TCG Mobile Reference Architecture v1.0 Revision 5 : Normative Requirements Check-list May 28th 2009

About This Document

1. This is an informative document, intended to catalogue the normative requirements of the "TCG Mobile Reference Architecture."
   Please note that in case of any discrepancies, the Mobile Reference Architecture specification is definitive, and overrides the information contained herein.

2. The check-list is provided for developers' own assurance purposes, or to assist in their private communications with customers and suppliers.
   This document is not related to any TCG Certification Program.

TCG Mobile Reference Architecture v1.0 Revision 5 : Normative Requirements Check-list					May 28th 2009		
Section	**Page**	**Id**	**Conditional on Other Requirement?**	**Requirement text**	**Subject**	**MUST/ SHOULD/ MAY**	**Applies to Ecosystem but not Device?**
2.2.1	12	1		Each structure MUST use big endian bit ordering, which follows the Internet standard and requires that the low-order bit appear to the far right of a word, buffer, wire format, or other area and the high-order bit appear to the far left.	Representation of Information	MUST	
2.2.2	12	2		All structures MUST be packed on a byte boundary.	Representation of Information	MUST	
2.4	14	3		Certain platform components MUST be provisioned or controlled by external parties meeting ecosystem requirements, and in such cases, each platform component MUST correctly authenticate the relevant controlling party before accepting provisioning or updates.	Overview	MUST	
4.1.1	18	4		If a superior engine provides resources to a subordinate engine, and the superior engine is working normally, those resources MUST conform to their published properties.	Architecture Overview	MUST	
4.1.1	18	5		If a superior engine provides resources to implement Protected Capabilities and/or Shielded Locations in a subordinate engine, and the superior engine is working normally, those resources MUST be compatible with the properties of Protected Capabilities and Shielded Locations defined in the TPM Main Specification Part I section 3.	Architecture Overview	MUST	
4.1.1	20	6		Engines in the DM_mandatoryEngineList MUST provide services subject to regulatory enforcement, MUST NOT provide indispensable services that are not subject to regulatory enforcement.	Architecture Overview	MUST	
4.1.1	20	7		Engines in the DO_mandatoryEngineList MUST NOT provide services subject to regulatory enforcement.	Architecture Overview	MUST	
4.1.1	20	8		Engines in the DM_mandatoryEngineList and DO_mandatoryEngineList MUST NOT facilitate interference by local operators with their service's access to TCG functionality.	Architecture Overview	MUST	
4.1.1	20	9		Each Engine in the DM_mandatoryEngineList and DO_mandatoryEngineList MUST have a Mobile Remote-owner Trusted Module (MRTM), whose Owner is the stakeholder of that engine.	Architecture Overview	MUST	
4.1.1	20	10		Engines in the DO_discretionaryEngineList MUST NOT provide services subject to regulatory enforcement, and MUST NOT provide non-regulatory indispensable services.	Architecture Overview	MUST	
4.1.1	20	11		Each Engine in the DO_discretionaryEngineList MUST have a Mobile Local-owner Trusted Module (MLTM), whose Owner is the stakeholder of that engine.	Architecture Overview	MUST	
4.1.1	21	12		The Device Manufacturer MUST have ultimate control over the list of engines in the DM_mandatoryEngineList.	Architecture Overview	MUST	
4.1.1	21	13		The Device Owner MUST have ultimate control over the list of engines in the DO_mandatoryEngineList. This control mechanism MUST be separate from that used to control the DO_discretionaryEngineList.	Architecture Overview	MUST	
4.1.1	21	14		The Device Owner MUST have ultimate control over the list of engines in the DO_discretionaryEngineList. This control mechanism MUST be separate from that used to control the DO_mandatoryEngineList.	Architecture Overview	MUST	
4.1.1	21	15		Engines MUST appear in exactly one of the DM_mandatoryEngineList, the DO_mandatoryEngineList, or the DO_discretionaryEngineList.	Architecture Overview	MUST	
4.1.1	21	16		One engine on the DM_mandatoryEngineList MUST be the Device Manufacturer's (DM's) own engine.	Architecture Overview	MUST	
4.1.1	21	17		The Device Manufacturer's engine MUST permit engines in the DM_mandatoryEngineList, DO_mandatoryEngineList, and DO_discretionaryEngineList to communicate with other engines in the platform and access generic resources provided by the DM's engine.	Architecture Overview	MUST	
4.1.1	21	18		Engines in the DM_mandatoryEngineList MUST be booted.	Architecture Overview	MUST	
4.1.1	21	19		Engines in the DO_mandatoryEngineList and DO_discretionaryEngineList should be booted, and failure to boot them MUST be treated as a serious error. In the event of such an error, the DM's engine MUST take appropriate manufacturer-specific action.	Architecture Overview	MUST	
4.1.1	22	20		If a dedicated RoT can be changed, it MUST verify evidence of sufficient privilege to perform an alteration (before permitting the alteration).	Architecture Overview	MUST	

4.1.1	23	21		If some resources are allocated, at least one dedicated resource MUST be responsible for ensuring that mandatory allocated resources are instantiated.	Architecture Overview	MUST	
4.1.1	23	22		Every stand-alone set of dedicated RoTs MUST have either: an Endorsement Key, Endorsement Credential, Platform Credential, and Conformance Credential, or: an Attestation Identity Key and AIK Credential as evidence that they are genuine.	Architecture Overview	MUST	
4.1.1	25	23		If a trusted engine boots, it MUST always use an authenticated boot mechanism, so integrity metrics are available once the platform has booted.	Architecture Overview	MUST	
4.1.1	25	24		All software executed on a secure-boot engine before the engine can protect existing processes from new processes, and vice versa, MUST be authenticated before execution.	Architecture Overview	MUST	
4.1.1	26	25		External certificates MUST be used when corresponding internal certificates are unavailable: when a secure-boot engine boots for the first time and has not yet reached the stage when it can execute a RIM conversion agent, for example.	Architecture Overview	MUST	
4.1.1	26	26		Each RTS in a secure-boot engine MUST have access to at least one persistent key that is the root of a key hierarchy for verifying a RIM_Auth certificate hierarchy.	Architecture Overview	MUST	
4.1.1	26	27		Each key in the key hierarchy MUST [if used at all] be used to verify other keys or a RIM certificate (though there MAY be unused leaf keys).	Architecture Overview	MUST	
4.1.1	26	28		The root key MUST be either immutable or MUST be used to verify its replacements.	Architecture Overview	MUST	
4.1.1	26	29		The RTS MUST store either the root key or a means to recognize the root key (its digest, for example).	Architecture Overview	MUST	
4.1.2	28	30		RoTs MUST be dedicated RoTs or allocated RoTs or services exported by another engine.	Reference Engine	MUST	
4.1.2	28	31		At least one RoT somewhere in a mobile trusted platform MUST be a dedicated instantiation with a preassigned EK or AIK because there will be nothing on the platform to measure it.	Reference Engine	MUST	
4.1.2	28	32		These RoTs MUST be instantiated and supplied in a way that guarantees that their EK or AIK remains secret.	Reference Engine	MUST	
4.1.2	28	33		The Internal Trusted Services MUST be instantiated either by resources belonging to the engine or by Trusted Services from another engine.	Reference Engine	MUST	
4.1.2	29	34		The Normal Resources MUST be instantiated either by resources belonging to the engine or by Normal Services exported by another engine and supplied with measurements and /or certificates of their trustworthiness.	Reference Engine	MUST	
4.1.2	29	35		Internal Trusted Services MUST be isolated from Normal Services.	Reference Engine	MUST	
4.1.3	30	36		For uniformity of reporting, measurements of the RTE, RTM, RTV, RTS and RTR MUST always be stored in the RTS.	Roots of Trust	MUST	
4.1.3	30	37		If any of the RTM, RTV and RTS in an engine are to be built from allocated resources, they MUST be built by trusted resources.	Roots of Trust	MUST	
4.1.3	30	38		RoTs composed of dedicated resources MUST perform a self-test before starting normal operation, and MUST shut-down if the test fails.	Roots of Trust	MUST	
4.1.3	30	39		If an Engine builds any Roots of Trust from allocated resources, the Engine MUST have an RTE.	Roots of Trust	MUST	
4.1.3	30	40		The RTE SHALL build all Roots-of-Trust of its engine that are based on allocated resources.	Roots of Trust	MUST	
4.1.3	30	41		At least part of an RTE MUST consist of dedicated resources.	Roots of Trust	MUST	
4.1.3	30	42		An RTE therefore MUST be supplied with an EK and / or AIK plus relevant credentials.	Roots of Trust	MUST	
4.1.3	30	43		The RTE may be immutable or may be mutable, but its integrity MUST always be intact: the RTEs integrity and authenticity MUST be maintained during the lifecycle of the platform.	Roots of Trust	MUST	
4.1.3	31	44	{{Customization by engine stakeholder}}	In this case, the RTE MUST contain a list of the services and resources that the stakeholder dictates.	Roots of Trust	MUST	
4.1.3	31	45		The RTE's supplier MUST ensure that the RTE's authenticity and integrity are preserved when the RTE is supplied and / or changed.	Roots of Trust	MUST	Yes
4.1.3	31	46		The methods of supplying and changing the RTE are outside the scope of this specification but replacement or modification MUST be performed only by an agent and method approved by the RTE supplier.	Roots of Trust	MUST	Yes
4.1.3	31	47		If the RTE changes during the lifecycle then the supplier MUST make sure that no rollback attacks can occur after an RTE has been upgraded.	Roots of Trust	MUST	Yes

4.1.3	31	48		Recustomisation of the RTE is outside the scope of this specification but the RTE's supplier MUST provide means that preserve the control of the RTE by the engine's stakeholder.	Roots of Trust	MUST	Yes
4.1.3	31	49		An engine's RTS and RTR MUST comply with the requirements of the specification "TPM Main Part 1 Design Principles" Section 3 and all sub-sections ("Protection"). This describes Protected Capabilities and Shielded Locations.	Roots of Trust	MUST	
4.1.3	31	50		A RTM MUST accurately measure the first software that is executed on the platform and MUST reliably record the result in the RTS.	Roots of Trust	MUST	
4.1.3	31	51		A Device Manufacturer's Engine MUST incorporate a RTM, RTS, RTV and RTR.	Roots of Trust	MUST	
4.1.3	31	52		Other Engines MUST support at minimum the RTM, RTR and RTS.	Roots of Trust	MUST	
4.1.3	31	53		For each Engine (with the exceptional case of an engine built entirely from dedicated resources), there MUST be one or more measurement events, and where the RTV exists, verification events.	Roots of Trust	MUST	
4.1.4	32	54		A trusted mobile phone MUST NOT implement means of controlling a mandatory engine via Physical Presence.	Physical Presence	MUST	
4.1.4	32	55		A trusted mobile phone MUST implement means of controlling a discretionary engine via Physical Presence.	Physical Presence	MUST	
4.1.4	32	56		A manufacturer may implement Physical Presence in any way but all indications of Physical Presence MUST accurately represent detection of appropriate physical interactions with the platform.	Physical Presence	MUST	
4.2.2	34	57		The Mobile Trusted Module contains the RTS and the RTR, and MUST be as defined in "TCG Mobile Trusted Module Specification"	Interfaces to the Roots of Trust	MUST	
5	38	58		The DM's Engine MUST support an RTV	Measurement and Verification	MUST	
5	38	59	{{User Engine supporting an RTV}}	In that case the local owner SHALL have full control over what measurements within the engine are verified, and what are considered "correct" values of those measurements.	Measurement and Verification	MUST	
5	38	60		All engines MUST support an RTM, RTS, RTR and a transitive chain of measurement, to provide an authenticated boot.	Measurement and Verification	MUST	
5.1.2	39	61		The RTV MUST have access to an integrity protected list of RIMs, contained in RIM Certificates.	RTV	MUST	
5.1.2	40	62		The RTV MUST either pass the responsibility for verifying measurements to other verification agents, or be the sole verification agent throughout the boot process.	RTV	MUST	
5.1.2	40	63		Requirements in 5.1.3 concerning configuration, upgrades and customization of the RTV also apply to other verification agents: however these requirements for other verification agents SHALL be met using a transitive chain of trust from the RTV (see Section 5.4).	RTV	MUST	
5.1.2	40	64		The RTV SHALL be resistant to all forms of software attack and to the forms of physical attack implied by the platform's Protection Profile.	RTV	MUST	
5.1.2	40	65		The RTV SHALL supply an accurate report of the availability of the corresponding Reference Integrity Metrics (RIMs).	RTV	MUST	
5.1.2	40	66		The RTV SHALL supply an accurate report of the relationship (equal or not equal) between the measurements and corresponding Reference Integrity Metrics (RIMs).	RTV	MUST	
5.1.2	41	67		Upon verification failure, the RTV SHALL either trigger the transition of the Engine to a FAILED state or where the entity verified was not a mandatory function, trigger an alternative execution path. The RTV MUST NOT continue the transitive trust boot process in the same manner as if there is no failure.	RTV	MUST	
5.1.3	41	68		The Root-of-Trust-for-Verification of an engine SHALL be provisioned by the stakeholder of that engine.	RTV	MUST	Yes
5.1.3	41	69		The supplier of the Root-of-Trust-for-Verification SHALL be responsible for the security of the provisioning process, i.e., the supplier MUST make sure that authenticity and integrity of the Root-of-Trust-for-Verification are preserved.	RTV	MUST	Yes
5.1.3.1	41	70		The supplier also SHALL be responsible for secure upgrades of the Root-of-Trust-for-Verification, i.e., for the authenticity and integrity of the provisioning of a new version of it and its credentials.	RTV	MUST	Yes
5.1.3.1	41	71		The supplier MUST make sure that no rollback attacks can occur after the Root-of-Trust-for-Verification has been upgraded.	RTV	MUST	Yes
5.1.3.2	41	72		The Root-of-Trust-for-Verification SHALL be customized by the engine's stakeholder in order to dictate which services have to be measured and against which RIMs the measurements have to be verified	RTV	MUST	Yes

