 5.16.2.4 TestCase\_BC00103\_MemoryNotManagedByLib ( )

**Description:**

TLC calls mcFreeWsm with memory not managed by MobiCore Driver

**Precondition**

Involved components running and functioning

**Focus:**

mcFreeWsm returns an error for a parameter points a memory block that is not controlled by driver.

**Verification:**

mcFreeWsm() returns MC\_DRV\_ERR\_INVALID\_PARAMETER, Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00400\_mcFreeWsm\_BC00103\_MemoryNotManagedByLib



### Involved Components and their required Functionality

### Involved Components and their required Functionality

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) mcOpenDevice()
3. McDriverModule
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection
4. Trustlet TIIntegrationTest
  - (a) Trustlet copied to Android on Test device in /data/app/ as 05010000000000000000000000000000.trst
  - (b) Trustlet runnable and bug-free

### Generic preconditions (setup)

Generic Preconditions are the same for every test case. They are executed always in front of every single test case.

- open device using the default ID by calling `mcOpenDevice()`
- allocate a WSM buffer (4k)
- open a session to the Trustlet

### Generic post processing (teardown)

Generic post processing is the same for every test case. It is executed always after every single test case.

- close session to the Trustlet
- free allocated memory of WSM buffer
- close Device by calling `mcCloseDevice()`



## 5.17.2 Function Documentation

### 5.17.2.1 TestCase\_GC00100\_050100000000000000000000000000 ( )

**Description:**

TLC opens a session to a trustlet

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession opens a session successfully

**Verification:**

mcOpenSession() returns MC\_DRV\_OK,

**Postcondition**

sessionId is not zero and not -1,  
Most significant bit in sessionId is not set,  
Test output is OK

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_GC00100\_050100000000000000000000000000

### 5.17.2.2 TestCase\_BC00101\_SessionNullPointer ( )

**Description:**

TLC opens a session with NULL pointer parameter for session

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession returns an error for a null session parameter.

**Verification:**

mcOpenSession() returns MC\_DRV\_ERR\_INVALID\_PARAMETER, Test output is  
OK

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00101\_SessionNullPointer

### 5.17.2.3 TestCase\_BC00102\_UuidNullPointer ( )

**Description:**

TLC opens a session with NULL pointer parameter for uuid

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession returns an error for a null uuid parameter.

**Verification:**

mcOpenSession() returns MC\_DRV\_ERR\_INVALID\_PARAMETER, Test output is OK

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00102\_UuidNullPointer

### 5.17.2.4 TestCase\_BC00103\_TciNullPointer ( )

**Description:**

TLC opens a session with NULL pointer parameter for tci

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession returns an error for a null tci parameter.

**Verification:**

mcOpenSession() returns MC\_DRV\_ERR\_INVALID\_PARAMETER, Test output is OK

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00103\_TciNullPointer

**5.17.2.5 TestCase\_BC00104\_LenBiggerMax ( )****Description:**

TLC opens a session with an invalid len parameter (len=MC\_MAX\_TCI\_LEN+1)

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession returns an error for a buffer exceeding length parameter.

**Verification:**

mcOpenSession() returns MC\_DRV\_ERR\_INVALID\_PARAMETER, Test output is OK

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00104\_LenBiggerMax

**5.17.2.6 TestCase\_BC00105\_LenMinus1 ( )****Description:**

TLC opens a session with an invalid len parameter (len=-1)

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession returns an error for a negative length parameter.

**Verification:**

mcOpenSession() returns MC\_DRV\_ERR\_INVALID\_PARAMETER, Test output is OK

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00105\_LenMinus1

#### 5.17.2.7 TestCase\_BC00106\_Len0 ( )

**Description:**

TLC opens a session with an invalid len parameter (len=0)

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession returns an error for a zero length parameter.

**Verification:**

mcOpenSession() returns MC\_DRV\_OK, Test output is OK.

**Note**

The outcome is rather unexpected, is a len of 0 ok? Actually, the session is created, but the Trustlet exits immediately. Exit can be recognized with mcWaitNotification() and mcGetSessionErrorCode() However, the session has to be closed anyhow

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00106\_Len0

#### 5.17.2.8 TestCase\_BC00107\_SessionhandleInvalid ( )

**Description:**

TLC opens a session with an invalid device id (-1)

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession returns an error for an unknown device ID.

**Verification:**

mcOpenSession() returns MC\_DRV\_ERR\_UNKNOWN\_DEVICE.  
Test output is OK.

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00107\_SessionhandleInvalid

**5.17.2.9 TestCase\_BC00108\_UuidUnknown123412341234 ( )****Description:**

TLC opens a session with an invalid UUID and that should be impossible.

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession returns an error for an invalid uuid parameter.

**Verification:**

mcOpenSession() returns MC\_DRV\_ERR\_UNKNOWN\_DEVICE.  
Test output is OK.

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00108\_UuidUnknown123412341234

**5.17.2.10 TestCase\_GC00109\_MaximumSessions ( )****Description:**

TLC opens 2 sessions, then out of memory

**Precondition**

Involved components running and functioning

**Focus:**

mcMallocWsm and mcOpenSession return errors to avoid memory exhaustion.

**Verification:**

Returns 2 times MC\_DRV\_OK, then MC\_DRV\_ERR\_INVALID\_DEVICE\_FILE.  
Test output is OK.

**Note**

This test also tests RTM ARM exception handling, at least on some boards.

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_GC00109\_MaximumSessions

#### 5.17.2.11 TestCase\_BC00110\_CloseDeviceWithOpenSession ( )

**Description:**

TLC tries to close a device with an open session attached to it

**Precondition**

Involved components running and functioning

**Focus:**

mcCloseDevice should return an error of a pending session.

**Verification:**

Returns MC\_DRV\_ERR\_SESSION\_PENDING, device and session stay open.  
Test output is OK.

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00110\_CloseDeviceWithOpenSession

#### 5.17.2.12 TestCase\_BC00111\_LeaveSession ( )

**Description:**

TLC tries to open a session and leave it open

**Precondition**

Involved components running and functioning

**Focus:**

mcOpenSession should open persistent sessions.

**Verification:**

Test output is OK.

**Postcondition**

-

**Test ID:**

McDrvApi\_00500\_mcOpenSession\_BC00111\_LeaveSession



- close session to the Trustlet
- free allocated memory of WSM buffer
- close Device by calling mcCloseDevice()

### 5.18.2 Function Documentation

#### 5.18.2.1 TestCase\_GC00100 ( )

**Description:**

TLC closes a default device (MC\_DEVICE\_ID\_DEFAULT) after successfully opening the device

**Precondition**

Involved components running and functioning

**Focus:**

mcCloseSession closes a session with default ID parameter.

**Verification:**

returns MC\_DRV\_OK.  
Test output is OK

**Postcondition**

**Test ID:**

McDrvApi\_00600\_mcCloseSession\_GC00100

#### 5.18.2.2 TestCase\_BC00101\_SessionNullPointer ( )

**Description:**

TLC calls function with a NULL pointer

**Precondition**

Involved components running and functioning

**Focus:**

mcCloseSession handles the NULL paramters

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK



**Postcondition****Test ID:**

McDrvApi\_00600\_mcCloseSession\_BC00101\_SessionNullPointer

**5.18.2.3 TestCase\_BC00102\_SessionInvalidId ( )****Description:**

TLC calls function with a malformed session id.  
Ensures the unknown session ids are handled.

