Service Level Agreements and Management

Mark Carlson, Sun / Policy WG Chair
Karl Schopmeyer, Open Group / Application WG Chair
Andrea Westerinen, Cisco / DMTF VP of Technology
Agenda

• Terminology
• Intersection between SLAs and Management
• CIM Schema in Support of SLAs
• New CIM Work
• Applications Use Case
• Storage Use Case
Terminology

• Service Level Agreement
  – Documented result of a negotiation between a customer/consumer and a provider of a service, that specifies the levels of availability, serviceability, performance, operation or other attributes of the service. [RFC2475]

• Service Level Objective
  – Partitions an SLA into individual metrics and operational information to enforce and/or monitor the SLA. It is a set of parameters and their values. The actions of enforcing and reporting monitored compliance can be implemented as one or more policies.
Intersection of SLAs and Management

- Formalize the concept of SLAs and their metrics
  - Standard and “exchangeable” definitions
- Monitor and programmatically enforce SLAs
  - Within and across boundaries
  - Across vendors and products
- Decompose high-level, business SLAs to device specific configurations and monitored parameters
  - View the SLAs as goals
- Define identity information to support per-customer billing and SLAs
  - Within and across boundaries
- Define and enforce application-specific SLAs
SLAs and Semantics

• SLAs require **understanding** of syntax and semantics
  – Describing what is managed/monitored (and when) and what happens if problems occur (e.g., policy)
  – Similar to The Economist’s article (July 2002) on English as the world’s language and humans’ ability to generalize abstract concepts

• Can never go from general to specific without general semantic framework
  – What does it mean to understand a French poodle without understanding a dog?
**Existing CIM Schema – An Example**

- **System** (See Core Model)
- **ApplicationSystem** (See Application Model)
- **DatabaseSystem**
- **CommonDatabase**
  - InstanceId: string (key)
  - DatabaseVersion: string
  - SizeAllocated: uint32
  - SizeUnits: uint16 (enum)
  - LastBackup: datetime
- **Service** (See Core Model)
- **DatabaseService**
  - StartupTime: datetime
  - OperationalStatus: uint16 (enum)
  - LastStatusChangeTime: datetime
  - ConnectionLimit: uint32
- **ServiceAvailableToDatabase**
  - AvailableState: uint16 (enum)
  - OtherAvailableState: string
  - ActiveTime: datetime
- **AssociatedDatabaseSystem**

Connections:
- 1..n between DatabaseSystem and AssociatedDatabaseSystem
- * between CommonDatabase and ServiceAvailableToDatabase
- * between ServiceAvailableToDatabase and DatabaseService
Existing CIM Schema – Metrics

```
ManagedElement

BaseMetricDefinition
- Id: string {key}
- Name: string
- DataType: uint16 {enum}
- Calculable: uint16 {enum}
- Units: string
- BreakdownDimensions: string[]

BaseMetricValue
- InstanceId: string {key}
- MetricDefinitionId: string {required}
- MeasuredElementName: string
- TimeStamp: datetime
- Duration: datetime
- MetricValue: string
- BreakdownDimension: string
- BreakdownValue: string
- Volatile: boolean

MetricForME

MetricDefForME

MetricInstance
```

(See Core Model)
New CIM Work in Policy

- Trigger
- Generic Condition - Based on query format
- Generic Action – Invocation of schema method or CIM Operation
- Issues with maintaining context
- Is a language an alternative?
Policy, Pigs and Ponder

• **Hard problem:** "If at least two of a farmer’s pigs are squealing, then he/she must feed one of the pigs that isn't squealing."

• **Another constraint:** Graft policy onto an information model by introducing references to the suitable objects in the model
  - PigKeeper and Pig classes, KeeperKeepsPig association, SquealingPig notification, and FeedPig method on the PigKeeper
Policy, Pigs and Ponder

- oblig feedNotSquealingPig {
  - On CIM_SquealingPigIndication(Pig1) &&
    CIM_SquealingPigIndication(Pig2);
  - // 2 squealing pigs
  - subject /pigkeepers;
  - // i.e., all pig keepers
  - do (t = self.KeeperKeepsPig->reject (isSquealing)) ->
    self.feedPig(t);
  - when Pig1!=Pig2 // they are different pigs
  - and Pig1.KeeperKeepsPig =
  - Pig2.KeeperKeepsPig = 'self';
  - // and I am the keeper
Thoughts on SLAs and SLOs

ServiceLevelAgreement
- InstanceID: string [Key]
- ObjectiveOfSLA

ManagedElement (Abstract)
- Caption: string
- Description string

ObjectiveAppliesToElement

PolicySet

ServiceLevelObjective
- InstanceID: string [Key]

MetricInObjective

ManagedObject
- Id: string [key]
- DefId: string [key, propagated]
- Time: datetime
- Value: string

BaseMetricDefinition
- Id: string [key]
- Name: string
- DataType: uint16
- Calculable: uint16
- Units: string

ManagedElement

ThresholdMetricDefinition
- ThresholdValue: string

MetricDefForME

DefinitionForValue

MetricForME

BaseMetricValue
SLOs and Policy Conditions

PolicySet (Abstract)
- PolicyDecisionStrategy: uint16
- PolicyRoles: string[]
- Enabled: uint16