5.1.3.2	41	73	{{Stakeholder recustomization}}	In any case, the stakeholder SHALL be responsible for preserving authenticity and integrity of this process.	RTV	MUST	Yes
5.1.3.2	41	74	{{Stakeholder recustomization}}	The stakeholder MUST make sure that no rollback attacks can occur after the list of measured services or the list of RIMs and RIM Certificates have been changed.	RTV	MUST	Yes
5.2	42	75		The Engine MUST be able to correctly establish the source of the RIMs being provisioned to the Engine.	Secure Provisioning of RIMs	MUST	
5.2	42	76		Further, having established the source, the Engine MUST be able to decide whether that source is authorized to supply RIMs.	Secure Provisioning of RIMs	MUST	
5.2	42	77		In addition, it MUST be able to determine that the RIMs have not been corrupted since leaving that source.	Secure Provisioning of RIMs	MUST	
5.2	42	78		The Engine MUST be able to determine if its installed RIMs have been completely removed by an attacker and replaced by an unauthorized set.	Secure Provisioning of RIMs	MUST	
5.2	42	79		The Engine MUST be able to determine that RIMs being provisioned by the source are newer than RIMs already installed on the Device.	Secure Provisioning of RIMs	MUST	
5.2	42	80		The Engine MUST be able to determine if its installed RIMs have been replaced by a set that was once valid, but older than the replaced set.	Secure Provisioning of RIMs	MUST	
5.2	42	81		The Engine MUST be able to determine that RIMs being provided by the Source are still considered to be valid by the Source.	Secure Provisioning of RIMs	MUST	
5.2	43	82		Any alternative [RIM provisioning] methods MUST have security properties at least as strong as the method defined below, and MUST NOT prevent the defined method operating alongside them.	Secure Provisioning of RIMs	MUST	
5.2.1	43	83	{{Support for recommended RIM provisioning method}}	Each Engine has a pre-configured public key called the Root Verification Authority Identifier (RVAI). This key MUST be integrity protected using shielded storage. The RVAI is a verification root key as defined in "TCG Mobile Trusted Module Specification"	RIM_Auths	MUST	
5.2.1	43	84	{{Support for recommended RIM provisioning method}}	However, for a general verification agent, the Root Verification Authority SHALL be able to delegate to other authorities.	RIM_Auths	MUST	Yes
5.2.1	43	85	{{Support for recommended RIM provisioning method}}	The structure TPM_Verification_Key defined in "TCG Mobile Trusted Module Specification" … MUST be used where the RIM_Auth_Certs need to be verified using a MTM.	RIM_Auths	MUST	
5.2.1	43	86	{{Support for recommended RIM provisioning method}}	A public key signature SHALL be provided as the proprietary authData.	RIM_Auths	MUST	
5.2.1	43	87	{{Support for recommended RIM provisioning method}}	Each RIM_Auth_Cert MUST contain at least the following information; this information is automatically contained if the structure TPM_Verification_Key is used: An Identifier for the Issuer Keys and Subject Keys of this Certificate; Flags indicating whether this RIM_Auth can sign RIM_Certs directly and/or can sign delegation structures for other RIM_Auths and/or can revoke what it has signed; The RIM_Auth's public key; The Signature of the private key that issued this RIM_Auth_Cert	RIM_Auths	MUST	
5.2.2	44	88	{{Support for recommended RIM provisioning method, and RIM_Auth Validity Lists}}	Every RIM_Auth which signs Validity Lists MUST ensure that it always has signed a Validity List whose "valid from" and "valid to" fields in UTCtime format enclose the current date and time.	RIM_Auth Validity Lists	MUST	Yes
5.2.2	44	89	{{Support for recommended RIM provisioning method}}	Whether a RIM_Auth signs RIM_Auth Validity Lists or not MUST be indicated by a key-usage flag in the TPM_Verification_Key structure (see "TCG Mobile Trusted Module Specification").	RIM_Auth Validity Lists	MUST	Yes
5.2.3	45	90	{{Support for recommended RIM provisioning method}}	Each Engine of the Device MUST have available the following root authorization data in an integrity-protected form (this may be stored or may be provided externally to the Engine e.g. over a network interface) : All the RIM_Auth_Certs associated with RIM_Certs that the Engine is currently using, and that are signed directly by the RVAI; If the RVAI key provides revocation information, then the most recent revocation information the Engine has been shown, that is signed by the RVAI private key	Storage and use of RIM_Auth Certs and RIM_Auth Validity Lists	MUST	

5.2.3	46	91	{{Support for recommended RIM provisioning method}}	In addition, each Engine of the Device MUST have available a full tree of authorization data in an integrity-protected form : Every RIM_Auth_Cert associated with RIM_Certs that the Engine is currently using, that presents a certificate chain up to the RVAI; For each RIM_Auth CA that provides revocation information, then the most recent revocation information the Engine has been shown. This MUST be a current Validity List if the RIM_Auth CA signs Validity Lists.	Storage and use of RIM_Auth Certs and RIM_Auth Validity Lists	MUST	
5.2.3	46	92	{{Support for recommended RIM provisioning method}}	If stored in the Engine, such authorization data MUST be updateable only by an authorized process.	Storage and use of RIM_Auth Certs and RIM_Auth Validity Lists	MUST	
5.2.3	46	93	{{Support for recommended RIM provisioning method}}	The above information MUST be available to the Engine for at least the following uses: Authorization of new RIM_Certs; Remote Attestation	Storage and use of RIM_Auth Certs and RIM_Auth Validity Lists	MUST	
5.2.3	46	94	{{Support for recommended RIM provisioning method}}	The Engine MUST be able to determine for a given external RIM_Cert whether the certificate was signed correctly using a RIM_Auth's public key.	Storage and use of RIM_Auth Certs and RIM_Auth Validity Lists	MUST	
5.2.3	46	95	{{Support for recommended RIM provisioning method}}	The Engine MUST also be able to determine whether the RIM_Auth was authorized to sign that sort of RIM Certificate (i.e. has a correct delegation chain with nothing revoked on that chain, and the RIM_Auth_Cert has the correct – optional - PCR and label settings)	Storage and use of RIM_Auth Certs and RIM_Auth Validity Lists	MUST	
5.2.3	46	96	{{Support for recommended RIM provisioning method}}	If any lists restricting key usage are present and bound to a RIM_Auth_Cert, then the Engine MUST respect the relevant restrictions on the RIM_Auth's authority. If any such lists are not present, then the Engine MUST assume that the RIM_Auth has no restrictions in the relevant aspect(s).	Storage and use of RIM_Auth Certs and RIM_Auth Validity Lists	MUST	
5.2.3	46	97	{{Support for recommended RIM provisioning method}}	If any of the RIM_Auth CAs for an Engine signs RIM_Auth Validity Lists, then the Engine MUST be able to process them.	RIM_Auth Validity Lists	MUST	
5.2.3	46	98	{{Support for recommended RIM provisioning method}}	The Engine MUST ensure whenever using Validity Lists that the information contained therein is still current, according to the most reliable clock the Engine has available. (If no clock is available then the Engine MUST just use the most recent Validity List that it has). If the Engine detects that a given RIM_Auth signs revocation information, and detects that the revocation information it has is no longer current, then the Engine MUST attempt to retrieve current revocation information using an online protocol (for example, by accessing the web-site of the relevant RIM_Auth). If the Engine cannot retrieve such information it MUST abort the current operation which relies on this information.	Storage and use of RIM_Auth Certs and RIM_Auth Validity Lists	MUST	
5.2.3	46	99	{{Support for recommended RIM provisioning method}}	The Engine MUST be able to present its current root authorization data, or full tree of authorization data, when attesting its trust state to service providers.	Storage and use of RIM_Auth Certs and RIM_Auth Validity Lists	MUST	
5.2.4	47	100	{{Support for recommended RIM provisioning method, and RIM Validity Lists}}	Every RIM_Auth which signs Validity Lists MUST ensure that it always has signed a Validity List whose "valid from" and "valid to" fields in UTCtime format enclose the current date and time.	RIM Validity Lists	MUST	Yes
5.2.4	47	101	{{Support for recommended RIM provisioning method}}	Whether a RIM_Auth signs RIM Validity Lists or not MUST be indicated by a key-usage flag in the TPM_Verification_Key structure (see "TCG Mobile Trusted Module Specification").	RIM Validity Lists	MUST	Yes
5.2.5	48	102	{{Support for recommended RIM provisioning method}}	Each Engine of the Device MUST have available the following data in an integrity-protected form: For each RIM_Auth_Cert that provides revocation information for its RIM_Certs, and which has signed RIM_Certs that the Engine is currently using, the most recent revocation information from that RIM_Auth that the Engine has been shown. (Note that if the Engine can detect that a given RIM_Auth never revokes its RIM_Certs, then nothing needs to be stored for that RIM_Auth.)	Storage and Use of RIM Validity Lists	MUST	
5.2.5	48	103	{{Support for recommended RIM provisioning method}}	If stored in the Engine, again such integrity protected storage MUST be updateable only by an authorized process.	Storage and Use of RIM Validity Lists	MUST	
5.2.5	48	104	{{Support for recommended RIM provisioning method}}	The revocation information MUST be available for at least the following uses: Revocation checking of new RIM_Certs; Preventing Replay of an old RIM Validity List	Storage and Use of RIM Validity Lists	MUST	