**Precondition**

Involved components running and functioning

**Focus:**

mcCloseSession handles a bad session id.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_SESSION.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00600\_mcCloseSession\_BC00102\_SessionInvalidId

**5.18.2.4 TestCase\_BC00103\_SessionInvalidDevice ( )****Description:**

TLC calls function with a malformed deviceId. Ensures the unknown device ids are handled.

**Precondition**

Involved components running and functioning

**Focus:**

mcCloseSession handles a bad device id.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_DEVICE.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00600\_mcCloseSession\_BC00103\_SessionInvalidDevice

**5.18.2.5 TestCase\_BC00104\_CloseMultipleTimes ( )****Description:**

TLC tries to close a device multiple times. Ensures there is no remained of previously closed session.

**Precondition**

Involved components running and functioning

**Focus:**

mcCloseSession handles an already closed session's close request.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_SESSION.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00600\_mcCloseSession\_BC00104\_CloseMultipleTimes

**5.19 00700\_mcNotify****Involved Components and their required Functionality****Functions**

- [TestCase\\_GC00100\\_notifyOnce \( \)](#)
- [TestCase\\_BC00101\\_SessionhandleNullPointer \( \)](#)
- [TestCase\\_BC00102\\_SessionhandleInvalidId \( \)](#)
- [TestCase\\_BC00103\\_SessionhandleInvalidDevice \( \)](#)

### 5.19.1 Detailed Description

### Involved Components and their required Functionality

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) mcOpenDevice()
3. McDriverModule
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection
4. Trustlet TIIntegrationTest
  - (a) Trustlet copied to Android on Test device in /data/app/ as 05010000000000000000000000000000.trst
  - (b) Trustlet runnable and bug-free

### Generic preconditions (setup)

Generic Preconditions are the same for every test case. They are executed always in front of every single test case.

- open device using the default ID by calling `mcOpenDevice()`
- allocate a WSM buffer (4k)
- open a session to the Trustlet

### Generic post processing (teardown)

Generic post processing is the same for every test case. It is executed always after every single test case.

- close session to the Trustlet
- free allocated memory of WSM buffer
- close Device by calling `mcCloseDevice()`

## 5.19.2 Function Documentation

### 5.19.2.1 TestCase\_GC00100\_notifyOnce ( )

**Description:**

Trustlet Connector notifies Trustlet of a new command.  
Ensures the notification has been performed successfully.

**Precondition**

Involved components running and functioning

**Focus:**

mcNotify is performed for normal parameters.

**Verification:**

returns MC\_DRV\_OK.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00700\_mcNotify\_GC00100\_notifyOnce

### 5.19.2.2 TestCase\_BC00101\_SessionhandleNullPointer ( )

**Description:**

TLC calls function with a NULL pointer

**Precondition**

Involved components running and functioning

**Focus:**

mcNotify handles a NULL parameter.

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00700\_mcNotify\_BC00101\_SessionhandleNullPointer

**5.19.2.3 TestCase\_BC00102\_SessionhandleInvalidId ( )****Description:**

TLC calls function with an unknown sessionId.

**Precondition**

Involved components running and functioning

**Focus:**

mcNotify handles an unknown session ID.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_SESSION.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00700\_mcNotify\_BC00102\_SessionhandleInvalidId

**5.19.2.4 TestCase\_BC00103\_SessionhandleInvalidDevice ( )****Description:**

TLC calls function with a malformed handle, bad deviceId.

**Precondition**

Involved components running and functioning

**Focus:**

mcNotify handles an unknown device ID.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_DEVICE.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00700\_mcNotify\_BC00103\_SessionhandleInvalidDevice

### Involved Components and their required Functionality

- [TestCase\\_GC00100\\_NotifyThenTimeoutInfinite \(\)](#)
- [TestCase\\_GC00101\\_NotifyThenTimeoutFinite \(\)](#)
- [TestCase\\_GC00102\\_NoNotifyThenTimeoutFinite \(\)](#)
- [TestCase\\_BC00103\\_NoNotifyThenThenNoTimeout \(\)](#)
- [TestCase\\_BC00104\\_SessionhandleNullPointer \(\)](#)
- [TestCase\\_BC00105\\_SessionhandleInvalidId \(\)](#)
- [TestCase\\_BC00106\\_SessionhandleInvalidDevice \(\)](#)

### Involved Components and their required Functionality

- ### Generic preconditions (setup)

- open device using the default ID by calling `mcOpenDevice()`
- allocate a WSM buffer (4k)

- open a session to the Trustlet

**Generic post processing (teardown)**

Generic post processing is the same for every test case. It is executed always after every single test case.

- close session to the Trustlet
- free allocated memory of WSM buffer
- close Device by calling mcCloseDevice()

**5.20.2 Function Documentation****5.20.2.1 TestCase\_GC00100\_NotifyThenTimeoutInfinite ( )****Description:**

Trustlet Connector notifies Trustlet and waits infinitely for a notification of the Trustlet (response)

**Precondition**

Involved components running and functioning

**Focus:**

mcWaitNotification arbitrarily awaits for a notification of an empty message.

**Verification:**

mcWaitNotification() returns MC\_DRV\_OK.  
Response ID is set in message header on TCI.  
No additional error code is set.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_00800\_mcWaitNotification\_GC00100\_NotifyThenTimeoutInfinite

**5.20.2.2 TestCase\_GC00101\_NotifyThenTimeoutFinite ( )****Description:**

Trustlet Connector notifies Trustlet and waits 1 second for a notification of the Trustlet (response)

**Precondition**

Involved components running and functioning

**Focus:**

mcWaitNotification awaits 1 sec for a notification without an early timeout.

**Verification:**

mcWaitNotification() returns MC\_DRV\_OK.  
Response ID is set in message header on TCI.  
No additional error code is set.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_00800\_mcWaitNotification\_GC00101\_NotifyThenTimeoutFinite

**5.20.2.3 TestCase\_GC00102\_NoNotifyThenTimeoutFinite ( )****Description:**

Trustlet Connector does not notify Trustlet and waits 1 second for a notification of the Trustlet (response)

**Precondition**

Involved components running and functioning

**Focus:****Verification:**

mcWaitNotification() returns MC\_DRV\_ERR\_TIMEOUT.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_00800\_mcWaitNotification\_GC00102\_NoNotifyThenTimeoutFinite



**5.20.2.4 TestCase\_BC00103\_NoNotifyThenThenNoTimeout ( )****Description:**

Trustlet Connector does not notify Trustlet and waits infinitely for a notification of the Trustlet (response)

**Precondition**

Involved components running and functioning

**Focus:**

mcWaitNotification handles a no notification case.

**Verification:**

mcWaitNotification() returns MC\_DRV\_ERR\_NOTIFICATION.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_00800\_mcWaitNotification\_BC00103\_NoNotifyThenThenNoTimeout

**5.20.2.5 TestCase\_BC00104\_SessionhandleNullPointer ( )****Description:**

TLC calls function with a NULL pointer

**Precondition**

Involved components running and functioning

**Focus:**

mcWaitNotification handles a NULL parameter case.

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00800\_mcWaitNotification\_BC00104\_SessionhandleNullPointer

#### 5.20.2.6 TestCase\_BC00105\_SessionhandleInvalidId ( )

**Description:**

TLC calls function with a malformed handle, bad sessionId.

