Requirements for Technical Construction File (TCF) for IoT Device

To create confidence on IoT Device, Manufacture shall maintain Technical Construction File having following information. Vendor need a provide information pertaining to the entire requirements mentioned below.

General

Sl.No	Requirements from Vendor	**Details need to be provided**
1.	General description of IoT Device, usage of IoT device and environment of use.	
2.	IoT Device Software & Hardware Bill of Material (As per Annexure 'A').	
3.	Risk Assessment of IoT Device including applicable system	
4.	Details of implementation of requirements mentioned in ISO/IEC 27402 IoT security and privacy — Device baseline requirements and OWASP Application Security Verification Standard 4.0 - Appendix C: Internet of Things Verification Requirements [#1 to #14 for Level 1, #1 to #24 for Level 2 & #1 to #34 for Level 3] (As per Annexure 'B')	

Certificates

Sl.No	Requirements from Vendor	**Details need to be provided**
1.	Certificate for ISO 9001 (Scope should cover IoT Device Development, Manufacturing and Service (Manufacturer).	
2.	Certificate for ISO 9001 (Scope should cover IoT Device Supply of IoT Device (Supplier/ Distributor) if applicable.	
3.	Certificate of Incorporation (Manufacturer).	
4.	Certificate of Incorporation (Supplier).	
5.	Manufacturer authorization to supplier to place devices in Indian Market if applicable.	

Annexure 'A'

Software Bill of Material (Ref: https://cyclonedx.org/guides/sbom/introduction/#software-bill-of-materials-sbom) & https://www.ntia.gov/files/ntia/publications/sbom_minimum_elements_report.pdf)

Author Name—usually the organization that develops the software and **country of origin**.
Vendor Name—the name of the software vendor, including aliases (alternative names). Vendor and author may be different if a supplier is creating an SBOM on behalf of the vendor.
Component Name—the name and possible aliases of the software component.
Version String—the format of the version information is free-form, but should follow common industry usage.
Component Hash—the best way to identify a software component is to use a cryptographic hash that serves as a unique identifier.
Unique Identifier—in addition to the hash, each component must have an ID number that identifies it within the SBOM.
Relationship—defines the relationship between the component and the package. In most cases, the relationship is "included", meaning that a certain component is included in a certain package.
Time Stamp-Record of the date and time of SBOM data assembly

Hardware Bill of Material (Ref: https://cyclonedx.github.io/cyclonedx-property-taxonomy/cdx/device)

Manufacturer Name- The name of the hardware manufacturer & vendor **and their country of origin**
Component Name—The name of the hardware component.
Device:quantity -The total number of the specified component.
Device:function- The purpose of the component (Bluetooth, network, storage, microprocessor, connector, etc).
Device:location- The location on the board or related daughter-boards where the device exists.
Device:deviceType- The type of component such as SMD, thru-hole, etc
Device:serialNumber- Unique identifier using serial number if available
Device:sku- Internal inventory reference if available
Device:lotNumber- Lot or batch identification for the component
Device:prodTimestamp- Production timestamp for the component
Device:macAddress- Hardware address for network interfaces

Note: In addition to these minimum requirements, an BOM can include additional information such as security scores, common vulnerabilities and exposure codes (CVEs) of known vulnerabilities.

Annexure 'B'

Cl.No	Requirements as per 'ISO/IEC 27402 IoT security and privacy — Device baseline requirements'	Status		
		Yes	No	Implementation details need to be provided
5.1	**Requirements for IoT device policies and documentation**			
5.1.1	**Risk management**			
5.1.1.1.1	IoT devices shall have documentation recording the results of a risk assessment process performed at the IoT device level in the context of a risk assessment at the system level.			
5.1.1.1.2	The risk assessment process shall take into account intended outcomes for the intended use case.			
5.1.1.1.3	The risk assessment process shall also take into account the needs and expectations of interested parties (e.g. those parties on networks to which the IoT device is connected), including physical and logical undesired effects.			
5.1.1.1.4	The risk assessment shall take into account that IoT devices can be constrained (e.g. limited battery, little memory, 'weak' CPU), which informs the risk treatment process.			
5.1.1.1.5	Risk assessment and treatment processes shall be defined and applied as follows:			
	a) determine if separate risk assessment and treatment processes are necessary for different products;			
	b) select appropriate risk treatment options, taking account of the risk assessment results;			
	c) determine all controls that are necessary to implement the risk treatment option(s) chosen;			
	d) identify all security and privacy features of the IoT device from the controls identified in c) above;			
	e) compare the features identified in d) above with			