5.2.5	48	105	{{Support for recommended RIM provisioning method}}	The Engine MUST be able to determine whether a given external RIM Certificate has been revoked or not by the RIM_Auth.	Storage and Use of RIM Validity Lists	MUST	
5.2.5	48	106	{{Support for recommended RIM provisioning method}}	The Engine MUST be able to tell if a supplied RIM Validity List is older than the one it has currently stored.	Storage and Use of RIM Validity Lists	MUST	
5.2.5	48	107	{{Support for recommended RIM provisioning method}}	If any of the RIM_Auths for an Engine signs RIM Validity Lists, then the Engine MUST be able to process them.	RIM Validity Lists	MUST	
5.2.5	48	108	{{Support for recommended RIM provisioning method}}	The Engine MUST ensure whenever using a RIM Validity List that the information contained therein is still current.	Storage and Use of RIM Validity Lists	MUST	
5.3.1	49	109		Each Engine supplier MUST allocate PCRs within the Engine's MTM consistently.	Measurement of Platform Behaviour	MUST	Yes
5.3.1	49	110		Each measurement agent that needs to extend to a given MTM SHALL have exclusive access to at least one dedicated PCR.	Measurement of Platform Behaviour	MUST	
5.3.1	49	111		Thus there MUST be at least as many PCRs as concurrent measurement agents within a given Engine.	Measurement of Platform Behaviour	MUST	
5.3.1	49	112		Where more than one RIM_Auth is entitled to provide RIMs to be verified in a given Engine, then the MTM SHALL contain at least one dedicated PCR for each RIM_Auth.	Measurement of Platform Behaviour	MUST	
5.3.1	49	113		Thus there MUST be at least as many PCRs as RIM_Auths for a given Engine.	Measurement of Platform Behaviour	MUST	
5.3.1	50	114		The DM engine's MTM MUST have at least 16 PCR registers (this is the same number of PCR registers as defined in [TPM Main Part 2 TPM Structures] for TPM_PERMANENT_DATA).	Measurement of Platform Behaviour	MUST	
5.4	52	115		Before completion of execution, the RTM MUST measure (and the RTV MUST verify) the executable load image of at least one other measurement agent (and at least one associated verification agent).	Transitive Chain of Trust for Measurement and Verification Agents	MUST	
5.4	52	116		Any verification failure for a mandatory function MUST trigger the transition of the Engine to a FAILED state.	Transitive Chain of Trust for Measurement and Verification Agents	MUST	
5.4	52	117		Any measurement agent which is a parent to a mandatory function MUST also be, or be associated with, a verification agent.	Transitive Chain of Trust for Measurement and Verification Agents	MUST	
5.4	52	118		A measurement or verification agent SHALL NOT be a parent to a mandatory function (such as a MTM or TSS) when the agent already needs to use that function (to store PCRs etc.)	Transitive Chain of Trust for Measurement and Verification Agents	MUST	
5.5	53	119		Each measurement agent SHALL perform measurement functions in the same way as the RTM, as defined in Sections 4.1.3 and 4.2.	Measurement agent operation at higher layer	MUST	
5.5	53	120		The measurement agent's code/configuration data SHALL implicitly or explicitly point at a list of Target Objects of measurement.	Measurement agent operation at higher layer	MUST	
5.5	53	121		Once each measurement is made, the measurement agent MUST either itself attempt a PCR extend in a MTM with which the measurement agent can communicate or, where verification is required, MUST pass the measurement to a corresponding verification agent.	Measurement agent operation at higher layer	MUST	
5.5	53	122		Each associated verification agent - if present - SHALL perform verification functions in the same way as the RTV.	Measurement agent operation at higher layer	MUST	
5.5	53	123		Requirements 1-3 for the RTV, as listed in Section 5.1.2, SHALL apply.	Measurement agent operation at higher layer	MUST	
5.5	53	124		Where verification of measurements is required, the associated verification agent MUST be able to retrieve corresponding Reference Integrity Metrics (RIMs).	Measurement agent operation at higher layer	MUST	
5.5	53	125		The verification agent MUST verify each measurement against a RIM, and if successfully verified, MUST attempt a PCR extend or verified extend in the MTM.	Measurement agent operation at higher layer	MUST	

5.5	53	126		If the verification agent detects a verification failure, or the MTM reports a failed verified extend, then this failure MUST either trigger the transition of the Engine to a FAILED state, or where the entity verified was not a mandatory function, trigger an alternative execution path (see Sections 6.3.3.2 and 7).	Measurement agent operation at higher layer	MUST	
6.1.1	55	127		The Storage Root Key (SRK) which is used as the foundation of the Root of Trust for Storage (RTS) component MUST be generated using a cryptographically strong process that meets or exceeds the requirements for the strength and equivalent security of the RTS itself (see TCG 1.0 Architecture Overview section 4.3.1.7)	Provisioning of AIK and SRK	MUST	
6.1.1	55	128		If the AIK is pre-generated and installed during manufacture time the SRK MUST be generated and installed at the same time.	Provisioning of AIK and SRK	MUST	
6.1.1.1	55	129		The RTS MUST provide integrity protection for the storage of the SRK that is capable of detecting if either the private or public key has been altered in any way.	Provisioning of AIK and SRK	MUST	
6.1.1.1	55	130		The SRK storage MUST provide protection, so that the private key is protected from disclosure to any external entity i.e. the private part MUST be stored in a shielded location.	Provisioning of AIK and SRK	MUST	
6.1.2	55+B 219	131		The Endorsement Key (EK) is … REQUIRED for locally-owned engines.	Endorsement Key	MUST	
6.1.2	55	132	{{EK exists in an engine}}	If the EK exists, then it MUST be created and bound to the engine, and MUST not be migratable.	Endorsement Key	MUST	
6.1.2	55	133	{{EK exists in an engine}}	The EK for the DM's engine (and if necessary, other engines) SHALL be generated by the device manufacturer and installed into the RTS.	Endorsement Key	MUST	Yes
6.1.2	56	134	{{EK exists in an engine}}	If the off-chip generation and installation method is used by the device manufacture, then the approach MUST provide a way to block removal of this key pair at a later time or replacing it with a new key pair once it is fielded.	Endorsement Key	MUST	
6.1.2	56	135	{{EK exists in an engine}}	If the on-chip generation option is used, the generation command MUST be disabled once the capabilities are ready for shipment.	Endorsement Key	MUST	
6.1.2	56	136	{{EK exists in an engine}}	Once the EK key pair is generated, the device manufacturer MUST build the EK credential and make it available once the device is ready to be shipped.	Endorsement Key	MUST	Yes
6.1.2	56	137	{{EK exists in an engine}}	If the engine includes the use of the EK, there are several optional MTM commands that MUST be supported (i.e. they become mandatory). These are defined in "TCG Mobile Trusted Module Specification"	Endorsement Key	MUST	
6.1.3	56	138		The Identity keys, or AIKs, and their associated certificates MUST be used by the platform to authenticate an Engine and to attest to the state of an engine.	Attestation Identity Key (AIK)	MUST	
6.1.3.1	56	139	{{EK exists in an engine}}	Where the engine supports an EK, the process of acquiring an AIK and associated certificates MUST make use of TPM_MakeIdentity and TPM_ActivateIdentity as defined in the TPM Main specification	Provisioning of AIK	MUST	
6.1.3.2	56	140	{{EK does not exist in an engine}}	If so an identity for the MTM functions SHALL be assigned by the device manufacturer.	Provisioning of AIK	MUST	Yes
6.1.3.2	56	141	{{EK does not exist in an engine}}	If the option of no EK is used, the manufacturer MUST generate and install the AIK and associated certificates during the manufacturing process of the engine and bind them to the device.	Provisioning of AIK	MUST	Yes
6.1.3.2	56	142	{{EK does not exist in an engine}}	For attestation interoperability, all pre-loaded certificates MUST be in compliance with the TCG Infrastructure credential standard "TCG Credential Profiles".	Provisioning of AIK	MUST	
6.1.3.2	56	143	{{EK does not exist in an engine}}	There is not an option for self generation of the AIK credentials, so they MUST be generated off-chip and then installed into the engine.	Provisioning of AIK	MUST	Yes
6.2.1	57	144		The Device Manufacturer's engine) MUST support the type of MTM defined as a "Mobile Remote owner Trusted Module (MRTM)"	Remote and Local Owners	MUST	
6.2.1	57	145		An engine designed to have or permit a local owner MUST support the type of MTM defined as a "Mobile Local owner Trusted Module (MLTM)"	Remote and Local Owners	MUST	
6.2.1	57	146		In the case of a remotely owned engine, a general model is that the engine's MRTM is already enabled and activated, and already has an owner set when the User takes possession of the device. This MUST be true of the Device Manufacturer's Engine, for example.	Remote and Local Owners	MUST	
6.2.1	57	147		In all cases, the remote owner MUST be protected from a User attempting to remove the remote owner's ownership of the engine, or attempting to disable or de-activate the engine's MRTM.	Remote and Local Owners	MUST	
6.2.1	57	148		In the case of a locally owned engine, a general model is that the engine's MLTM does not yet have an owner set when the User takes possession of the device; it MUST then be possible for the local User to take ownership.	Remote and Local Owners	MUST	
6.2.1	57	149		The User engine's MLTM MUST allow the local User to change the [enabled, activated] flags using assertions of Physical Presence.	Remote and Local Owners	MUST	