**Precondition**

Involved components running and functioning

**Focus:**

mcWaitNotification handles an invalid session ID case.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_SESSION.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00800\_mcWaitNotification\_BC00105\_SessionhandleInvalidId

#### 5.20.2.7 TestCase\_BC00106\_SessionhandleInvalidDevice ( )

**Description:**

TLC calls function with a malformed handle, bad deviceId.

**Precondition**

Involved components running and functioning

**Focus:**

mcWaitNotification handles an invalid device ID case.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_DEVICE.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00800\_mcWaitNotification\_BC00106\_SessionhandleInvalidDevice



- allocate a WSM buffer (4k)
- open a session to the Trustlet

**Generic post processing (teardown)**

Generic post processing is the same for every test case. It is executed always after every single test case.

- close session to the Trustlet
- free allocated memory of WSM buffer
- close Device by calling mcCloseDevice()

**5.21.2 Function Documentation****5.21.2.1 TestCase\_GC00050\_NoNotifications ( )****Description:**

Ensures that no session error is returned if there is none.

**Precondition**

Involved components running and functioning

**Focus:**

mcGetSessionErrorCode() returns MC\_DRV\_OK if there is no error.

**Verification:**

mcGetSessionErrorCode() returns MC\_DRV\_OK.  
No additional error code is set.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_00900\_mcGetSessionErrorCode\_GC00050\_NoNotifications

**5.21.2.2 TestCase\_GC00100\_MultipleNotifications ( )****Description:**

Ensures that trustlet Connector handles multiple notification of test trustlet successfully.

**Precondition**

Involved components running and functioning

**Focus:**

mcWaitNotification handles 3 notifications of trustlet.

**Verification:**

mcWaitNotification() returns MC\_DRV\_OK.  
Response ID is set in message header on TCI.  
No additional error code is set.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_00900\_mcGetSessionErrorCode\_GC00100\_MultipleNotifications

**5.21.2.3 TestCase\_GC00101\_TrustletExits ( )****Description:**

Trustlet Connector handles the "last" notification of the Trustlet after the Trustlet terminates itself with a wrong exit code.

**Precondition**

Involved components running and functioning

**Focus:**

mcGetSessionErrorCode is ERR\_INVALID\_EXIT\_CODE as the last error.

**Verification:**

mcWaitNotification() returns MC\_DRV\_INFO\_NOTIFICATION.  
The additional error code is set to -1: invalid exit code.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_00900\_mcGetSessionErrorCode\_GC00101\_TrustletExits

#### 5.21.2.4 TestCase\_GC00103\_TrustletCrashes ( )

**Description:**

Trustlet Connector handles the notification of the TL after the TL causes a kernel exception.

**Precondition**

Involved components running and functioning

**Focus:**

mcGetSessionErrorCode() returns "ERR\_INVALID\_OPERATION"

**Verification:**

mcWaitNotification() returns MC\_DRV\_INFO\_NOTIFICATION.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_00900\_mcGetSessionErrorCode\_GC00103\_TrustletCrashes

#### 5.21.2.5 TestCase\_BC00104\_SessionhandleNullPointer ( )

**Description:**

TLC calls function with a NULL pointer session

**Precondition**

Involved components running and functioning

**Focus:**

mcGetSessionErrorCode handles a null session.

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00900\_mcGetSessionErrorCode\_BC00104\_SessionhandleNullPointer

**5.21.2.6 TestCase\_BC00105\_LastErrNullPointer ( )****Description:**

TLC calls function with a NULL pointer lastErr

**Precondition**

Involved components running and functioning

**Focus:**

mcGetSessionErrorCode handles a null last error pointer.

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00900\_mcGetSessionErrorCode\_BC00105\_LastErrNullPointer

**5.21.2.7 TestCase\_BC00106\_SessionhandleInvalidId ( )****Description:**

TLC calls function with a malformed handle, bad sessionId.

**Precondition**

Involved components running and functioning

**Focus:**

mcGetSessionErrorCode handles an unknown session ID.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_SESSION.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00900\_mcGetSessionErrorCode\_BC00106\_SessionhandleInvalidId

#### 5.21.2.8 TestCase\_BC00107\_SessionhandleInvalidDevice ( )

**Description:**

TLC calls function with a malformed handle, bad deviceId.

**Precondition**

Involved components running and functioning

**Focus:**

mcGetSessionErrorCode handles an unknown device ID.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_DEVICE.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_00900\_mcGetSessionErrorCode\_BC00107\_SessionhandleInvalidDevice

## 5.22 01000\_mcMap

**Involved Components and their required Functionality****Functions**

- [TestCase\\_GC00100\\_Map4k \( \)](#)
- [TestCase\\_BC00101\\_SessionhandleNullPointer \( \)](#)
- [TestCase\\_BC00102\\_MapinfoNullPointer \( \)](#)
- [TestCase\\_BC00103\\_SessionhandleInvalidId \( \)](#)
- [TestCase\\_BC00104\\_SessionhandleInvalidDevice \( \)](#)
- [TestCase\\_BC00105\\_MultipleMappingsOfSameBuffer \( \)](#)
- [TestCase\\_BC00106\\_MapAReadOnlyBuffer \( \)](#)

### 5.22.1 Detailed Description

**Involved Components and their required Functionality**

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection



## 2. McDriverDaemon

- (a) Daemon running
- (b) Daemon socket accessible by TLC
- (c) mcOpenDevice()

### 3. McDriverModule

- (a) Module loaded
- (b) Module socket accessible by daemon
- (c) Module listens for connection

#### 4. Trustlet TlIntegrationTest

- (a) Trustlet copied to Android on Test device in /data/app/ as 0501000000000000000000000000.trst
- (b) Trustlet runnable and bug-free

### Generic preconditions (setup)

Generic Preconditions are the same for every test case. They are executed always in front of every single test case.

- open device using the default ID by calling `mcOpenDevice()`
- allocate a WSM buffer (4k)
- open a session to the Trustlet

### Generic post processing (teardown)

Generic post processing is the same for every test case. It is executed always after every single test case.

- close session to the Trustlet
- free allocated memory of WSM buffer
- close Device by calling `mcCloseDevice()`

### 5.22.2 Function Documentation

#### 5.22.2.1 TestCase\_GC00100\_Map4k ( )

**Description:**

TLC maps additional Bulk memory to Trustlet

### Precondition

Involved components running and functioning

**Focus:**

mcMap does not map additional bulk memory.

**Verification:**

returns MC\_DRV\_OK.  
returned Buffer length is equal to requested length.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_01000\_mcMap\_GC00100\_Map4k

**5.22.2.2 TestCase\_BC00101\_SessionhandleNullPointer ( )****Description:**

TLC calls function with a NULL pointer session

**Precondition**

Involved components running and functioning

**Focus:**

mcMap handles a null session pointer.

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01000\_mcMap\_BC00101\_SessionhandleNullPointer

**5.22.2.3 TestCase\_BC00102\_MapinfoNullPointer ( )****Description:**

TLC calls function with a NULL pointer mapInfo

**Precondition**

Involved components running and functioning

**Focus:**

mcMap handles a null map info pointer.

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01000\_mcMap\_BC00102\_MapinfoNullPointer

**5.22.2.4 TestCase\_BC00103\_SessionhandleInvalidId ( )****Description:**

TLC calls function with a malformed handle, bad sessionId.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap handles an invalid session ID.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_SESSION.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01000\_mcMap\_BC00103\_SessionhandleInvalidId

**5.22.2.5 TestCase\_BC00104\_SessionhandleInvalidDevice ( )****Description:**

TLC calls function with a malformed handle, bad deviceId.

