What is TOSCA?

OASIS TOSCA is domain-specific language (DSL) for defining
  – Portable deployment and automated management of services on a wide variety of infrastructure platforms…
  – using model-based descriptions of services, platforms, infrastructure and data components, along with their relationships, requirements, capabilities, configurations and operational policies.

TOSCA is Declarative
  – As opposed to Imperative
  – Focuses on the WHAT, not the HOW

WHAT can mean …
  – What does the service look like?
    • Expressed using Models
  – What is the desired service outcome (behavior)?
    • Expressed using (declarative) Policies
TOSCA Modeling Constructs support all phases of the service lifecycle

**Run-time**
- Simple constructs to model deployed services
- Typically stored in *Active Inventory*
- Can represent both instantiated services as well as available resources

**Orchestration time**
- Add modeling constructs in support of orchestration
- Typically stored in *Service Catalog*
- Define *Inputs* for deployment-specific configuration

**Design time**
- Types define reusable components
- Typically stored in Component Catalog
- Add “specifications” to define standard collections of attributes for similar components

**DSL design**
- Language defines schema for creating new types
- For all TOSCA entities
Run-Time Modeling Constructs
TOSCA Instance Object Model

Small set of general-purpose modeling constructs
- Service Topologies are graphs that consist of Nodes and Relationships
- Nodes represent managed objects: components in the topology structure that need to be managed
- Relationships describe dependencies between nodes (modeling containment and adjacency)

Nodes are essentially collections of properties and attributes
- Name-value pairs
- Property values reflect desired state (requested by orchestrators and other automation tools)
- Attribute values reflect actual state
- All properties have implicit attributes

Built-in functions define how properties are propagated between nodes along relationships
- E.g. to propagate IP address from network port to logical termination point using that port
  - get_attribute
  - get_property
The TOSCA Instance Object Model is sufficiently generic that it can/should be used to model all entities in Active Inventory
– Services and service components created through orchestration
– Services made available as resources to other services
– Available infrastructure resources (onboarded or discovered)
  • Physical resources
  • Logical resources

TOSCA does not distinguish between resources or any other service components
– Resource is a “role”, not an intrinsic characteristic of an entity

TOSCA Instance Object Model does not include domain-specific “classification”
– Such classifications are specified in TOSCA Type System
TOSCA is based on the Component Pattern

TOSCA supports only one single level of decomposition
- Topologies are composed of nodes
- Keeps composition in TOSCA extremely simple

Further decomposition is provided by the Component Pattern
- A single node in one topology can be modeled as/substituted by an entire “sub” topology consisting of other nodes

Benefits of Component Pattern
- Avoids the need to define service components as atomic or composite
  - Whether an entity is atomic or composite is not an intrinsic characteristic of that entity (even atoms are not atomic)
  - But rather it is a reflection of where the modeler stops decomposing
- Supports abstraction
  - Drilling-down into lower-levels of abstraction
  - Rolling-up into higher-levels of abstraction
- Allows dynamic reconfiguration of service topologies at run-time
  - By replacing different sub-topologies that substitute for the same node
- Preferred mechanism for composing large systems from modular building blocks
Composition – Substitution of Abstract Services

Service Topology 1

Cloud Application (Topology)
- Java Application
- Web Application Server
- SQL Datastore
- Monitoring Service (Abstract)

Service Topology 2

Monitoring Service (Topology)
- Collector
- Monitoring Framework
- Logger
- Analytics Service (Abstract)

Service Topology 3

Analytics Service (Topology)
- Analytics Engine
- Hadoop
TOSCA Modeling Constructs in Support of Orchestration

Templates
- Parameterized constructors from which services and service components are created
- Include (or reference) all necessary configuration and Infrastructure requirements

Node Template Modeling Constructs
- Requirements and Capabilities

Node and Relationship Template Modeling Constructs
- Interfaces, Operations, and Artifacts

Service Template Modeling Constructs
- Inputs
  - Allow users to provide deployment-specific service characteristics
- Policies and Groups
Requirements and Capabilities

Node templates include Capabilities and Requirements

- Relationships link Node Requirements to Node Capabilities
- Requirement anchors the source of a relationship
- Capability anchors the target of a relationship