6.2.1	58	150		If there is already an owner set in the User's engine's MLTM (e.g. someone previously used the device), then the local User MUST be able to remove that owner using an assertion of Physical Presence (i.e. TPM_ForceClear, which MUST be supported by a MLTM).	Remote and Local Owners	MUST	
6.2.2	58	151	{{Locally owned engine supports secure boot}}	If so, the engine's MLTM MUST support the local verification commands defined in Section 7 of "TCG Mobile Trusted Module Specification". Also, the engine MUST allow the local owner to act as the stakeholder in control of that secure boot functionality.	Remote and Local Owners	MUST	
6.2.2	58	152	{{Locally owned engine supports secure boot}}	The local owner MUST be able to set the RVAI public key for the engine (see Section 6.3.1), that is to be used by the RTV and other verification agents of that engine.	Remote and Local Owners	MUST	
6.2.2	58	153	{{Locally owned engine supports secure boot}}	The local owner MUST be able to control which PCRs are set as verifiedPCRs (see Section 6.3.1.2) in that engine's MLTM .	Remote and Local Owners	MUST	
6.2.2	58	154	{{Locally owned engine supports secure boot}}	The local owner MUST have exclusive control over which RIM_Auths' TPM_Verification_Keys can be loaded into the engine's MLTM, and hence over which external RIM_Certs are accepted by the engine's MLTM.	Remote and Local Owners	MUST	
6.2.2	58	155	{{Locally owned engine supports secure boot}}	In particular, the MTM's integrityCheckRootData (see Section 6.3.1.1) MUST be set to NULL and the flag loadVerificationRootKeyEnabled MUST be set to FALSE.	Remote and Local Owners	MUST	
6.2.2	58	156	{{Locally owned engine supports secure boot}}	The local owner MUST have exclusive control over which internal RIM_Certs are generated by the engine's MLTM (see Section 6.3.4.1), and over the counterRIMProtect used to revoke such internal RIM_Certs.	Remote and Local Owners	MUST	
6.2.2	58	157	{{Locally owned engine supports secure boot. No owner set yet}}	In that case, there will be no valid internal RIM_Certs, and so the engine MUST follow a pristine boot process, as defined in Section 6.3.3.1.	Remote and Local Owners	MUST	
6.2.2	58	158	{{Locally owned engine supports secure boot. No owner set yet}}	This MUST provide a limited secure boot, just sufficient to enable the local User to securely take ownership, and establish their own secure boot control (including setting own RVAI key etc.)	Remote and Local Owners	MUST	
6.3.1	59	159		The RVAI key-pair MUST be generated by the engine's stakeholder.	Setting of RVAI and VerifiedPCRs	MUST	Yes
6.3.1	59	160		The RVAI for the device manufacturer's engine MUST be installed on the device during the platform manufacturing process,	Setting of RVAI and VerifiedPCRs	MUST	Yes
6.3.1	59	161		Once this key is installed, at any time if the RVAI (or integrity protection value) is found to have been tampered with, the engine MUST go to a "FAILED" state and block the platform from entering a "SUCCESS" state (see Section 7.1.2).	Setting of RVAI and VerifiedPCRs	MUST	
6.3.1.1	60	162	{{loadVerificationRootKeyEnabled is set to TRUE at power-up}}	The RTV MUST set the flag to FALSE (using MTM_LoadVerificationRootKeyDisable) before handing over execution control.	Setting of RVAI and VerifiedPCRs	MUST	
6.3.1.2	60	163	{{loadVerificationRootKeyEnabled is permanently set to FALSE}}	If the loadVerificationRootKeyEnabled flag is set permanently to FALSE, then the verifiedPCRs selection in the MTM Permanent Data MUST be set permanently at manufacture, for remotely owned engines, or MUST be set/reset under owner authorization, for locally owned engines (using MTM_SetVerifiedPCRSelection).	Setting of RVAI and VerifiedPCRs	MUST	
6.3.1.2	60	164	{{loadVerificationRootKeyEnabled is set to TRUE at power-up}}	If the flag is initially set to TRUE on each power-up cycle, then the RTV MUST ensure correct settings for the verifiedPCR selection at each power-up (using MTM_SetVerifiedPCRSelection).	Setting of RVAI and VerifiedPCRs	MUST	
6.3.2	60	165		Each engine which supports secure boot MUST utilize its counterBootstrap counter to verify external RIM_Certs during "pristine" boot.	Monotonic Counters	MUST	
6.3.2	60	166		The counterBoostrap counter … MUST be stored in non-volatile storage	Monotonic Counters	MUST	
6.3.2	60	167		A value of the counterBootstrap counter MUST be installed in the external version of each RIM_Cert used to verify a pristine boot	Monotonic Counters	MUST	Yes
6.3.2	60	168		The counterBootstrap value MUST be checked during pristine boot to ensure that the RIM_Cert being used is not a revoked version.	Monotonic Counters	MUST	
6.3.2	61	169		Each Engine MUST utilize a dedicated counterRIMProtect counter to verify internal RIM_Certs during "standard" boot.	Monotonic Counters	MUST	
6.3.2	61	170		The counterRIMProtect counter … MUST be stored in non-volatile storage	Monotonic Counters	MUST	
6.3.2	61	171		A value of the counterRIMProtect counter MUST be installed in the internal version of each RIM_Cert used to verify a normal boot, and MUST be checked during normal boot to ensure that the RIM_Cert being used is not a revoked version.	Monotonic Counters	MUST	

6.3.2	61	172		Each Engine SHALL support an additional counterStorageProtect monotonic counter	Monotonic Counters	MUST	
6.3.2	61	173		The counterStorageProtect counter MUST be stored in non-volatile storage	Monotonic Counters	MUST	
6.3.3.1	62	174		If there are no valid applicable internal RIM_Certs, then the DM engine MUST attempt a pristine boot.	Pristine Boot	MUST	
6.3.3.1	62	175		As for a standard boot, measurement agents MUST perform target measurements in the intended sequence as defined by their configuration data, and the resulting TIMs MUST be verified against corresponding RIMs, as defined in 6.3.3.2, but external RIM_Certs MUST be used to provide the RIMs.	Pristine Boot	MUST	
6.3.3.1	62	176		If any errors are encountered during the pristine boot process, then the Engine MUST go to a “FAILED” state (see Section 7.1.2).	Pristine Boot	MUST	
6.3.3.1	62	177		This pristine boot process MUST be completed and in a “SUCCESS” state before any “RIM Conversion Agent” (at least one per Engine) can run to do a certificate conversion (see Section 6.3.4.1), creating internal RIM_Certs ready for the next (standard) boot.	Pristine Boot	MUST	
6.3.3.1	63	178		During this conversion process, ownerAuth or verificationAuth data MUST be entered into the engine, or otherwise made available to the engine.	Pristine Boot	MUST	
6.3.3.1	63	179		In the case where verificationAuth data is stored on the platform, it MUST therefore be sealed to the expected PCR state that will exist after the pristine boot is successfully completed, but before the conversion process runs.	Pristine Boot	MUST	
6.3.3.2	63	180		During standard boot, each measurement agent (starting from the RTM) MUST perform its Target Measurements in order of execution, as defined by its measurement configuration data, and where the configuration data requires it, a corresponding verification agent MUST check the results against RIMs.	Standard Boot	MUST	
6.3.3.2	63	181		If the agents are implemented as separate functions, the measurement agent SHALL pass to the verification agent a label for the measurement, the PCR index to be extended, and the value to be extended.	Standard Boot	MUST	
6.3.3.2	63	182		The verification agent SHALL identify the correct RIM_Cert by matching the measurement label and PCR index to corresponding fields in the RIM_Cert	Standard Boot	MUST	
6.3.3.2	63	183		The verification agent SHALL check that the measured value provided by the measurement agent (and the PCR index to extend, where passed by the measurement agent) matches the expected value in the RIM_Cert.	Standard Boot	MUST	
6.3.3.2	63	184		Otherwise if the measurement does not match the RIM_Cert value the engine SHALL either transition to a “FAILED” state or attempt an alternative execution path.	Standard Boot	MUST	
6.3.3.2	63	185		The verification agent MUST also check for the condition where a target object which must be verified (according to the measurement configuration data) nevertheless has no RIM available. Then in this case the Engine MUST either transition to a “FAILED” state or attempt an alternative execution path.	Standard Boot	MUST	
6.3.3.2	63	186		The verification agent MUST also ensure that before extending, the PCRs in the MTM match the prerequisite state (see Mobile Trusted Module Specification Section 5.2) defined by the RIM_Cert... If there is a mis-match, the Engine MUST either transition to a “FAILED” state or attempt an alternative execution path.	Standard Boot	MUST	
6.3.3.2	64	187		If supported, the alternative execution path MUST evict from memory all trace of the target object that failed to verify.	Standard Boot	MUST	
6.3.3.2	64	188		If there is no alternative target object, which is the case where the original target object was a mandatory function, the engine MUST transition to a “FAILED” state. If there is an alternative target object, the boot SHALL continue with that object according to the rules specified for standard boot.	Standard Boot	MUST	
6.3.4.1	64	189		For pristine boot, an Engine MUST use external RIM_Certs directly.	When to Use External and when Internal Certs	MUST	
6.3.4.1	64	190		If the counterRIMProtect is incremented by the platform [then] any internal RIM_Certs that have a lower value SHALL be considered revoked.	When to Use External and when Internal Certs	MUST	
6.3.4.1	65	191	{{Using internal RIM_Certs previously converted from external RIM_Certs}}	Any trusted RIM Conversion Agent MUST check that both the external RIM_Certs themselves, and any RIM_Auths in chains used to sign the external RIM_Certs, have NOT been revoked before doing the conversion.	When to Use External and when Internal Certs	MUST	

6.3.4.1	65	192	{{Using internal RIM_Certs previously converted from external RIM_Certs}}	The RIM Conversion Agent MUST check that each RIM_Auth is authorized to sign each external RIM_Cert, by checking all the optional constraints on Device and Engine identifiers, PCRs, timlabels etc. in the RIM_Auth certificates.	When to Use External and when Internal Certs	MUST	
6.3.4.1	65	193	{{Using internal RIM_Certs previously converted from external RIM_Certs}}	The MTM MUST authenticate the trusted RIM Conversion Agent using verificationAuth or ownerAuth data (as defined in "TCG Mobile Trusted Module Specification").	When to Use External and when Internal Certs	MUST	
6.3.4.1	65	194	{{Using internal RIM_Certs previously converted from external RIM_Certs}}	The ownerAuth data MUST be used for locally owned engines.	When to Use External and when Internal Certs	MUST	
6.3.4.1	65	195	{{Using internal RIM_Certs previously converted from external RIM_Certs}}	The verificationAuth (or ownerAuth) MUST be used to run a RIM_Cert conversion command (MTM_InstallRIM).	When to Use External and when Internal Certs	MUST	
6.3.4.1	65	196	{{Using internal RIM_Certs previously converted from external RIM_Certs}}	In most implementations, the trusted RIM Conversion Agent will reside on the device and the verificationAuth MUST be sealed to a PCR state which ensures that only a trusted conversion agent can access that data.	When to Use External and when Internal Certs	MUST	
6.3.4.1	65	197	{{Using internal RIM_Certs previously converted from external RIM_Certs}}	Where ownerAuth data is used instead of sealed verificationAuth data, then the operation to create internal RIM_Certs MUST require owner input and control.	When to Use External and when Internal Certs	MUST	
6.3.4.1	65	198	{{Using external RIM_Certs directly during standard boot}}	However, any verification agent that is trusted to call MTM_VerifyRIMCertAndExtend with an external RIM_Cert MUST check that both the external RIM_Cert, and any RIM_Auth in the chain used to sign the external RIM_Cert, is fully authorized and has NOT been revoked before using the certificate.	When to Use External and when Internal Certs	MUST	
6.3.4.1	65	199	{{Using external RIM_Certs directly during standard boot}}	The verification agent MUST also check that each RIM_Auth is authorized to sign each external RIM_Cert, e.g. by checking all the optional constraints on Device and Engine identifiers, PCRs, labels etc. in the RIM_Auth certificates.	When to Use External and when Internal Certs	MUST	
6.3.4.2	65	200		Once a platform is fielded there are often requirements to authorize updates to the software elements that were included in the original release. The engine MUST be in a "SUCCESS" state before this update process begins.	RIM Cert Updates and Revocations	MUST	
6.3.4.2	66	201		Each RIM_Auth whose target object needs updating SHALL prepare the software update package and include a new RIM_Cert that defines that package.	RIM Cert Updates and Revocations	MUST	Yes
6.3.4.2	66	202		If the RIM_Auth signs Validity Lists, it SHALL sign a new Validity List containing the new RIM_Cert (and possibly omitting previous versions of that RIM_Cert).	RIM Cert Updates and Revocations	MUST	Yes
6.3.4.2	66	203		A RIM Conversion Agent in the Engine SHALL verify/validate the corresponding external RIM_Cert	RIM Cert Updates and Revocations	MUST	
6.3.4.2	66	204		The RIM Conversion Agent SHALL also verify/validate the RIM_Auth that signed the new RIM_Cert using a certificate chain (if applicable) back to the RVAI key.	RIM Cert Updates and Revocations	MUST	
6.3.4.2	66	205		The RIM Conversion agent MUST determine what is a valid RIM by using a full path verification process (including checking for revocation status, checking all the optional constraints on Device and Engine identifiers, PCRs, labels etc. in the RIM_Auth certificates).	RIM Cert Updates and Revocations	MUST	
6.3.4.2	66	206		The RIM Conversion Agent MUST validate that the external RIM_Cert counter value (if present) is the same or greater than the CounterBootstrap value associated with the platform, and if not, it MUST reject this new RIM_Cert.	RIM Cert Updates and Revocations	MUST	
6.3.4.2	66	207		If the certificate count is greater than the CounterBootstrap version associated with the platform then the CounterBootstrap version MUST be updated to match the new value in the certificate.	RIM Cert Updates and Revocations	MUST	
6.3.4.2	66	208		Any new Internal RIM_Cert MUST be added to the Engine's internal store of RIM_Certs.	RIM Cert Updates and Revocations	MUST	
6.3.4.3	66	209	{{Use of legacy device management to generate internal RIM Certs}}	If legacy updates are supported, the RIM Conversion Agent MUST create (or add to) the set of valid internal RIM_Certs based on those updates.	RIM Cert Updates and Revocations	MUST	