**Precondition**

Involved components running and functioning

**Focus:**

mcMap handles an unknown device id.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_DEVICE.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01000\_mcMap\_BC00104\_SessionhandleInvalidDevice

**5.22.2.6 TestCase\_BC00105\_MultipleMappingsOfSameBuffer ( )****Description:**

TLC maps the same additional bulk memory to one Trustlet twice

**Precondition**

Involved components running and functioning

**Focus:**

mcMap handles a multiple memory mapping on same Trustlet.

**Verification:**

returns MC\_DRV\_ERR\_BULK\_MAPPING.  
The bulk buffer cannot be mapped to the same address.  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_01000\_mcMap\_BC00105\_MultipleMappingsOfSameBuffer

**5.22.2.7 TestCase\_BC00106\_MapAReadonlyBuffer ( )****Description:**

TLC tries to map read-only data to Trustlet

**Precondition**

Involved components running and functioning

**Focus:**

mcMap handles a readonly memory area mapping.

**Verification:**

returns MC\_DRV\_ERR\_BULK\_MAPPING.

The bulk buffer cannot be mapped because it points to readonly data.

Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_01000\_mcMap\_BC00106\_MapAReadOnlyBuffer

## 5.23 01100\_mcUnmap

**Involved Components and their required Functionality****Functions**

- [TestCase\\_GC00100\\_Unmap4k \(\)](#)
- [TestCase\\_BC00101\\_SessionhandleNullPointer \(\)](#)
- [TestCase\\_BC00102\\_BufNullPointer \(\)](#)
- [TestCase\\_BC00103\\_MapinfoNullPointer \(\)](#)
- [TestCase\\_BC00104\\_SessionhandleInvalidId \(\)](#)
- [TestCase\\_BC00105\\_SessionhandleInvalidDevice \(\)](#)
- [TestCase\\_BC00106\\_BufferPointerNotManagedByLib \(\)](#)
- [TestCase\\_BC00107\\_MapinfoAddress0 \(\)](#)
- [TestCase\\_BC00108\\_MapinfoAddressMinus1 \(\)](#)
- [TestCase\\_BC00110\\_UnmapAnAlreadyUnmappedBulkBuffer \(\)](#)
- [TestCase\\_GC00111\\_UnmapDoesntClear \(\)](#)

### 5.23.1 Detailed Description

**Involved Components and their required Functionality**

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running

- ### 5.23.2 Function Documentation

**Description:**

### Precondition

### Focus:

**Verification:**

### Postcondition

**Test ID:**

#### 5.23.2.2 TestCase\_BC00101\_SessionhandleNullPointer ( )

### Precondition

Trustonic MobiCore®

**Focus:**

mcUnMap handles a null session pointer.

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_BC00101\_SessionhandleNullPointer

**5.23.2.3 TestCase\_BC00102\_BufNullPointer ( )****Description:**

TLC calls function with a NULL pointer bulkBuffer

**Precondition**

Involved components running and functioning

**Focus:**

mcUnMap handles a null bulk data pointer.

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_BC00102\_BufNullPointer

**5.23.2.4 TestCase\_BC00103\_MapinfoNullPointer ( )****Description:**

TLC calls function with a NULL pointer mapInfo

**Precondition**

Involved components running and functioning

**Focus:**

mcUnMap handles a null buffer info pointer.

**Verification:**

returns MC\_DRV\_ERR\_INVALID\_PARAMETER.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_BC00103\_MapinfoNullPointer

**5.23.2.5 TestCase\_BC00104\_SessionhandleInvalidId ( )****Description:**

TLC calls function with a malformed handle, bad sessionId.

**Precondition**

Involved components running and functioning

**Focus:**

mcUnMap handles a unknown session ID.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_SESSION.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_BC00104\_SessionhandleInvalidId

**5.23.2.6 TestCase\_BC00105\_SessionhandleInvalidDevice ( )****Description:**

TLC calls function with a malformed handle, bad deviceId.

**Precondition**

Involved components running and functioning



**Focus:**

mcUnMap handles an invalid device ID.

**Verification:**

returns MC\_DRV\_ERR\_UNKNOWN\_DEVICE.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_BC00105\_SessionhandleInvalidDevice

**5.23.2.7 TestCase\_BC00106\_BufferPointerNotManagedByLib ( )****Description:**

TLC calls function with a bad bulkBuf.

**Precondition**

Involved components running and functioning

**Focus:**

mcUnMap handles an oversized bulk buffer.

**Verification:**

returns MC\_DRV\_ERR\_BULK\_UNMAPPING.  
Test output is OK

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_BC00106\_BufferPointerNotManagedByLib

**5.23.2.8 TestCase\_BC00107\_MapinfoAddress0 ( )****Description:**

TLC calls function with a malformed mapInfo

**Precondition**

Involved components running and functioning

**Focus:**

mcUnMap handles a null virtual addressed map info.

**Verification:**

returns MC\_DRV\_ERR\_BULK\_UNMAPPING.  
Test output is OK.

**Note**

This test also tests RTM ARM exception handling.

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_BC00107\_MapinfoAddress0

**5.23.2.9 TestCase\_BC00108\_MapinfoAddressMinus1 ( )****Description:**

TLC calls function with a malformed mapInfo

**Precondition**

Involved components running and functioning

**Focus:**

mcUnMap handles a bad virtual addressed map info.

**Verification:**

returns MC\_DRV\_ERR\_BULK\_UNMAPPING.  
Test output is OK.

**Note**

This test also tests RTM ARM exception handling.

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_BC00108\_MapinfoAddressMinus1

**5.23.2.10 TestCase\_BC00110\_UnmapAnAlreadyUnmappedBulkBuffer ( )****Description:**

TLC unmaps the same bulk buffer from Trustlet twice

**Precondition**

Involved components running and functioning

**Focus:**

mcUnMap handles a multiple unmapping on the same session.

**Verification:**

returns MC\_DRV\_ERR\_BULK\_UNMAPPING.

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_BC00110\_UnmapAnAlreadyUnmappedBulkBuffer

**5.23.2.11 TestCase\_GC00111\_UnmapDoesntClear ( )****Description:**

TLC writes something in buffer and then maps and unmaps the buffer to Trustlet

**Precondition**

Involved components running and functioning

**Focus:**

mcMap and mcUnMap handle an already written bulk buffer.

**Verification:**

Unmapping does not clear the buffer  
Test output is OK.

**Postcondition****Test ID:**

McDrvApi\_01100\_mcUnmap\_GC00111\_UnmapDoesntClear



**Verification:**

mapped area includes non-zero values after RNG seed read operation successfully performed.

**Postcondition****Test ID:**

McDrvApi\_01200\_mcGetRngSeed\_GC01200\_GetRngSeedByDriver

## 5.25 Runtime

Test of the runtime behavior of SWd components.