	those in 5.2, and verify that no necessary features have been omitted;			
	f) produce a Statement of Applicability that contains the necessary features [see steps d) and e)] and justification for inclusions and the justification for exclusions of features from 5.2;			
	g) if other standards related to device requirements are used, implement the requirements of those standards after steps a) through to f);			
	h) formulate a risk treatment plan;			
	i) inform the risk owner of the risk treatment plan and any residual risks, or where applicable, obtain their approval of the plan and acceptance of the residual risks.			
5.1.1.1.6	IoT devices shall implement the features and controls identified as necessary in its Statement of Applicability, as well as features and controls identified in 5.1.1.1.5, step g).			
5.1.1.1.7	The documentation shall be available for the supported lifetime of the product.			
5.1.2	**Information disclosure**			
5.1.2.1.1	IoT devices shall have user documentation that lists the features that the IoT device provides to support controls for security and privacy, making it clear if any of the IoT device requirements in 5.2 are not included.			
5.1.2.1.2	Such information shall be publicly available for the period of time the IoT device is supported.			
5.1.2.1.3	IoT devices shall be covered by a security support policy and other supporting documentation wherein users are made aware in advance of when security updates will be discontinued.			
5.1.3	**Vulnerability disclosure and handling processes**			
5.1.3.1.1	IoT devices shall have documentation that defines the vulnerability disclosure and handling processes that will apply for the supported lifetime of the device.			

5.1.3.1.2	Vulnerability disclosure and handling processes shall include, at a minimum, a capability to receive reports of potential vulnerabilities from the public.			
5.2	**Requirements for IoT device capabilities and operations**			
5.2.1	**General- This clause includes IoT device features to be used with a risk assessment and treatment process in accordance with 5.1.1.**			
5.2.2	**Configuration**			
5.2.2.1.1	If the configuration settings of the IoT device can be modified, only authorized entities shall be able to modify the configuration settings of the IoT device.			
5.2.2.1.2	If IoT devices are capable of changing the configuration of IoT and other devices, they shall only be capable of making such changes when authorized.			
5.2.3	**Software reset**			
5.2.3.1.1	If IoT devices have the capability to be reset, that process shall be secure.			
5.2.3.1.2	This capability shall only be executable by an authorized entity.			
5.2.4	**User data removal**			
5.2.4.1.1	If the IoT device stores user data, it shall provide a function for deleting appropriate user data stored on the device in any type of memory.			
5.2.4.1.2	The function shall be restricted to authorized entities only.			
5.2.5	**Protection of data**			
5.2.5.1.1	IoT devices shall be capable of protecting the data they store and transmit from unauthorized access, modification and disclosure.			
5.2.5.1.2	This shall include configuration settings, identifying data, user data, event logs and sensitive security parameters.			
5.2.5.1.3	IoT devices shall be capable of protecting their software (including firmware) from unauthorized			

	access and modification.			
5.2.5.1.4	IoT devices shall use cryptography (e.g. encryption with authentication, cryptographic hashes, digital signature validation) to prevent the confidentiality and integrity of data requiring protection from being compromised.			
5.2.5.2	**Additional recommendation**			
5.2.5.2.1	**General** When IoT devices are started up, they should check the integrity and authenticity of the software and/or firmware and enforce security controls. If the IoT device fails these checks, it should: — notify the user of any violation, — render itself inoperable, — operate in a fail-safe mode that provides security protection, or — initiate device recovery if recovery actions can be performed with integrity.			
	Upon first installation or maintenance, IoT devices should set themselves to secure default configurations. User configuration options should prevent users from choosing insecure configurations or provide a warning. If capable, IoT devices should have the ability to provide compartmentalization. IoT devices should use function modules to restrict access to system resources, which should only be granted to authorized entities. Trusted computing bases (TCB) should be kept as small as possible to minimize the surface that is exposed to attackers and to reduce the probability that a bug or feature can be used to circumvent security protections. Memory protection mechanisms such as memory safe languages, stack canaries, address space layout randomization (ASLR) and limited or no execute permissions are recommended wherever applicable.			