6.3.4.3	66	210	{{Use of legacy device management to generate internal RIM Certs}}	Note that where legacy Device Management is used, it MUST have the same security properties as were defined for the framework of external RIM_Certs (Source, Newness, Currency).	RIM Cert Updates and Revocations	MUST	
6.3.4.5	67	211		The platform MUST be in the "SUCCESS" state before the process of RIM revocation begins	Revoking RIMs	MUST	
6.3.4.5	67	212		The RIM Conversion Agent MUST review the revocation info to confirm that all of the external RIM_Certs that were used to create current internal RIM_Certs are still considered valid by the RIM_Auth.	Revoking RIMs	MUST	
6.3.4.5	67	213		If the RIM Conversion Agent finds an internal RIM_Cert whose external counterpart has been revoked, then it MUST remove that RIM_Cert from the Engine's internal store.	Revoking RIMs	MUST	
6.3.4.5	67	214		If there is a RIM_Auth whose cert has been revoked by the parent RIM_Auth, then the RIM Conversion Agent MUST identify all internal RIM_Certs that were authorized by that RIM_Auth and remove them all from the internal store.	Revoking RIMs	MUST	
6.3.4.5	67	215		Once all of the revoked internal RIM_Certs have been removed, then the RIM Conversion Agent MUST re-validate the complete set of Internal RIM_Certs for the Engine by running MTM_InstallRIM with an incremented counter value.	Revoking RIMs	MUST	
6.3.4.5	67	216		Once the new set of Internal RIM_Certs has been created, the RIM Conversion Agent MUST then increment the Engine-specific counterRIMProtect.	Revoking RIMs	MUST	
6.3.4.5	67	217		To complete the RIM revocation process, all the old RIM_Certs MUST be removed from the Engine's internal store of RIM_Certs and the new RIM_Certs MUST be added.	Revoking RIMs	MUST	
6.3.4.7	69	218		The reporting key MUST have a credential proving to the Authorized Party that the key belongs to the Engine concerned.	Reporting of RIMs	MUST	
6.3.4.7	69	219		For security reasons, this key MUST NOT be an Attestation Identity Key... Instead, the Engine MUST generate an additional key-pair for signing reports, and have the public half of the key signed by an Attestation Identity Key (using TPM_CertifyKey)	Reporting of RIMs	MUST	
6.3.5.2	70	220	{{Replacing internal RIM_Certs from backup}}	Once the information is restored then a reset MUST be triggered to start the standard boot process as defined earlier in this section.	Recovery/Restore of RIM Certs	MUST	
7.1.2	74	221	{{Engine supports an RTV}}	The "SUCCESS" state is at least one in which there is some confidential, protected information available on the platform, and such data MUST NOT be available in the "FAILED" state.	Maintaining Integrity After Boot – Security States	MUST	
7.1.2	74	222	{{Engine supports an RTV}}	Thus the RTS MUST be operational in a "SUCCESS" state, and MUST NOT be operational in a "FAILED" state.	Maintaining Integrity After Boot – Security States	MUST	
7.1.3	76	223	{{Engine supports an RTV}}	Run-time integrity is REQUIRED to protect the integrity of mandatory platform functions	Maintaining Integrity After Boot – Protecting Mandatory Functions	MUST	
7.2.1	79	224	{{Engine supports an RTV}}	One or more forms of hardware protection MUST form the basis of any preventative approach.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.1	79	225	{{Engine supports an RTV}}	Such hardware protected capabilities are REQUIRED to implement the Roots of Trust, and are REQUIRED to enforce a security response on exiting a "SUCCESS" state.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.2	79	226	{{Engine supports an RTV}}	Such software isolated capabilities are REQUIRED to implement the Roots of Trust and at least one run-time Verification Agent, and are REQUIRED to safeguard the Roots of Trust if the platform exits a "SUCCESS" state.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.3	80	227	{{Engine supports an RTV}}	The Device design MUST provide protection ensuring that attacks using kernel mode privileges could not subvert the Roots of Trust, OR ensure that the kernel is of sufficiently low complexity as to be certifiably resistant to such attacks.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.4	80	228	{{Engine supports an RTV}}	Given that preventative measures are not perfect, the Device MUST ensure that any failure to correctly restrict software privileges within the main OS either cannot impair the platform's mandatory functions (so can't force the platform out of a "SUCCESS" state) or else cannot impair the platform's security response when leaving a "SUCCESS" state.	Maintaining Integrity After Boot – Preventative Methods	MUST	

7.2.5	81	229	{{Engine supports an RTV}}	Further security techniques MAY be deployed to prevent malicious (or just badly written) applications being loaded onto the device at all... While these techniques MAY be used in general, they MUST be used in some restricted circumstances (see below), and thus capabilities to support them are REQUIRED by this specification.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.5	81	230	{{Engine supports an RTV and the RIM Update Protocol}}	Where the Device supports the RIM update protocol, it MUST follow the instructions of RIM Auths in respect of whether and when to check loaded software against a RIM.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.5	81	231	{{Engine supports an RTV}}	The Device MAY prevent certain applications from being installed onto the Device post manufacture. This MUST be controlled by a security policy	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.5	81	232	{{Engine supports an RTV}}	The Engine (or its associated RIM Conversion Agent, if external to the Engine) MUST have a capability to determine the expected image of some installed applications, as they will appear at application launch.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.5	81	233	{{Engine supports an RTV}}	This expected launch image SHALL be composed into the form of an internal RIM Cert, created by a RIM Conversion Agent.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.5	81	234	{{Engine supports an RTV}}	The internal RIM Cert MUST be associated with a target object and time of measurement, indicating to a suitable Measurement Agent that the installed application code (target object) must match the RIM either prior to any application launch or at any application launch (target time).	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.5	81	235	{{Engine supports an RTV}}	Such an internal RIM Cert MUST be created for any installed applications whose execution could impair mandatory functions.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.2.5	81	236	{{Engine supports an RTV}}	In particular, where the application itself is defined as a mandatory function (by a RIM_Auth through an external RIM Cert) then an internal RIM Cert MUST be created.	Maintaining Integrity After Boot – Preventative Methods	MUST	
7.3.1	83	237	{{Engine supports an RTV}}	Each Engine which supports an RTV MUST enforce a Reactive Response dictated by the security policy of its stakeholder, whenever a TIM is found to not match its associated RIM in that Engine.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	83	238	{{Engine supports an RTV}}	Upon an IF an engine MUST immediately deny access to security sensitive assets, including the RTS: this functionality MUST only be restored when the engine's integrity has been restored, through a secure boot.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	83	239	{{Engine supports an RTV}}	Such protected capabilities MUST be available to the engine, and where used, MUST ensure that the RTS can be turned OFF upon an IF notification and that it will stay off until the next boot cycle.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	83	240	{{Engine supports an RTV}}	TCG_Reactive capabilities MUST be able to enforce the security policy set by the stakeholder: that policy may require an immediate engine RESET.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	83	241	{{Engine supports an RTV}}	The Engine MUST support at least one MVA which carries on running as a mandatory function after OS start-up, and this MUST run within protected capabilities.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	83	242	{{Engine supports an RTV}}	The PRMVA MUST perform at least one form of scheduled i.e. time-based integrity measurement, and verify it using a RIM_run Cert	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	84	243	{{Engine supports an RTV}}	Unless The PRMVA is able to perform all run-time integrity checks by itself, The Engine MUST support and use at least one Secondary RMVA (SRMVA) running outside protected capabilities.	Maintaining Integrity After Boot – Reactive Methods	MUST	

7.3.1	84	244	{{Engine supports an RTV}}	The PRMVA SHALL use RIM_run Certs (see below) to check the operation and integrity of the SRMVA, if this exists.	Maintaining Integrity After Boot - Reactive Methods	MUST	
7.3.1	84	245	{{Engine supports an RTV and SRMVA}}	The PRMVA MUST have access to a clock that cannot be changed by code running outside protected capabilities.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	84	246	{{Engine supports an RTV and SRMVA}}	At least some checks by the PRMVA on the SRMVA SHALL be time-based, so that tampering with the SRMVA can only occur for a limited time before being detected.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	84	247	{{Engine supports an RTV and SRMVA}}	If it exists, the Secondary RMVA SHALL then use RIM_run Certs to check The operation and integrity of other engine components, which MAY include further RMVAs	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	84	248	{{Engine supports an RTV}}	Other secure boot Engines MAY also have protected capabilities to support their own PRMVAs. If they don't, they MUST still have RMVAs, and there MUST be a transitive run-time trust chain from the Device Manufacturer's PRMVA through to the RMVAs of each other Engine.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	84	249	{{Engine supports an RTV and a Watchdog Timer}}	A WDT SHALL have the following functional properties: Upon an event failure the PRMVA MUST generate a Mandatory Error Response. The Mandatory Error Response: a. MUST cause access-denial to all cell phone telephony resources and services (where these are available to the Engine), with the possible exception of emergency assistance services when the integrity of those services can be assured. b. MUST disable or block MTM functionality until the next time the engine boots.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	84	250	{{Engine supports an RTV and a Watchdog Timer}}	The PRMVA MUST require access to its control interface only from authorized software.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	85	251	{{Engine supports an RTV and a Watchdog Timer}}	The PRMVA MUST have a timing [clock] source which is ensured to run whenever the Host CPU is running. The PRMVA MUST be able to detect when it has not been called within a (configurable) time period. If the time period elapses without a call, the PRMVA MUST treat this as an event failure.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	85	252	{{Engine supports an RTV and a WDT that measures other Engine code}}	The secure integrity measurement: MUST scan and measure the memory content of the host engine on one or more segments of contiguous physical memory locations, MUST control the addressing and have read access to tested host memory content independent of run-time host software, MUST have configuration data which can be locked down on each boot-cycle until the next engine boot independent of run-time host software, MUST have scan configuration data for each segment which includes control of: a. The address range of each memory segment to be measured b. The expected/reference value of each segment's measurement c. The rate at which memory is to be scanned d. The number of memory segments to be scanned	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	85	253	{{Engine supports an RTV and a WDT with Algorithmic Sequence Checker}}	The Algorithm Sequence Checker: MUST have a pseudo-random sequence which is started on each boot-cycle, and uniquely seeded, with the seed distributed to both the calling function and the PRMVA, MUST generate a new pseudo-random sequence state/count upon receiving an INCREMENT command, MUST receive and validate a number [ExpectedState] with every INCREMENT command against its new pseudo-random sequence state/number, MUST generate a PRMVA Mandatory Error Response upon a validation FAILURE.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	85	254	{{Engine supports an RTV}}	It is REQUIRED that if there is a non-trivial transitive chain of trust (i.e. if the PRMVA does not perform all run-time verifications), then links in that chain are supported by integrity measurements, verified using a non-trivial set of RIM_run Certs (i.e. at least one for each link in the chain).	Maintaining Integrity After Boot - Reactive Methods	MUST	
7.3.1	85	255	{{Engine supports an RTV}}	An uncorrectable failure of a TIM to match a RIM_run MUST cause the engine to transition to a “FAILED” state.	Maintaining Integrity After Boot - Reactive Methods	MUST	