**Modules**

- [00100\\_TrustletCommands](#)  
*Involved Components and their required Functionality*
- [00300\\_Dead\\_Trustlet](#)  
*Involved Components and their required Functionality*
- [00301\\_Trustlet\\_Endless\\_Loop](#)  
*Involved Components and their required Functionality*
- [00400\\_Communication](#)  
*Parallel Notifications and Mappings.*
- [00500\\_DynamicDriver](#)  
*Involved Components and their required Functionality*
- [00502\\_DynamicDriver3](#)  
*Involved Components and their required Functionality*
- [00503\\_DynamicDriver4](#)  
*Involved Components and their required Functionality*
- [00600\\_CpuCoreAvailability](#)  
*Involved Components and their required Functionality*
- [Concurrency Tests](#)
- [Trustlet Communication Tests](#)

Test of the runtime behavior of SWd components.

### Involved Components and their required Functionality

- `TestCase_BC00100_TrustletCommandHasRespIdSet ()`
- `TestCase_BC00101_UnknownTestCommand ()`
- `TestCase_GC00102_DoNothing ()`

### Involved Components and their required Functionality

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) mcOpenDevice()
3. McDriverModule
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection
4. Trustlet TIIntegrationTest
  - (a) Trustlet copied to Android on Test device in /data/app/ as 05010000000000000000000000000000.trst

#### 5.26.2.1 TestCase\_BC00100\_TrustletCommandHasRespIdSet ( )

Trustlet Connector notifies Trustlet and waits infinitely for a notification of the Trustlet (response) TLC uses a bad command, with RSP\_ID set. Trustlet just answers back, no error message

**Precondition**

Session has been opened by setup()

**Focus:**

Use mcNotify() to send a valid command to the trustlet  
Wait for notification from Trustlet

**Verification:**

mcWaitNotification() returns MC\_DRV\_OK.  
No additional error code is set.

**Postcondition**

Session is closed by teardown()

**Test ID:**

Runtime\_00100\_TrustletCommands\_BC00100\_TrustletCommandHasRespIdSet

**5.26.2.2 TestCase\_BC00101\_UnknownTestCommand ( )****Description:**

Trustlet Connector notifies Trustlet and waits infinitely for a notification of the Trustlet (response) TLC uses a bad command, with RSP\_ID set. Trustlet just answers back, no error message

**Precondition**

Session has been opened by setup()

**Focus:**

Use mcNotify() to send an unknown command to the trustlet  
Wait for notification from Trustlet

**Verification:**

mcWaitNotification() returns MC\_DRV\_OK.  
check that tci->responseHeader.returncode == RET\_ERR\_UNKNOWN\_CMD  
No additional error code is set.

**Postcondition**

Session is closed by teardown()

**Test ID:**

Runtime\_00100\_TrustletCommands\_BC00101\_UnknownTestCommand

### 5.26.2.3 TestCase\_GC00102\_DoNothing ( )

**Description:**

Trustlet does nothing (Immediate open/close session). Setup() opens the session. TestCase does nothing and Teardown() closes the session.

**Precondition**

Session has been opened by setup()

**Focus:**

- 

**Verification:**

- 

**Postcondition**

Session is closed by teardown()

**Test ID:**

Runtime\_00100\_TrustletCommands\_GC00102\_DoNothing

## 5.27 00300\_Dead\_Trustlet

**Involved Components and their required Functionality****Functions**

- [TEST\\_SCENARIO](#) (00300\_Dead\_Trustlet)
- [TestCase\\_BC00100\\_NotifyDeadTrustletWaitInfinite](#) ()
- [TestCase\\_BC00101\\_MapMemoryToDeadTrustlet](#) ()

### 5.27.1 Detailed Description

**Involved Components and their required Functionality**

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running



- (b) Daemon socket accessible by TLC
  - (c) mcOpenDevice()
3. McDriverModule
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection
4. Trustlet TIIntegrationTest
  - (a) Trustlet copied to Android on Test device in /data/app/ as 05010000000000000000000000000000.trst
  - (b) Trustlet runnable and bug-free

### 5.27.2 Function Documentation

#### 5.27.2.1 TEST\_SCENARIO ( 00300\_Dead\_Trustlet )

Setup: TLC handles the "last" notification of the Trustlet after the Trustlet terminates itself with a wrong exit code.

#### 5.27.2.2 TestCase\_BC00100\_NotifyDeadTrustletWaitInfinite ( )

**Description:**

Afterwards TLC tries to notify the Trustlet Then it waits for the Trustlet

### Precondition

Involved components running and functioning

**Verification:**

mcWaitNotification() returns MC\_DRV\_INFO\_NOTIFICATION.  
The additional error code is set to -1: invalid exit code.  
mcNotify returns no error mcWaitNotification returns an error Test output is OK.

**Test ID:**

```
Runtime 00300 Dead Trustlet BC00100 NotifyDeadTrustletWaitInfinite
```

### 5.27.2.3 TestCase\_BC00101\_MapMemoryToDeadTrustlet ( )

**Description:**

Afterwards TLC tries to map bulk memory to the Trustlet

See also `Test10_mcMap_BC_pagefault_on_wrong_address` for a Test of `mcUnmap()` on a dead trustlet.

**Precondition**

Involved components running and functioning

**Verification:**

`mcMap()` returns `MC_DRV_ERR_DAEMON_UNREACHABLE`.

**Note**

This test also tests RTM ARM exception handling. Test output is OK.

**Test ID:**

`Runtime_00300_Dead_Trustlet_BC00101_MapMemoryToDeadTrustlet`

## 5.28 00301\_Trustlet\_Endless\_Loop

**Involved Components and their required Functionality****Functions**

- [TestCase\\_BC00100\\_TrustletInEndlessLoop \(\)](#)

### 5.28.1 Detailed Description

**Involved Components and their required Functionality**

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) `mcOpenDevice()`
3. McDriverModule
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection

#### 4. Trustlet TlIntegrationTest

- (a) Trustlet copied to Android on Test device in /data/app/ as 0501000000000000000000000000.trst
- (b) Trustlet runnable and bug-free

### 5.28.2 Function Documentation

#### 5.28.2.1 TestCase\_BC00100\_TrustletInEndlessLoop ( )

**Description:**

TLC tells Trustlet to go into infinite loop. Then TLC closes Session.

### Precondition

Involved components running and functioning

**Verification:**

Session can be closed. Test output is OK.

**Test ID:**

Runtime\_00301\_Trustlet\_Endless\_Loop\_BC00100\_TrustletInEndlessLoop

## 5.29 00400\_Communication

### Parallel Notifications and Mappings.

## Functions

- `TestCase_GC00100_NoNotifyThenWait ()`
- `TestCase_GC00101_NotifyT2ThenWait ()`
- `TestCase_GC00102_NotifyT1ThenWait ()`
- `TestCase_GC00103_NotifyT1T2ThenWait ()`
- `TestCase_GC00104_NotifyT2T1ThenWait ()`
- `TestCase_GC00105_SharedMemoryT1WritesT2Reads ()`
- `TestCase_GC00106_UnmapSharedMemoryFromT1ThenCheckT2CanStillRead ()`

### 5.29.1 Detailed Description

### Parallel Notifications and Mappings. Involved Components and their required Functionality

## 1. ClientLib

- (a) Can connect to daemon

- ### 5.29.2 Function Documentation

**Description:**

### Precondition

**Verification:**

### Note

**Test ID:**

TruStonic MobiCore®

**5.29.2.2 TestCase\_GC00101\_NotifyT2ThenWait ( )****Test ID:**

Runtime\_00400\_Communication\_GC00101\_NotifyT2ThenWait

**5.29.2.3 TestCase\_GC00102\_NotifyT1ThenWait ( )****Test ID:**

Runtime\_00400\_Communication\_GC00102\_NotifyT1ThenWait

**5.29.2.4 TestCase\_GC00103\_NotifyT1T2ThenWait ( )****Test ID:**

Runtime\_00400\_Communication\_GC00103\_NotifyT1T2ThenWait

**5.29.2.5 TestCase\_GC00104\_NotifyT2T1ThenWait ( )****Test ID:**

Runtime\_00400\_Communication\_GC00104\_NotifyT2T1ThenWait

**5.29.2.6 TestCase\_GC00105\_SharedMemoryT1WritesT2Reads ( )****Description:**

TLC maps buffer to two Trustlets  
Trustlet 1 writes in buffer.  
Trustlet 2 reads the buffer.