5.2.5.2.2	**Event logging** If capable, IoT devices should record sufficient details for each event to facilitate an authorized entity's ability to identify anomalous events and meaningfully analyse the associated data.			
5.2.5.2.3	**Sensitive security parameters** The outcome of the risk assessment in 5.1.1 should help determine whether an IoT device may include hard-coded or shared sensitive security parameters, if such parameters are unique per device and not universal.			
5.2.5.3	**Additional information**			
5.2.5.3.1	**General** Hardware-based solutions such as built-in crypto accelerators and dedicated hardware can enhance the use of cryptographic modules and cryptographic key protection capabilities to protect the data in storage and transit to meet the performance requirements. Physical countermeasures can support resistance to side channel attacks. Such functions can include hardware-based root of trust (RoT). RoT is a foundational feature to provide platform integrity and ensure a foundation to develop and support the device's chain of trust. The root of trust is ideally based on a hardware-validated boot process to ensure the system can be started using code from an immutable source. As such, RoT is essential to enable platform attestation including for a verified boot process. When used to protect secrets and device correctness, hardware can support a foundational root of trust upon which rich software functionality can be implemented more securely and safely. Compartments are protected by hardware-enforced boundaries to prevent a flaw or breach in one software compartment from propagating to other			

	software compartments in the system. Compartmentalization introduces additional protection boundaries within the hardware and software stack to create additional layers of defence in depth. For example, a common technique is to use operating systems processes or independent virtual machines as compartments. Integrity checking and recovery modes may not be appropriate in safety critical applications where continuous operation is essential.			
5.2.5.3.2	**Event logging** Implementation of event logging, including editing of logs, depends on device storage capabilities. IoT devices can support remote logging.			
5.2.6	**Interface access**			
5.2.6.1.1	IoT devices shall have mechanisms to limit logical access to its interfaces to authorized entities only.			
5.2.6.1.2	IoT devices shall employ appropriate authentication and access control mechanisms.			
5.2.6.1.3	Security and privacy requirements shall be assessed when designing and implementing the functions of IoT devices regarding creation and use of identifiers.			
5.2.6.1.4	IoT devices shall ensure that common values for critical security parameters, such as global private keys or standard passwords, are replaced by values that are unique per device or explicitly defined by an appropriate external entity before they are put into operation.			
5.2.6.2	**Additional recommendation(s)** The IoT device should be capable of being logically identified. While identifiers can enable a host of cybersecurity controls (such as asset management, automatic device discovery, and software updates), creating or using persistent identifiers should be avoided unless such use is unavoidable. Where such uses arise, the existence of such identifiers should			

	be made clear to users. Mechanisms to limit logical access (to authorized entities) should be applied to the following: a) the ability to enable or disable, through software or hardware means, any interfaces (including local and network interfaces); b) the ability to restrict access (e.g. through authentication) to all remote interfaces; c) the ability to identify or block devices not supported by an IoT device when it is attempting to access interfaces.			
5.2.6.3	**Additional information**			
5.2.6.3.1	**General** Examples of user interfaces include administrative consoles, web pages, APIs or other externally-exposed IoT device interfaces. Injection, XML external entities, cross site scripting and insecure deserialization are examples of common attacks to remote interfaces. Hardware-based capabilities can harden interface access protection against privilege escalation and control-flow attacks.			
5.2.6.3.2	**Identifiers** IoT devices can use identifiers in order to operate within an IoT system. Examples of such identifiers include serial numbers, cryptographic keys, and certificates.			
5.2.7	**Software and firmware updates**			
5.2.7.1.1	If the IoT device supports software updates, updates shall be performed using a secure procedure.			
5.2.7.1.2	Updates shall only be initiated by authorized entities.			
5.2.7.1.3	Unexpected interruption of an update shall leave the IoT device in a state that minimizes potential for harm, taking into account the risks of the IoT device not functioning as expected.			