Node Templates can have dangling requirements

- Requirements that are not fulfilled by other nodes in the template
  - Expressing intended target capability
  - (Optionally) expressing Node Filter that further constraints nodes and capabilities with which to satisfy the requirement
Fulfilling Requirements at Orchestration Time

Dangling requirements enable modular service design
- Allow dynamic decisions about establishing relationships
  - Made at deployment time
- Rather than having to statically specify relationships in service templates
  - At service design time
- Avoids model proliferation

Service template specifies requirements

Entities in Active Inventory expose capabilities

Orchestrator matches capabilities to requirements

Orchestrator

Active Inventory
TOSCA Interfaces, Operations, and Artifacts

Nodes and Relationships have **Interfaces**
- Used by orchestrator to operate on nodes and relationships
- When orchestrator runs workflows

**Interfaces Types define**
- *Operations* that can be invoked on nodes or relationships by the Orchestrator
- *Inputs* that need to be provided to each operation
- *Outputs* that specify which node and relationship properties are affected by each operation

Artifacts Implement Operations
- Service template designers provide operation implementations through (typed) artifacts
  - Bundled in Cloud Service Archive (CSAR) file
  - Or, stored in external repository and referenced by service templates

**TOSCA includes normative interface types**
- Normative interface types are associated with normative node and relationship types
Normative Interface Types

Normative interface type for **managing lifecycle of nodes**

- TOSCA Orchestrator
- TOSCA Lifecycle Operation
- TOSCA Node State

- **create()**
  - initial
  - creating
- **configure()**
  - created
  - configuring
- **start()**
  - configured
  - starting
  - started

Normative interface type for **configuring relationships between nodes**

- TOSCA Orchestrator
- TOSCA Lifecycle Operation
- TOSCA Node State

- **create()**
  - initial
  - creating
  - configure_source()
  - configure_target()
- **configure()**
  - post_configure_source()
  - post_configure_target()
- **start()**
  - add_target()
  - add_source()
  - remove_target()
  - target_changed()
  - configured
  - starting
  - started
TOSCA Workflows are Declarative

Automatically generated by orchestrator
- Based on normative interface types for lifecycle management of nodes
- Based on normative interface types for configuration of relationships

Generating declarative workflows
- For each node in the topology, create a sequence of calls to interface operations
  - To move node into the desired state
- For each relationship in the topology, create a sequence of calls to interface operations
  - To move relationship into the desired state
- Ordering of calls determined by weaving of
  - Interface operations defined on nodes
  - Interface operations defined on relationships for those nodes
TOSCA Type System
TOSCA Type System

TOSCA Types add Domain-Specific Specifications to the generic TOSCA modeling constructs

- Define the set of named properties and attributes (and associated data types) that must be supported by all entities of a given type
- Define the set of named capabilities (and associated capability types) that must be supported by all nodes of a given type
- Define the set of named interfaces (and associated interface types) that must be supported by all nodes and relationships of a given type

All TOSCA Entities have Types

- Promotes re-use. Reusable building blocks (e.g. VNFs) can be packaged as a type
- Enables validation (correctness and completeness of service templates)

TOSCA Types support Inheritance

- To extend and/or further constrain parent types
TOSCA Profiles

A TOSCA profile is a collection of types for a specific application domain

TOSCA Simple Profile for YAML
- In addition to defining the grammar of the language, also defines a set of common base types that can be used by cloud application designers
  - Like a “standard library”

TOSCA NFV Profile
- Collection of TOSCA types that encode the ETSI NFV IFA information models
TOSCA Resources
TOSCA Resources – Learn More

TOSCA Technical Committee (TC) Public Page (*TC approved updates on plans, documents, resources, and more*)

OASIS TOSCA Wiki
— https://wiki.oasis-open.org/tosca/

OASIS TOSCA LinkedIn Group: *(latest news, community and eco-system updates, etc. Join now to stay informed!)*
— https://www.linkedin.com/groups/8505536

OASIS YouTube Channel, TOSCA Playlist

Contact the Technical Committee Co-Chairs (also part of TOSCA TC Panel):
— Paul Lipton, paul.lipton@ca.com; Chris Lauwers, lauwers@ubicity.com