**Precondition**

Involved components running and functioning

**Verification:**

Trustlet 2 reads what Trustlet 1 wrote.  
Test output is OK.

**Test ID:**

Runtime\_00400\_Communication\_GC00105\_SharedMemoryT1WritesT2Reads

### 5.29.2.7 `TestCase_GC00106_UnmapSharedMemoryFromT1ThenCheckT2CanStillRead ( )`

**Description:**

TLC writes something in buffer and then maps it to two Trustlets.  
Then TLC unmaps the buffer from one Trustlet.  
The other Trustlet can still read the buffer.

**Precondition**

Involved components running and functioning

**Verification:**

Unmapping does not unmap the buffer everywhere.  
Test output is OK.

**Test ID:**

Runtime\_00400\_Communication\_GC00106\_UnmapSharedMemoryFromT1ThenCheckT2CanStillRead

## 5.30 00500 DynamicDriver

**Involved Components and their required Functionality****Functions**

- [TestCase\\_GC00100\\_DriverAllocOk \( \)](#)
- [TestCase\\_BC00101\\_DriverAllocTooMuch \( \)](#)
- [TestCase\\_GC00102\\_DoNothing \( \)](#)

### 5.30.1 Detailed Description

**Involved Components and their required Functionality**

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) `mcOpenDevice()`
3. DrIntegrationTest Driver
  - (a) Module loaded

- (b) Module socket accessible by daemon
  - (c) Module listens for connection
4. Trustlet TITestIntegrationTest
- (a) Trustlet copied to Android on Test device in /data/app/ as 05010000000000000000000000000000.trst
  - (b) Trustlet runnable and bug-free

### 5.30.2 Function Documentation

#### 5.30.2.1 TestCase\_GC00100\_DriverAllocOk ( )

**Description:**

Trustlet Connector notifies Trustlet and allocate 1 resource Trustlet just answers with accept message, no error message

### Precondition

Involved components running and functioning

**Verification:**

mcWaitNotification() returns MC\_DRV\_OK.

**Verification:**

returnCode == RET\_OK No additional error code is set.  
Test output is OK.

**Test ID:**

Runtime\_00500\_DynamicDriver\_GC00100\_DriverAllocOk

#### 5.30.2.2 TestCase\_BC00101\_DriverAllocTooMuch ( )

**Description:**

Trustlet Connector notifies Trustlet and allocate 2 resource Trustlet answers first with accept message, no error message Trustlet answers first with not accept message, error message

### Precondition

Involved components running and functioning

**Verification:**

mcWaitNotification() returns MC\_DRV\_OK.

**Verification:**

first returnCode == RET\_OK No additional error code is set.  
Test output is OK.

**Test ID:**

Runtime\_00500\_DynamicDriver\_BC00101\_DriverAllocTooMuch

**5.30.2.3 TestCase\_GC00102\_DoNothing ( )****Description:**

Testcase without doing anything Open/CloseSession without any duration between

**Precondition**

Involved components running and functioning

**Verification:**

mcWaitNotification() returns MC\_DRV\_OK.

**Verification:**

first returnCode == RET\_OK No additional error code is set.  
Test output is OK.

**Test ID:**

Runtime\_00500\_DynamicDriver\_GC00102\_DoNothing

**5.31 00502\_DynamicDriver3****Involved Components and their required Functionality****Functions**

- [TEST\\_SCENARIO](#) (00502\_DynamicDriver3)
- [TestCase\\_GC00100\\_DriverMemoryMapping](#) ()

**5.31.1 Detailed Description****Involved Components and their required Functionality**

1. ClientLib
  - (a) Can connect to daemon



- (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) mcOpenDevice()
3. DrIntegrationTest Driver
  - (a) Module loaded
  - (b) Module socket accessible by daemon
  - (c) Module listens for connection
4. Trustlet TITestIntegrationTest
  - (a) Trustlet copied to Android on Test device in /data/app/ as 05010000000000000000000000000000.trst
  - (b) Trustlet runnable and bug-free

### 5.31.2 Function Documentation

#### 5.31.2.1 TEST\_SCENARIO ( 00502\_DynamicDriver3 )

## !!THP faked driver as trustlet

### 5.31.2.2 TestCase\_GC00100\_DriverMemoryMapping ( )

**Description:**

## Trustlet Connector opens two Trustlets

### Precondition

Involved components running and functioning Testflow.

**Test ID:**

Runtime\_00502\_DynamicDriver3\_GC00100\_DriverMemoryMapping

### 5.32 00503\_DynamicDriver4

### Involved Components and their required Functionality

## Functions

- [TestCase\\_GC00100\\_MultipleDriverLoading\\_WithoutCall \(\)](#)
- [TestCase\\_GC00101\\_MultipleDriverLoading\\_WithCall \(\)](#)
- [TestCase\\_GC00102\\_MultipleDriverLoading\\_WithCall\\_2Trustlets \(\)](#)



**Precondition**

Involved components running and functioning Testflow.

**Test ID:**

Runtime\_00503\_DynamicDriver4\_GC00101\_MultipleDriverLoading\_WithCall

!!THP faked driver as trustlet

**5.32.2.3 TestCase\_GC00102\_MultipleDriverLoading\_WithCall\_2Trustlets ( )****Description:**

Trustlet Connector reopens multiple time trustlet+driver with ping trustlet->driver with 2 trustlets

**Precondition**

Involved components running and functioning Testflow.

**Test ID:**

Runtime\_00503\_DynamicDriver4\_GC00102\_MultipleDriverLoading\_WithCall\_2Trustlets

!!THP faked driver as trustlet

## 5.33 00600\_CpuCoreAvailability

**Involved Components and their required Functionality****Functions**

- [TestCase\\_GC00100\\_PlugOut\\_and\\_Plugin\\_Cpu1 \( \)](#)

### 5.33.1 Detailed Description

**Involved Components and their required Functionality**

1. ClientLib
  - (a) Can connect to daemon
  - (b) Can reserve memory for connection
2. McDriverDaemon
  - (a) Daemon running
  - (b) Daemon socket accessible by TLC
  - (c) mcOpenDevice()

### 5.33.2 Function Documentation

#### 5.33.2.1 TestCase\_GC00100\_PlugOut\_and\_Plugin\_Cpu1 ( )

**Description:**

Mobicore caused other secondary cpu boots to fail when the process runs on Cpu1 this test has been developed to make sure if a process over Cpu1 shuts down another secondary cpu can hotplug the core back up again. Important note: Since Power Management (PM) aggressively takes the un-required cores down this test results as passed when PM is enabled, yet the test is not run actually.