5.2.8	**User Notifications**			
	IoT devices to notify users about about a negative event or condition. Some IoT devices do not have capabilities to actively inform the user (e.g. write a message on the screen, emit a sound or light), but they can respond with a message when queried or accessed remotely. IoT devices that do not have capabilities to directly inform users can send notifications and alerts via a local hub. A user query can be as simple as trying to access the device with a browser, mobile application, or something more complex. Alternatively, IoT devices can send a message to an alarm, monitoring, or logging device within the IoT system.			

Cl.No	**OWASP Application Security Verification Standard 4.0 - Appendix C: Internet of Things Verification Requirements**	**Status**		
		Yes	**No**	**Implementation details need to be provided**
	Level 1/2/3			
C.1	Verify that application layer debugging interfaces such USB, UART, and other serial variants are disabled or protected by a complex password.			
C.2	Verify that cryptographic keys and certificates are unique to each individual device.			
C.3	Verify that memory protection controls such as ASLR and DEP are enabled by the embedded/IoT operating system, if applicable.			
C.4	Verify that on-chip debugging interfaces such as JTAG or SWD are disabled or that available protection mechanism is enabled and configured appropriately.			
C.5	Verify that trusted execution is implemented and enabled, if available on the device SoC or CPU.			
C.6	Verify that sensitive data, private keys and certificates are stored securely in a Secure Element, TPM, TEE			

	(Trusted Execution Environment), or protected using strong cryptography.			
C.7	Verify that the firmware apps protect data-in-transit using transport layer security.			
C.8	Verify that the firmware apps validate the digital signature of server connections.			
C.9	Verify that wireless communications are mutually authenticated.			
C.10	Verify that wireless communications are sent over an encrypted channel.			
C.11	Verify that any use of banned C functions are replaced with the appropriate safe equivalent functions.			
C.12	Verify that each firmware maintains a software bill of materials cataloging third-party components, versioning, and published vulnerabilities.			
C.13	Verify all code including third-party binaries, libraries, frameworks are reviewed for hardcoded credentials (backdoors).			
C.14	Verify that the application and firmware components are not susceptible to OS Command Injection by invoking shell command wrappers, scripts, or that security controls prevent OS Command Injection.			
	Level 2/3			
C.15	Verify that the firmware apps pin the digital signature to a trusted server(s).			
C.16	Verify the presence of tamper resistance and/or tamper detection features.			
C.17	Verify that any available Intellectual Property protection technologies provided by the chip manufacturer are enabled.			
C.18	Verify security controls are in place to hinder firmware reverse engineering (e.g., removal of verbose debugging symbols).			
C.19	Verify the device validates the boot image signature before loading.			
C.20	Verify that the firmware update process is not vulnerable			

	to time-of-check vs time-of-use attacks.			
C.21	Verify the device uses code signing and validates firmware upgrade files before installing.			
C.22	Verify that the device cannot be downgraded to old versions (anti-rollback) of valid firmware.			
C.23	Verify usage of cryptographically secure pseudo-random number generator on embedded device (e.g., using chip-provided random number generators).			
C.24	Verify that firmware can perform automatic firmware updates upon a predefined schedule.			
	Level 3			
C.25	Verify that the device wipes firmware and sensitive data upon detection of tampering or receipt of invalid message.			
C.26	Verify that only micro controllers that support disabling debugging interfaces (e.g. JTAG, SWD) are used.			
C.27	Verify that only micro controllers that provide substantial protection from de-capping and side channel attacks are used.			
C.28	Verify that sensitive traces are not exposed to outer layers of the printed circuit board.			
C.29	Verify that inter-chip communication is encrypted (e.g. Main board to daughter board communication).			
C.30	Verify the device uses code signing and validates code before execution.			
C.31	Verify that sensitive information maintained in memory is overwritten with zeros as soon as it is no longer required.			
C.32	Verify that the firmware apps utilize kernel containers for isolation between apps.			
C.33	Verify that secure compiler flags such as -fPIE, -fstack-protector-all, -Wl,-z,noexecstack, -Wl,-z,noexecheap are configured for firmware builds.			
C.34	Verify that micro controllers are configured with code protection (if applicable).			