7.3.1	85	256	{{Engine supports an RTV}}	Each Engine which supports an RTV MUST have a capability to take measurements at intervals of some executing code and MUST have a capability to check each such measurement against an authorised expected value.	Maintaining Integrity After Boot - Reactive Methods	MUST	
7.3.1	85	257	{{Engine supports an RTV}}	The measurement intervals MUST be defined by the Engine's stakeholder.	Maintaining Integrity After Boot - Reactive Methods	MUST	Yes
7.3.1	85	258	{{Engine supports an RTV}}	In general, a RIM_run Cert MUST be created where the engine stakeholder (or a RIM_Auth delegate of the stakeholder) has instructed that the executing image be subject to run-time integrity.	Maintaining Integrity After Boot - Reactive Methods	MUST	
7.3.1	86	259	{{Engine supports an RTV and mechanism of Explict instruction when to make measurements}}	Where explicit, an instruction indicating when to make measurements SHALL be provided through an extension to a RIM_Cert [see "TCG Mobile Trusted Module Specification"].	Maintaining Integrity After Boot - Reactive Methods	MUST	
7.3.1	86	260	{{Engine supports an RTV and mechanism of Explict instruction when to make measurements}}	The semantics of this extension MUST therefore describe the following options: - A specified (one-off) event in the boot sequence, as discussed in Section 5 And/Or a specified (possibly recurrent) event in the run-time environment, such as an application install, application launch, a hardware event (e.g. TPM command), a write event to certain files or areas of memory, or a system interrupt And/Or a specified (necessarily) recurrent interval of measurement in the run-time environment, defined in seconds and exceeding a platform minimum, or if not specified, a platform default interval.	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	86	261	{{Engine supports an RTV}}	The MTM MUST be able to store keys needed to verify external RIM_Certs	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	86	262	{{Engine supports an RTV}}	The MTM MUST be able to create, export, import and process internal RIM_Certs	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.1	86	263	{{Engine supports an RTV}}	The MTM MUST be able to store PCR values corresponding to previous measurements e.g. for comparison with current measurements	Maintaining Integrity After Boot – Reactive Methods	MUST	
7.3.2	87	264	{{Engine supports an RTV and Internal RIM_run Certs for Static portions of code images}}	The internal RIM_run Cert MUST be associated with a target object and time of measurement, typically indicating to a suitable Measurement Agent that the static part of the executing code is to be checked at a regular interval (a defined frequency) or at particular events.	Maintaining Integrity After Boot – Reactive Methods	MUST	
4.1.1	21	265		It is RECOMMENDED that the lists support at least three further engines, for a User, Communications Carrier and Service Provider	Architecture Overview	SHOULD	
4.1.3	30	266	{{Actual measurements obtained while an allocated RoT is built}}	When possible, these actual measurements SHOULD be compared against a signed value provided by each RoT's supplier.	Roots of Trust	SHOULD	
4.1.4	32	267		It is RECOMMENDED that the Device Manufacturer's engine detects Physical Presence and provides appropriate indications to other engines and to the engine's Owner.	Physical Presence	SHOULD	
5	38	268		Other engines with remote owners (i.e. owners who are not local Users of the platform) SHOULD support an RTV.	RTV	SHOULD	
5.1.2	40	269		At later stages of boot, the Root-of-Trust-for-Verification SHOULD pass the responsibility for verifying measurements to other Verification Agents (see Section 5.5).	RTV	SHOULD	
5.1.3.1	41	270		The supplier SHOULD establish a mechanism in order in allow for authentic and integral upgrades of the Root-of-Trust-for-Verification.	RTV	SHOULD	Yes
5.2	43	271		In the following paragraphs (Sections 5.2.1 to 5.2.5), a standardized method is defined for provisioning RIMs… this method SHOULD be supported for interoperability.	RIM Provisioning Method	SHOULD	
5.2.1	43	272	{{Support for recommended RIM provisioning method}}	The structure TPM_Verification_Key defined in "TCG Mobile Trusted Module Specification" SHOULD be used for RIM_Auth_Certs	Recommended RIM Provisioning Method	SHOULD	

5.2.2	44	273	{{Support for recommended RIM provisioning method}}	If the Root Verification Authority (or other RIM_Auth acting as a CA) wishes to retract delegated authorization, then it SHOULD do so by signing a periodic RIM_Auth_Validity_List indicating the key identifiers of which of its delegates are still valid.	Recommended RIM Provisioning Method	SHOULD	
5.2.3	46	274	{{Support for recommended RIM provisioning method and for processing additional advisory information}}	However, if the Engine processes the list of target objects (labels) for which a RIM_Auth is expected to provide RIMs, and notices it is missing a RIM for an object on this list, it SHOULD attempt to obtain one.	Recommended RIM Provisioning Method	SHOULD	
5.2.3	46	275	{{Support for recommended RIM provisioning method}}	However, for future proofing, it is strongly RECOMMENDED that all engines can process Validity Lists.	Recommended RIM Provisioning Method	SHOULD	
5.2.4	47	276	{{Support for recommended RIM provisioning method}}	RIM_Auths that are able to sign RIM Certificates SHOULD be able to revoke such certificates (typically also issuing a replacement).	Recommended RIM Provisioning Method	SHOULD	
5.2.4	47	277	{{Support for recommended RIM provisioning method}}	If a RIM_Auth is able to revoke its RIM_Certs, then it SHOULD do so by signing a periodic RIM_Validity_List indicating the serial numbers of which of its certs are still valid.	Recommended RIM Provisioning Method	SHOULD	
5.2.5	48	278	{{Support for recommended RIM provisioning method}}	In any case, it is strongly RECOMMENDED that all engines can process RIM Validity Lists	Recommended RIM Provisioning Method	SHOULD	
5.3.1	50	279		The following PCR allocation is RECOMMENDED for the engine of the device manufacturer. PCR 0: Relevant (non-identifying) characteristics of the HW platform. PCR 1: Relevant (non-identifying) information pertaining to the Roots of Trust is to be measured into PCR 1. PCR 2: Engine-Load events for the DM Engine are to be measured into PCR 2. PCR 3-6: PCRs 3-6 are to be used for DM proprietary measurements. PCR 7: DM Engine Operating System is to be measured into PCR 7. PCR 8-12: PCRs 8-12 are reserved for DM proprietary measurements. PCR 13-15: Unallocated at present.	Allocation of PCRs	SHOULD	
5.3.1	50	280		It is RECOMMENDED that PCRs 0 to 7 are verifiedPCRs	Allocation of PCRs	SHOULD	
5.3.2	50	281		The events SHOULD be described using the following set of parameters: Name: Name of the event Syntax: The actual byte-level representation of the event in BNF. Strings are in US ASCII.	Allocation of PCRs	SHOULD	
5.3.3	51	282		The RTE Diagnostic event SHOULD be generated to record diagnostic information about the trusted resources (RTS, RTR, RTE, RTM, RTV).			
5.3.4	51	283		The Engine Load event SHOULD be generated whenever new code or configuration that may affect its integrity is loaded into an engine.			
5.3.5	51	284		The Debug Mode event SHOULD be generated whenever an engine or its RTV enters debug mode.	Allocation of PCRs	SHOULD	
5.5	53	285		All higher layer extend or verified extend actions SHOULD also be recorded in a PCR Event Log	Other measurement and verification agents	SHOULD	
6.1.1.1	55	286		This key SHOULD be at least 2048 bits for an RSA key or an equivalent Elliptic Curve key of size at least 256 bits with an appropriate curve, or another key type permitted by the TPM specification for use as an SRK.	Provisioning of AIK and SRK	SHOULD	
6.1.3.2	56	287		For attestation interoperability… it is RECOMMENDED to use the “unified” credential	Provisioning of AIK and SRK	SHOULD	
6.2.1	58	288		For example, the DM Engine SHOULD allow the User to assert a physical presence, and then present that status to a User engine	Remote and Local Owners	SHOULD	
6.2.2	58	289	{{Locally owned engine supports secure boot}}	As this requires an RVAI public key to be set (and possibly IntegrityCheckRootData to be set as well), any “owner-less” secure boot SHOULD use a default combination of RVAI/IntegrityCheckRootData/RIM_Auths/RIM_Certs provided by the Device Manufacturer for that engine.	Remote and Local Owners	SHOULD	
6.2.2	58	290	{{Locally owned engine supports secure boot}}	If the local owner chooses to relinquish ownership, it is RECOMMENDED that the default boot settings are restored, enabling a local User to take ownership again if so desired (and if the MLTM flags permit).	Remote and Local Owners	SHOULD	
6.3.1	60	291	{{For new engines, insertion of RVAI is postponed until ownership is taken}}	In which case the engine SHOULD provide a proprietary owner authorized command to set the RVAI	Setting of RVAI and VerifiedPCRs	SHOULD	

6.3.1.1	60	292		Except in the case of a User engine, it is RECOMMENDED that either the RVAI key itself, or a hash of the RVAI key, is stored in the Mobile Trusted Module by use of the field integrityCheckRootData defined in the MTM_ Permanent_Data (see "TCG Mobile Trusted Module Specification".)	Setting of RVAI and VerifiedPCRs	SHOULD	
6.3.1.1	60	293		This integrityCheckRootData MAY be set at manufacture of the platform, or MAY be set at Engine creation, or MAY be set when taking ownership of an Engine (see above)… one of the three options SHOULD be used if integrityCheckRootData is set.	Setting of RVAI and VerifiedPCRs	SHOULD	
6.3.1.1	60	294		In case integrityCheckRootData is set, the flag loadVerificationRootKeyEnabled SHOULD be permanently set to FALSE, as the MTM would never be required to load a verification key without integrity checks or authorization.	Setting of RVAI and VerifiedPCRs	SHOULD	
6.3.1.1	60	295		If no such record or integrity check of the RVAI is held in the MTM, then the MTM is dependent on the RTV of its Engine to load in the correct RVAI key at the start of boot. In such cases, the flag loadVerificationRootKeyEnabled SHOULD be initially set to TRUE on each power-up cycle, to enable the RTV to load in the RVAI key without integrity checks.	Setting of RVAI and VerifiedPCRs	SHOULD	
6.3.2	60	296		The counterBoostrap counter SHOULD reside on the main processor (or else be crypto-graphically bound to the main processor)… and SHOULD resist tampering to reset it to a previous state to the extent defined in the "TCG Mobile Trusted Module specification" section 6.1.4.	Monotonic Counters	SHOULD	
6.3.2	61	297		The counterRIMProtect counter SHOULD reside on the main processor (or else be crypto-graphically bound to the main processor)… and SHOULD resist tampering to reset it to a previous state to the extent defined in the "TCG Mobile Trusted Module specification" section 6.1.4.	Monotonic Counters	SHOULD	
6.3.2	61	298		The counterStorageProtect counter SHOULD be protected on (or bound to) the main processor… and SHOULD resist tampering to reset it to a previous state to the extent defined in the "TCG Mobile Trusted Module specification" section 6.1.4.	Monotonic Counters	SHOULD	
6.3.3.1	62	299		If the RVAI is not already loaded into the MTM, then it SHOULD be loaded by the RTV using the MTM_LoadVerificationKey command with the parentKey field null.	Pristine Boot	SHOULD	
6.3.3.1	62	300		For the DM engine, this will be the only verification root key that SHOULD be used by the MTM, and the MTM_LoadVerificationRootKeyDisable command SHOULD be issued to prevent any further root keys being loaded.	Pristine Boot	SHOULD	
6.3.3.1	62	301		Starting from the RVAI key, the trust chain for each RIM_Auth_Cert required to reach the next level of operation will be verified by the RTV using the MTM_LoadVerificationKey command. Once each of the RIM_Auth_Certs has been verified, then each of the external RIM_Certs can be verified by using the MTM_VerifyRIMCert command with the external RIM_Cert in the rimCert field and the RIM_Auth_Cert in the rimKey field. This process SHOULD be used to validate the structure and trust chain of the external RIM_Certs.	Pristine Boot	SHOULD	
6.3.3.1	62	302		During the above validation process, the RTV SHOULD read the value of the counterBootstrap counter using the TPM_GetCapability command	Pristine Boot	SHOULD	
6.3.3.1	62	303		If any RIM_Cert or RIM_Auth_Cert indicates that the counterBootstrap counter has been increased, then the appropriately authorized certificate that will authorize the incrementing of the counter SHOULD be identified and the MTM_IncrementBootstrapCounter SHOULD be called.	Pristine Boot	SHOULD	
6.3.4.1	64	304		For standard boot, the Engine SHOULD use internal RIM_Certs previously converted from external RIM_Certs.	When to Use External and when Internal Certs…	SHOULD	
6.3.4.1	65	305	{{Using internal RIM_Certs previously converted from external RIM_Certs}}	This check SHOULD be done by accessing up to date Validity Lists	When to Use External and when Internal Certs…	SHOULD	
6.3.4.1	65	306		If a remotely-owned engine has no ownerAuth data, the verificationAuth data SHOULD be static and assigned at manufacture.	When to Use External and when Internal Certs…	SHOULD	
6.3.4.1	65	307	{{Using external RIM_Certs directly during standard boot}}	This check SHOULD be done by accessing up to date Validity Lists	When to Use External and when Internal Certs…	SHOULD	
6.3.4.2	65	308		If the engine cannot get to a "SUCCESS" state to perform this update, then the platform SHOULD be forced into a mode where the FLASH can be reloaded and the procedure for pristine boot can be followed.	Updates and Revocations	SHOULD	