**Precondition**

Running linux kernel

**Focus:**

fread returns 1 for Cpu2 from /sys/devices/system/cpu/cpu2/online

**Verification:**

read ingredients of /sys/devices/system/cpu/cpu2/online after plugging out and plugging in.

**Postcondition****Test ID:**

Runtime\_00600\_CpuCoreAvailability\_GC00100\_PlugOut\_and\_Plugin\_Cpu1

### 5.34 Concurrency Tests

### 5.35 Trustlet Communication Tests

### 5.36 Trustlet API (TIApi)

These tests contain good cases and bad cases for the API functions.

**Modules**

- [Inter-world communication functions \[COM\]](#).
- [Security support functions \[SEC\]](#).
- [MobiCore system functions \[SYS\]](#).

### 5.36.1 Detailed Description

These tests contain good cases and bad cases for the API functions. Some Trustlet API functions are also tested implicitly by other test subjects like *Trustlet Runtime* and *Performance*.

#### Introduction

Tests checking the basic functionality of the Trustlet interface.

## 5.37 MCD\_MCDIMPL\_DAEMON

### Files

- file [MobiCoreDriverCmd.h](#)  
*Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1.*

### Defines

- #define [MC\\_DEVICE\\_ID\\_DEFAULT](#) 0  
*The default device ID.*

## 5.38 TLAPI\_SEC

### Files

- file [TlApiCrypto.c](#)  
*TlApi security functions.*

## 5.39 TlApi

### Data Structures

- struct [tlApiCrAbort\\_t](#)  
*Marshaled function parameters.*
- struct [marshalingParam\\_t](#)  
*Marshaled union.*

## Files

- file [TIApiMarshal.h](#)  
*Marshaling types and declarations.*

## Defines

- #define [MAX\\_MAR\\_LIST\\_LENGTH](#) 8  
*Maximum number of parameter .*

## Typedefs

- typedef void \* [void\\_ptr](#)  
*a pointer to anything.*
- typedef [void\\_ptr](#) [addr\\_t](#)  
*an address, can be physical or virtual*
- typedef [addr\\_t](#) [buffer\\_t](#)  
*a data buffer*

## Enumerations

- enum [kpdFuncID\\_t](#) {  
    [FID\\_KPD\\_GRAB\\_KEYPAD](#) = 0,  
    [FID\\_KPD\\_GET\\_C](#),  
    [FID\\_KPD\\_GET\\_MULTIPLE\\_C](#),  
    [FID\\_KPD\\_RELEASE\\_KEYPAD](#) }  
*Each function must have its own ID.*
- enum [cryptoFuncID\\_t](#) {  
    [FID\\_CRYPT\\_MD](#) = 0,  
    [FID\\_CRYPT\\_MD\\_INIT](#) = 1,  
    [FID\\_CRYPT\\_MD\\_INIT\\_WITH\\_DATA](#) = 2,  
    [FID\\_CRYPT\\_MD\\_UPDATE](#) = 3,  
    [FID\\_CRYPT\\_MD\\_DOFINAL](#) = 4,  
    [FID\\_CRYPT\\_SIG](#) = 5,  
    [FID\\_CRYPT\\_SIG\\_INIT](#) = 6,  
    [FID\\_CRYPT\\_SIG\\_INIT\\_WITH\\_DATA](#) = 7,

```

FID_CRYPT0_SIG_UPDATE = 8,
FID_CRYPT0_SIG_SIGN = 9,
FID_CRYPT0_SIG_VERIFY = 10,
FID_CRYPT0_RNG = 11,
FID_CRYPT0_RNG_GENERATE_DATA = 12,
FID_CRYPT0_CIPHER = 13,
FID_CRYPT0_CIPHER_INIT = 14,
FID_CRYPT0_CIPHER_INIT_WITH_DATA = 15,
FID_CRYPT0_CIPHER_UPDATE = 16,
FID_CRYPT0_CIPHER_DOFINAL = 17,
FID_CRYPT0_SESSION_ABORT = 18,
FID_CRYPT0_GENERATE_KEY_PAIR = 19,
FID_CRYPT0_GENERATE_KEY_PAIR_BUFFER = 20,
FID_SECURITY_WRAP_OBJECT = 21,
FID_SECURITY_UNWRAP_OBJECT = 22,
FID_SECURITY_GET_SUID = 23,
FID_SECURITY_IS_DEVICE_BOUND = 24,
FID_SECURITY_BIND_DEVICE = 25,
FID_SYSTEM_GET_VERSION = 26,
FID_SECURITY_GET_TIME_STAMP = 27 }

```

*Each function must have its own ID.*

## Functions

- `_TLAPI_EXTERN_C tlApiResult_t callCryptoDriver (const marshalingParam_ptr pMarParam)`

*Forwards call to Trustlet.*

### 5.39.1 Define Documentation

#### 5.39.1.1 `#define MAX_MAR_LIST_LENGTH 8`

Maximum number of parameter .

Maximum list of possible marshaling parameters.

### 5.39.2 Typedef Documentation

#### 5.39.2.1 `typedef void* void_ptr`

a pointer to anything.

### 5.39.3 Enumeration Type Documentation

#### 5.39.3.1 enum kpdFuncID\_t

Each function must have its own ID.

Extend this list if you add a new function.

Enumerator:

***FID\_KPD\_GRAB\_KEYPAD*** Function to reserve the keypad.

***FID\_KPD\_GET\_C*** Function to read a character from the keypad.

***FID\_KPD\_GET\_MULTIPLE\_C*** Function to read multiple characters from the keypad.

***FID\_KPD\_RELEASE\_KEYPAD*** Function to release the grabbed keypad.

#### 5.39.3.2 enum cryptoFuncID\_t

Each function must have its own ID.

Extend this list if you add a new function.

Enumerator:

***FID\_CRYPTO\_MD*** ID for message digest algorithms.

***FID\_CRYPTO\_MD\_INIT*** Function to init a message digest.

***FID\_CRYPTO\_MD\_INIT\_WITH\_DATA*** Function to init a message digest with data.

***FID\_CRYPTO\_MD\_UPDATE*** Function to update a message digest.

***FID\_CRYPTO\_MD\_DOFINAL*** Function to finalize a message digest.

***FID\_CRYPTO\_SIG*** ID for signature algorithms.

***FID\_CRYPTO\_SIG\_INIT*** Function to init a signature.

***FID\_CRYPTO\_SIG\_INIT\_WITH\_DATA*** Function to init a signature with data.

***FID\_CRYPTO\_SIG\_UPDATE*** Function to update a signature.

***FID\_CRYPTO\_SIG\_SIGN*** Function to make a signature.

***FID\_CRYPTO\_SIG\_VERIFY*** Function to verify a signature.

***FID\_CRYPTO\_RNG*** ID for RNG algorithms.

***FID\_CRYPTO\_RNG\_GENERATE\_DATA*** Function to generate random data.

***FID\_CRYPTO\_CIPHER*** ID for cipher algorithms.

***FID\_CRYPTO\_CIPHER\_INIT*** Function to init a cipher.

***FID\_CRYPTO\_CIPHER\_INIT\_WITH\_DATA*** Function to init a cipher with data.

***FID\_CRYPTO\_CIPHER\_UPDATE*** Function to update a cipher.

***FID\_CRYPTO\_CIPHER\_DOFINAL*** Function to finalize a cipher.

***FID\_CRYPTO\_SESSION\_ABORT*** Function to abort a crypto session.

***FID\_CRYPTO\_GENERATE\_KEY\_PAIR*** Function to generate a key pair.



**FID\_CRYPTO\_GENERATE\_KEY\_PAIR\_BUFFER** Function to generate a key pair into a buffer.

**FID\_SECURITY\_WRAP\_OBJECT** Function to wrap given data and create a secure object.

**FID\_SECURITY\_UNWRAP\_OBJECT** Function to unwrap given secure object and create plaintext data.

**FID\_SECURITY\_GET\_SUID** Function to acquire System on chip Unique ID.

**FID\_SECURITY\_IS\_DEVICE\_BOUND** Unused.

**FID\_SECURITY\_BIND\_DEVICE** Unused.

**FID\_SYSTEM\_GET\_VERSION** Function to get information about the underlying MobiCore version.

**FID\_SECURITY\_GET\_TIME\_STAMP** Function to get a secure time stamp.

#### 5.39.4 Function Documentation

##### 5.39.4.1 `_TLAPI_EXTERN_C tlApiResult_t callCryptoDriver ( const marshalingParam_ptr pMarParam )`

Forwards call to Trustlet.

