Technical and solution architects must take care of various artifacts throughout the software development lifecycle. These artifacts help in defining, designing, and implementing the architecture of a system. Some common artifacts that architects typically work with include:

1. **Architecture Vision and Requirements:** Define the high-level goals and requirements of the system, aligning them with business objectives.
2. **Architecture Design:** Create architectural designs, including high-level and detailed designs, diagrams (like UML diagrams), and documentation to communicate the architecture to stakeholders.
3. **Architecture Decisions:** Document key decisions made during the design process, including rationale and alternatives considered.
4. **Architecture Views and Viewpoints:** Develop different views (logical, physical, process, etc.) and viewpoints to address concerns of various stakeholders.
5. **Architecture Patterns and Styles:** Identify and apply appropriate architectural patterns and styles (e.g., layered architecture, microservices, event-driven architecture) to meet system requirements.
6. **Technology Stack Selection:** Choose the right technologies, frameworks, and tools for implementing the architecture based on system requirements, constraints, and industry best practices.
7. **Security Design:** Define security requirements, controls, and mechanisms to ensure the system's Security Design: security and compliance with regulations.
8. **Performance and Scalability Design:** Design the system to meet performance and scalability requirements, considering factors like load balancing, caching, and data partitioning.
9. **Integration Design:** Plan how different components and systems will integrate with each other, including protocols, data formats, and messaging mechanisms.
10. **Deployment and Infrastructure Design:** Define the deployment architecture, including server configurations, cloud services, and network topology.
11. **Operational Considerations:** Address operational aspects like monitoring, logging, and maintenance to ensure the system is reliable and easy to manage.
12. **Quality Attributes:** Specify non-functional requirements such as reliability, availability, maintainability, and usability that the system should meet.
13. **Migration and Evolution Strategy:** Plan how the system will evolve over time, including strategies for migrating data and transitioning to new technologies.
14. **Compliance and Governance:** Ensure that the architecture complies with legal, regulatory, and organizational policies and standards.
15. **Communication and Collaboration:** Facilitate communication and collaboration among team members and stakeholders, ensuring everyone understands and agrees on the architecture.