6.3.4.2	66	309		If verified and validated, the RIM Conversion Agent SHOULD create a new Internal RIM_Cert using MTM_InstallRIM, and pass on the updated software for installation (and use during next boot).	Updates and Revocations	SHOULD	
6.3.4.2	66	310		The old Internal RIM_Cert that corresponded to the update request SHOULD be removed from the Engine's internal store of RIM_Certs.	Updates and Revocations	SHOULD	
6.3.4.4	67	311		Alternatively, the measurement agent and its configuration data SHOULD have been updated as well as part of the software update package.	Updates and Revocations	SHOULD	
6.3.4.5	67	312		RIM_Auths SHOULD send updated revocation info (preferably a RIM validity list) to the Engine via any suitable update protocol, as described in Section 6.3.4.6	Updates and Revocations	SHOULD	
6.3.4.7	68	313		It is not RECOMMENDED to provide certs which uniquely identify the device during attestation.	Reporting of RIMs	SHOULD	
7.1.2	74	314	{{Engine supports an RTV}}	If "off", then clearly no secret keys or other confidential info are available, unless they were being temporarily stored outside the RTS (and if they were, they SHOULD now be erased to stay confidential).	Maintaining Integrity After Boot – Security States	SHOULD	
7.1.3	77	315	{{Engine supports an RTV}}	In addition to specifying mechanisms for, and requiring runtime integrity protection for mandatory functions and components, this specification also allows for these mechanisms to be used to protect the runtime integrity of discretionary platform components, and it is RECOMMENDED that the runtime integrity of discretionary platforms components is protected in this way.	Maintaining Integrity After Boot – Protecting Mandatory Functions	SHOULD	
7.2.5	81	316	{{Engine supports an RTV}}	The Device MAY prevent certain applications from being installed onto the Device post manufacture. This MUST be controlled by a security policy, and the state of the security policy SHOULD be protected by a RIM.	Maintaining Integrity After Boot – Preventative Methods	SHOULD	
7.2.5	81	317	{{Engine supports an RTV}}	whether the application install is blocked: The application code is supplied with additional data indicating compatible Devices, and this Device is not one of them The application code is supplied with an internal integrity check, and the check fails. The application code does not match a RIM which the Device has been provided with to check such an application pre-install. The application code is intended to execute in an OS with a privileged API structure, but does not clearly declare what privileges (APIs) etc it requires to execute. The application code declares privileged APIs, but is not recognizable as "trusted" according to the security policy. (For example, it is not signed, the signature is invalid, there is no code-signing certificate, the code-signing certificate is not valid or is issued by an untrusted CA). The application code declares APIs whose use could harm one or more of the Device's stakeholders (especially the Device User), but the source of the application is not identifiable. The application code is identifiable by the Device as revoked (e.g. it has a revoked code-signing certificate or matches a revoked RIM). The application code matches known signatures for malware (viruses, Trojans etc.)	Maintaining Integrity After Boot – Preventative Methods	SHOULD	
7.2.5	81	318	{{Engine supports an RTV}}	Alternatively, where the Device allows an install despite some of the above conditions, the Device SHOULD warn the Device User that there may be danger in installing the application.	Maintaining Integrity After Boot – Preventative Methods	SHOULD	
7.2.5	81	319	{{Engine supports an RTV}}	If the User continues anyway, the Device SHOULD restrict the application's function by denying it access to privileged APIs	Maintaining Integrity After Boot – Preventative Methods	SHOULD	
7.2.5	82	320	{{Engine supports an RTV}}	In addition, even where not defined as a mandatory function, an internal RIM Cert SHOULD be created for any OS updates, and SHOULD be created by default for applications using privileged APIs.	Maintaining Integrity After Boot – Preventative Methods	SHOULD	
7.2.5	82	321	{{Engine supports an RTV}}	Accordingly, if it is the default policy to create internal RIM Certs at install, it is RECOMMENDED that they specify a validity time of "at launch" rather than "prior to launch" as this enables a Preventative response.	Maintaining Integrity After Boot – Preventative Methods	SHOULD	

7.2.5	82	322	{{Engine supports an RTV}}	It is RECOMMENDED that the decision about whether to “Prevent” or “React” is determined through the time of measurement associated with the RIM_Cert.	Maintaining Integrity After Boot – Preventative Methods	SHOULD	
7.2.5	82	323	{{Engine supports an RTV}}	It is RECOMMENDED that if an Engine does pre-emptively block a launch in this way, then it gives a warning to the Device User what has happened.	Maintaining Integrity After Boot – Preventative Methods	SHOULD	
7.2.5	82	324	{{Engine supports an RTV}}	The User SHOULD then have an option to uninstall the application concerned, or attempt a repair/re-install.	Maintaining Integrity After Boot – Preventative Methods	SHOULD	
7.3.1	83	325	{{Engine supports an RTV}}	To assure this response, the Engine SHOULD utilize TCG protected capabilities, termed “TCG_Reactive” protected capabilities.	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.1	83	326	{{Engine supports an RTV}}	If the DM’s Engine is affected, and the Reactive Response does not require an immediate engine RESET, then the Engine SHOULD inform the user of a serious security error.	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.1	83	327	{{Engine supports an RTV}}	If relevant, the Device SHOULD warn the user that data/running processes on the affected Engine could be lost or else functionality temporarily impaired.	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.2	87	328	{{Engine supports an RTV}}	An internal RIM_run Cert SHOULD be created for any launched applications which are defined as mandatory functions, or whose malfunction would compromise mandatory functions.	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.2	87	329	{{Engine supports an RTV}}	In particular, such a RIM_run Cert SHOULD be created for the OS kernel.	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.2	87	330	{{Engine supports an RTV}}	Also, to ensure a transitive chain of trust, such a RIM_run Cert SHOULD be created for at least one run-time Verification Agent within the main OS	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.2	87	331	{{Engine supports an RTV}}	Any part of the executing code image of any launched application which is defined as a mandatory function which – by Device design - is expected to be static SHOULD be measured and composed into the form of an internal RIM_run Cert.	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.2	88	332	{{Engine supports an RTV}}	The use of event-based checking... is RECOMMENDED.	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.2	88	333	{{Engine supports an RTV}}	In particular, at the level of the PRMVA it is possible to identify lots of potential trigger events... some of which SHOULD be used to carry out an integrity check.	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.2	88	334	{{Engine supports an RTV}}	As well as the mandatory uses mentioned in Section 7.3.1, the MTM SHOULD be used to detect certain triggering events for run-time measurements.	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
7.3.2	89	335	{{Record in the MTM the fact that a re-measurement has happened}}	If repeated measurements are extended into the MTM, it is RECOMMENDED to use a different - but matching - PCR from the original PCR, and then keep this matching PCR changing while the original PCR retains its boot-time value (or other first extended value)	Maintaining Integrity After Boot – Reactive Methods	SHOULD	
4.1.1	18	336		An entity MAY perform one or more stakeholder roles.	Architecture Overview	MAY	
4.1.1	20	337		Engines in the DM_mandatoryEngineList… MAY provide non-critical services whose access to TCG functionality can be denied by the local Operator.	Architecture Overview	MAY	

4.1.1	20	338		Engines in the DO_mandatoryEngineList… MAY provide indispensable services that are not subject to regulatory enforcement, and MAY provide non-critical services whose access to TCG functionality can be denied by the local Operator	Architecture Overview	MAY	
4.1.1	20	339		Engines in the DO_discretionaryEngineList… MAY provide non-critical services whose access to TCG functionality can be denied by the local Operator..	Architecture Overview	MAY	
4.1.1	21	340		The DM's engine MAY use an integrity challenge to determine whether an engine is working properly	Architecture Overview	MAY	
4.1.1.	26	341		There MAY be unused leaf keys.	Architecture Overview	MAY	
4.1.2	28	342		The Internal Trusted Services MAY export Trusted Services that will be the Internal Trusted Services for another engine.	Reference Engine and Exported Services	MAY	
4.1.3	30	343		In cases where explicit measurement would be circular and have no security value (such as a measurement of the RTM, or of an RTE used to build the RTM or RTS), then the measurement values MAY be supplied with a RoT by the RoT's supplier and passed to the RTM on request. For example, such a pre-supplied RoT "measurement" MAY just consist of a component label (like a MTM Manufacturer Name and Part Name), and a component version number.	Roots of Trust	MAY	
4.1.3	30	344		Alternatively, such measurements MAY consist of actual values obtained while an allocated RoT was built.	Roots of Trust	MAY	
4.1.3	31	345		The RTE MAY support customization by the engine's stakeholder in order to dictate the trusted services and resources that have to be present.	Roots of Trust	MAY	
4.1.3	31	346		A Device Manufacturer's Engine … MAY incorporate an RTE.	Roots of Trust	MAY	
5	38	347		An engine with a local owner (i.e. a User engine) MAY support an RTV	RTV	MAY	
5.1.3.2	41	348		If the stakeholder wants to re-customize the Root-of-Trust-for-Verification at a later point in time he MAY replace the whole customisation by a new version of it or only components of the customisation (for instance, he MAY only want to change the lists of services which have to be measured or the list of RIMs and RIM Certificates).	RTV	MAY	
5.2	42	349		This determination MAY be implicit: if the Engine can detect that a particular source will never revoke any of its RIMs, it does not need to retrieve explicit information on whether a given RIM is still valid.	RIM Provisioning Method	MAY	
5.2.1	43	350	{{Support for recommended RIM provisioning method}}	For simplicity, the Root Verification Authority MAY directly sign the RIM_Certs that must be checked by the RTV (or other verification agents) of the Engine.	Recommended RIM Provisioning Method	MAY	
5.2.1	43	351	{{Support for recommended RIM provisioning method}}	Delegates that are authorized directly by the Root Verification Authority are called Primary RIM_Auths; the Primary RIM_Auths MAY in turn delegate further RIM_Auths.	Recommended RIM Provisioning Method	MAY	
5.2.1	43	352	{{Support for recommended RIM provisioning method}}	Where a RIM_Auth is able to act as a CA, it MAY also issue X.509 certificates to other RIM_Auths.	Recommended RIM Provisioning Method	MAY	
5.2.1	43	353	{{Support for recommended RIM provisioning method}}	The RIM_Auth_Cert MAY also be securely bound to any of the following lists restricting key usage (e.g. via the TPM_Verification_Key extension digest field). Such lists MAY contain wild-cards, from/to ranges etc. A list of platforms and Engines for which this RIM_Auth is allowed to provide RIMs. A list of PCRs that this RIM_Auth is allowed to instruct verification agents to extend. A list of target objects (labels) for which this RIM_Auth is allowed to provide RIMs.	Recommended RIM Provisioning Method	MAY	
5.2.1	44	354	{{Support for recommended RIM provisioning method}}	The RIM_Auth_Cert MAY also be bound to advisory information… . Such information could include: A list of target objects (labels) for which this RIM_Auth is expected to provide RIMs. A URL for the RIM_Auth, indicating where the most recent information signed by that RIM_Auth (e.g. a full set of RIM_Certs and validity lists) MAY be obtained.	Recommended RIM Provisioning Method	MAY	
5.2.3	46	355	{{Support for recommended RIM provisioning method}}	An Engine MAY ignore additional advisory information that is bound to the RIM_Auth_Cert.	Recommended RIM Provisioning Method	MAY	
5.2.3	46	356	{{Support for recommended RIM provisioning method}}	If it is known at Engine design that none of the RIM_Auths will ever sign RIM_Auth Validity Lists (i.e. that no RIM_Auths will ever be revoked), then this processing functionality MAY be omitted from the Engine.	Recommended RIM Provisioning Method	MAY	
5.3.1	49	357		However, once the measurement agent has finished running, its PCR MAY then be assigned to another measurement agent which has just started up.	Allocation of PCRs	MAY	