.Copies parameters from stack to shared memory .Calls Trustlet and returns response

##### Parameters

<code>pMarParam</code>	pointer to parameters marshalled in a struct	1whitetableShadegray
------------------------	--	----------------------

##### Returns

response of Crypto Driver to this operation.



## Chapter 6

# Data Structure Documentation

### 6.1 marshalingParam\_t Struct Reference

Marshaled union.

```
#include <TlApiMarshal.h>
```

#### Data Fields

- uint32\_t [functionId](#)  
*Function identifier.*

#### 6.1.1 Detailed Description

Marshaled union.

#### 6.1.2 Field Documentation

##### 6.1.2.1 uint32\_t marshalingParam\_t::functionId

Function identifier.

The documentation for this struct was generated from the following file:

- [TlApiMarshal.h](#)

### 6.2 TestUtils::BypassDaemon::notification\_t Struct Reference

Notification data structure.

## Data Fields

- uint32\_t [sessionId](#)  
*Session ID.*
- int32\_t [payload](#)  
*Additional notification information.*

### 6.2.1 Detailed Description

Notification data structure.

### 6.2.2 Field Documentation

#### 6.2.2.1 uint32\_t TestUtils::BypassDaemon::notification\_t::sessionId

Session ID.

#### 6.2.2.2 int32\_t TestUtils::BypassDaemon::notification\_t::payload

Additional notification information.

The documentation for this struct was generated from the following file:

- TestUtils.cpp

## 6.3 tIApiCrAbort\_t Struct Reference

Marshaled function parameters.

```
#include <TIApiMarshal.h>
```

### 6.3.1 Detailed Description

Marshaled function parameters. structs and union of marshaling parameters via TIApi.

#### Note

The structs can NEVER be packed !  
The structs can NOT used via sizeof(..) !

The documentation for this struct was generated from the following file:

- [TIApiMarshal.h](#)

## Chapter 7

# File Documentation

### 7.1 MobiCoreDriverCmd.h File Reference

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1.

```
#include <inttypes.h>
#include "mcUuid.h"
#include "mcVersionInfo.h"
```

#### Defines

- #define [MC\\_DEVICE\\_ID\\_DEFAULT](#) 0  
*The default device ID.*

#### 7.1.1 Detailed Description

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAM-

AGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 7.2 TlApiCrypto.c File Reference

TlApi security functions.

```
#include <stdint.h>
#include <stdio.h>
#include "TlApiMarshal.h"
#include "TlApi/TlApiCrypto.h"
#include "mcDriverId.h"
```

### Enumerations

- enum [bool](#)

*Marshalling of Trustlet API crypto Function calls are mapped to a flat call.*

### 7.2.1 Detailed Description

TlApi security functions.

## 7.3 TlApiMarshal.h File Reference

Marshaling types and declarations.

```
#include <stdint.h>
#include "TlApi/TlApiError.h"
#include "TlApi/TlApiCrypto.h"
#include "TlApi/TlApiSecurity.h"
#include "TlApi/TlApiMcSystem.h"
```

### Data Structures

- struct [tlApiCrAbort\\_t](#)

*Marshaled function parameters.*

- struct [marshalingParam\\_t](#)

*Marshaled union.*

## Defines

- #define [MAX\\_MAR\\_LIST\\_LENGTH](#) 8

*Maximum number of parameter .*

## Typedefs

- typedef void \* [void\\_ptr](#)  
*a pointer to anything.*
- typedef [void\\_ptr](#) [addr\\_t](#)  
*an address, can be physical or virtual*
- typedef [addr\\_t](#) [buffer\\_t](#)  
*a data buffer*

## Enumerations

- enum [kpdFuncID\\_t](#) {  
    [FID\\_KPD\\_GRAB\\_KEYPAD](#) = 0,  
    [FID\\_KPD\\_GET\\_C](#),  
    [FID\\_KPD\\_GET\\_MULTIPLE\\_C](#),  
    [FID\\_KPD\\_RELEASE\\_KEYPAD](#) }  
*Each function must have its own ID.*
- enum [cryptoFuncID\\_t](#) {  
    [FID\\_CRYPT\\_MD](#) = 0,  
    [FID\\_CRYPT\\_MD\\_INIT](#) = 1,  
    [FID\\_CRYPT\\_MD\\_INIT\\_WITH\\_DATA](#) = 2,  
    [FID\\_CRYPT\\_MD\\_UPDATE](#) = 3,  
    [FID\\_CRYPT\\_MD\\_DOFINAL](#) = 4,  
    [FID\\_CRYPT\\_SIG](#) = 5,  
    [FID\\_CRYPT\\_SIG\\_INIT](#) = 6,  
    [FID\\_CRYPT\\_SIG\\_INIT\\_WITH\\_DATA](#) = 7,  
    [FID\\_CRYPT\\_SIG\\_UPDATE](#) = 8,

```

FID_CRYPT0_SIG_SIGN = 9,
FID_CRYPT0_SIG_VERIFY = 10,
FID_CRYPT0_RNG = 11,
FID_CRYPT0_RNG_GENERATE_DATA = 12,
FID_CRYPT0_CIPHER = 13,
FID_CRYPT0_CIPHER_INIT = 14,
FID_CRYPT0_CIPHER_INIT_WITH_DATA = 15,
FID_CRYPT0_CIPHER_UPDATE = 16,
FID_CRYPT0_CIPHER_DOFINAL = 17,
FID_CRYPT0_SESSION_ABORT = 18,
FID_CRYPT0_GENERATE_KEY_PAIR = 19,
FID_CRYPT0_GENERATE_KEY_PAIR_BUFFER = 20,
FID_SECURITY_WRAP_OBJECT = 21,
FID_SECURITY_UNWRAP_OBJECT = 22,
FID_SECURITY_GET_SUID = 23,
FID_SECURITY_IS_DEVICE_BOUND = 24,
FID_SECURITY_BIND_DEVICE = 25,
FID_SYSTEM_GET_VERSION = 26,
FID_SECURITY_GET_TIME_STAMP = 27 }

```

*Each function must have its own ID.*

## Functions

- `_TLAPI_EXTERN_C tIApiResult_t callCryptoDriver (const marshalingParam_ptr pMarParam)`

*Forwards call to Trustlet.*

### 7.3.1 Detailed Description

Marshaling types and declarations. Functions for the marshaling of function ID and parameters.