5.4	52	358		Agents that are not parents MAY be candidates for run-time measurement (and verification) agents, i.e. entities which continue to measure after boot	Other measurement and verification agents	MAY	
6.1.1	55	359		The SRK MAY be generated on the platform (refer to TCG 1.0 Architecture Overview section 4.3.1.5 for Random Number Generator guidelines, and to FIPS 140-2 for general guidance on RNG and key generation).	Provisioning of AIK and SRK	MAY	
6.1.1	55	360		For the DM engine (and if necessary, other remotely owned engines), the SRK MAY be generated externally and inserted into the engine during manufacture time based on limitations of the engine performance.	Provisioning of AIK and SRK	MAY	
6.1.2	55	361		The Endorsement Key (EK) is an OPTIONAL element for remotely-owned engines in the Mobile environment	Endorsement Key	MAY	
6.1.3.2	56	362		However, an engine MAY not include the EK	Provisioning of AIK	MAY	
6.2.1	57	363		However, a remote owner MAY be able to take ownership at a later date, if not already set, through TPM_TakeOwnership (which is OPTIONAL within the MRTM command set).	Remote and Local Owners	MAY	
6.2.1	57	364		The remote owner MAY also be able to relinquish ownership through TPM_OwnerClear (which is also OPTIONAL for a MRTM).	Remote and Local Owners	MAY	
6.2.1	57	365		The User engine's MLTM MAY already be enabled and activated; if not, the User will need to set those flags before taking ownership.	Remote and Local Owners	MAY	
6.2.2	58	366		An engine with a local owner (i.e. a User engine) MAY also provide secure boot functionality.	Remote and Local Owners	MAY	
6.2.2	58	367		A secure boot User engine MAY have no owner set yet: in particular this will be the case on a User engine's very first boot.	Remote and Local Owners	MAY	
6.3.1	59	368		The RVAI for other secure boot engines MAY also be installed at manufacture.	Setting of RVAI and VerifiedPCRs	MAY	
6.3.1	59	369		Special considerations apply either 1) where the RVAI for an engine is not known at manufacture, or 2) where the engine is only created on the platform after manufacture. In case 1), the Device Manufacturer MAY choose to initialize each engine with a copy of the DM's own RVAI, but assign each copy a distinct key identifier ("my_id").... In case 2), the Device Manufacturer MAY provide a means to specify the RVAI key when an engine is created.	Setting of RVAI and VerifiedPCRs	MAY	
6.3.1	59	370		Alternatively, insertion of an RVAI key MAY be postponed until someone has taken full ownership of the new engine	Setting of RVAI and VerifiedPCRs	MAY	
6.3.1.1	60	371		This integrityCheckRootData MAY be set at manufacture of the platform, or MAY be set at Engine creation, or MAY be set when taking ownership of an Engine (see above).	Setting of RVAI and VerifiedPCRs	MAY	
6.3.2	61	372		The counterStorageProtect counter … MAY be used to support the off-chip storage images of other counter values	Monotonic Counters	MAY	
6.3.3.1	63	373		A similar process MAY apply to other engines on the platform during pristine boot i.e. an engine is started, discovers it has no internal RIM_Certs and attempts to boot using external RIM_Certs.	Pristine Boot	MAY	
6.3.3.1	63	374		Alternatively, the pristine boot process MAY be designed so that the DM Engine completes its own building, and then creates the other engines fully built, but in a simplified state.	Pristine Boot	MAY	
6.3.3.1	63	375		The other engines MAY not need their own pristine boot sequences.	Pristine Boot	MAY	
6.3.3.2	64	376		The alternative execution path MAY specify an alternative target object to execute in the case of a failure to verify an object.	Standard Boot	MAY	
6.3.3.2	64	377		The alternative execution path MAY also extend a RIM_Cert into a PCR in order to record the failure of a target object.	Standard Boot	MAY	
6.3.4.1	64	378		The [external] RIM_Certs themselves MAY be revoked using the counterBootstrap monotonic counter.	When to Use External and when Internal Certs	MAY	
6.3.4.1	64	379		For standard boot, the Engine… MAY use internal RIM_Certs previously converted from legacy Device Management protocols that are not RIM aware.	When to Use External and when Internal Certs	MAY	
6.3.4.1	64	380		These internal RIM_Certs MAY be revoked using the counterRIMProtect monotonic counter.	When to Use External and when Internal Certs	MAY	
6.3.4.1	65	381		This check ...MAY use any equivalent revocation checking mechanism for legacy DM, or MAY be implicit.	When to Use External and when Internal Certs	MAY	
6.3.4.1	65	382		verificationAuth MAY be a mirror of ownerAuth for remotely-owned Engines	When to Use External and when Internal Certs	MAY	

6.3.4.1	65	383		For standard boot, the Engine MAY use external RIM_Certs directly.	When to Use External and when Internal Certs	MAY	
6.3.4.1	65	384		This check … MAY use any equivalent revocation checking mechanism or MAY be implicit.	When to Use External and when Internal Certs	MAY	
6.3.4.3	66	385		A RIM ConversionAgent MAY also be called upon during legacy updates (as well as upon updates of external RIM_Certs).	Updates and Revocations	MAY	
6.3.4.4	67	386		As new software is required to be measured and verified, the Engine may find it also needs to modify the measurement configuration data of an associated measurement agent. The RIM Conversion Agent MAY create a further internal RIM_Cert to protect this modified configuration data.	Updates and Revocations	MAY	
6.3.4.5	67	387		During an update process, it MAY happen that existing Internal RIM_Certs need to be removed from the set of authorized RIMs because of revoked RIM_Auths or revoked external RIM_Certs.	Revoking RIMs	MAY	
6.3.4.5	67	388		A RIM MAY also be removed entirely, based on a larger update that makes an old component obsolete, or just removes the component from the verified boot process.	Revoking RIMs	MAY	
6.3.4.5	67	389		Alternatively, the whole measurement agent and its configuration data MAY be updated as part of the software update package.	Revoking RIMs	MAY	
6.3.4.7	68	390		As a further privacy protection, the reported certs MAY also be encrypted to the Authorized Party.	Reporting of RIMs	MAY	
6.3.5.2	70	391		If a engine determines that the set of internal RIM_Certs are missing or corrupted then it MAY attempt to replace them from a backup version.	Recovery/Restore of RIM Certs	MAY	
7.2.5	80	392	{{Engine supports an RTV}}	Further security techniques MAY be deployed to prevent malicious (or just badly written) applications being loaded onto the device at all.	Maintaining Integrity After Boot – Preventative Methods	MAY	
7.2.5	81	393	{{Engine supports an RTV}}	These techniques MAY be used in general	Maintaining Integrity After Boot – Preventative Methods	MAY	
7.2.5	81	394	{{Engine supports an RTV}}	The Device MAY prevent certain applications from being installed onto the Device post manufacture.	Maintaining Integrity After Boot – Preventative Methods	MAY	
7.2.5	81	395	{{Engine supports an RTV}}	This determination MAY be achieved in some cases simply by measuring the code image just after it has been installed.	Maintaining Integrity After Boot – Preventative Methods	MAY	
7.2.5	82	396	{{Engine supports an RTV}}	For simplicity, an internal RIM Cert MAY be created by default after any application install.	Maintaining Integrity After Boot – Preventative Methods	MAY	
7.2.5	82	397	{{Engine supports an RTV}}	Alternatively, if the time indication is “at launch” then the Engine MAY still measure the target application immediately before launch, and if necessary, pre-emptively block the launch.	Maintaining Integrity After Boot – Preventative Methods	MAY	
7.3.1	83	398	{{Engine supports an RTV}}	If an Engine other than the DM’s Engine is affected, the DM’s Engine MAY attempt to restart the affected Engine.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.1	84	399	{{Engine supports an RTV}}	In order to decrease the risk of carefully timed attacks, the PRMVA MAY randomize the intervals between its measurements.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.1	84	400	{{Engine supports an RTV}}	If it exists, the Secondary RMVA SHALL then use RIM_run Certs to check the operation and integrity of other engine components, which MAY include further RMVAs, e.g. running as applications within the OS.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.1	84	401	{{Engine supports an RTV}}	The SRMVA MAY perform regular time-based measurements, or irregular event-based measurements (e.g. triggered by an alert that another RMVA wants to measure and verify something).	Maintaining Integrity After Boot – Reactive Methods	MAY	

7.3.1	84	402	{{Engine supports an RTV}}	Other secure boot Engines MAY also have protected capabilities to support their own PRMVAs. If they don't… This MAY involve the DM's PRMVA making direct measurement and verification of other Engines, or indirect measurement and verification through the DM's secondary RMVA, or even more indirectly through other RMVAs on the DM's Engine.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.1	84	403	{{Engine supports an RTV}}	A specific functional definition for a PRMVA and associated TCG_Reactive capabilities is now given, referred to as a Watchdog Timer (WDT). The PRMVA and TCG_Reactive capabilities MAY be implemented as a WDT.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.1	85	404	{{Engine supports an RTV}}	The integrity measurement MAY include functional requirements in the form of an Algorithm Sequence Checker (ASC).	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.1	85	405	{{Engine supports an RTV}}	The Algorithm Sequence Checker… MAY set a configurable pseudo-random number equation (i.e. feedback taps) on each boot-cycle.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.1	85	406	{{Engine supports an RTV}}	Abstractly, a RIM_run Cert is any structure which defines the authorized expected value of a run-time measurement. Concrete implementations MAY have the same structure as the RIM Cert defined in "TCG Mobile Trusted Module Specification", or MAY have a simpler (proprietary) structure.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.1	85	407	{{Engine supports an RTV}}	RIM_run Certs MAY be internal or external.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.1	85	408	{{Engine supports an RTV}}	In general, a RIM_run Cert MUST be created where the engine stakeholder (or a RIM_Auth delegate of the stakeholder) has instructed that the executing image be subject to run-time integrity. This instruction mechanism MAY be supported explicitly through the mechanism of external RIM_Certs, or MAY be fixed implicitly by Engine design, or MAY be updateable through trusted Device management.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.2	88	409	{{Engine supports an RTV}}	As well as the mandatory uses mentioned in Section 8.2.2, the MTM SHOULD be used to detect certain triggering events for run-time measurements. For example, certain PCR extends or uses of certain high sensitivity commands (like migration, management, delegation or owner changes) MAY act as triggering events.	Maintaining Integrity After Boot – Reactive Methods	MAY	
7.3.2	89	410	{{Engine supports an RTV}}	The engine stakeholder MAY wish to record in the MTM the fact that a re-measurement has happened (e.g. a proof that run-time integrity is working might be needed for attestation purposes); if so, just discarding repeat measurement values will not achieve this.	Maintaining Integrity After Boot – Reactive Methods	MAY	Yes