PolicyCondition (Abstract)

ConstraintCondition

PolicyRule
- CreationClassName: string[key]
- PolicyRuleName: string[key]
- ConditionListType: uint16
- RuleUsage: string
- Mandatory: boolean
- SequencedActions: uint16
- ExecutionStrategy: uint16

ServiceLevelObjective
- SLOComponent
- PolicyRule
- SLO
### SLA Payment Action

**PolicyAction (Abstract)**

- **SystemCreationClassName**: string[key]
- **SystemName**: string[key]
- **PolicyRuleCreationClassName**: string[key]
- **PolicyRuleName**: string[key]
- **CreationClassName**: string[key]
- **PolicyActionName**: string[key]

---

**SLAPaymentAction**

- **PaymentType**: uint16
  - (Enum: Absolute, Relative)
- **PaymentCurrency**: string
- **PaymentAmount**: sint32
- **PaymentPercent**: sint8
Service Level is the User Experience

• Services levels are a measure of the user experience
  – How much
  – How fast
  – How available
  – How reactive to problems

• What will keep the user happy
Application Management and Service Level

- Typical runtime Service Level Parameters for applications
  - User perspective on performance
    - Interactive responsiveness
      - Transaction Response time / Time to accomplish task for client
      - Throughput / How many users can simultaneously maintain the above response times or How many things can it do in some unit of time.
    - Batch turnaround
    - Critical deadlines
  - Availability
    - Percentage of time service is available
    - Maximums on lack of service / How long is it down
- Other non-runtime SLA issues
  - Recoverability, Problem responsiveness, Affordability
Holy Grail is Mgmt Not Just Monitoring

Output - SLA reports

Service Level Measurement

Analyse → Monitor → Resources

SLOs → Analyse → Control

Policy → Analyse And Decide

SLOs → Monitor → Resources
SLA Domains

- Networks
- Databases
- Server and Client systems
- Applications
- Transactions
Goals of Application Measurement

• Provide monitors for
  – Service Level management
    • Need both information and controls so that analysis can be done, and decisions made and implemented
  – Business and Business Process management

• Provide the application controls for
  – Fault determination
  – Performance characteristic attribution
  – Appl monitor, management and manipulation in terms of the application’s components, aggregated into a whole to support SLA
• Monitor to provide information for SLA reporting
• Provide controls for SLA tuning
• Provide means to find why you are not meeting SLA.

It does not do much good to know that response time is too long if you have no way to know why.

It does even less good to have a response time SLO and not be able to measure it on the system.
Applications Are Complex Aggregations

• Complex collection of software components
• Multilayered functionally
  – Ex. Presentation, application, database
• Dynamically assembled into the “application”
• GOAL
  – Model the components as viewed in runtime AND their interactions
  – Aggregate information into the whole
  – Disaggregate information from the whole back into the components
Monitoring for SLO information

- End-End Response and Availability Measurements
- Server and Application, component performance, response, performance, resource utilization measurements

Response time measurement is the Unit of Work model

App and component definition, performance and resource measurements - The Runtime Application Model
Modeling the Transaction – Unit of Work

• Objectives
  – Measure a time interval
  – Identify the transaction
  – Identify the application
  – Provides information for correlation of multiple measurements
  – Provides information to understand component UofWork (parent/child units of work)
  – Provides metric information places for resource, etc. information
  – Marry with the instrumentation technology - ARM
Unit of Work Metric Model

Managed Element

Logical Element

UOW Definition

Logical Element performs UOW

CIM_Logical Element

Logical Element UOW Definition

UOW Metric Definition

UOW Metric

Metric Definition

Managed Element

UOW

Sub UOW

Started UOW

W

Sub UOW

Metric Value
Unit Of Work Definition

• Defines a type of work
• Examples:
  – Update account balance
  – Backup file system
  – Query data server
  – Execute subroutine/method
UOW Definition Properties

• **ID** - 16 bytes long [key]
  - Use of OSF UUID is recommended
    • MAC address (or substitute)
    • Current date and time
    • Counter to handle high-frequency allocations
    • A clock sequence and related state to handle the retrograde movement of clocks.

• **Name**

• **Context** (such as application name)
Unit of Work

- Represents a UOW that has started (and maybe completed) executing
- Always associated to its definition
  - Weak association with propagated key
  - This provides separation of the namespace to avoid collisions
Unit of Work Measurements

- Response time or elapsed time (if still executing)
- Status
  - Active
  - Suspended
  - Completed (with unknown state)
  - Completed Good
  - Completed Failed
  - Completed Aborted
- Metrics (in a separate class)
The Application Runtime Model

- Define the application system runtime logical structure
- Define the application component structure
- Define the interactions between the applications and components
- Define relation to resources
- Define relationships to software elements
- Define the relationship between the application and UofWork
- Model for information about utilization, resources, etc.
- Aggregation of information from components to the whole
Application Runtime Model Concepts

- **System Sub-Model**
  - ApplicationSystem <<System>>
    - Parameter, Status, InnerError, InnerTimeBound, InnerAmountBound
  - DistributedAppSystem <<System>>
  - LocalAppSystem <<System>>

- **Function Sub-Model**
  - ApplicationService <<SoftwareService>>
    - Dependencies: CIM_Service, NamedCommunication (SAP), LogicalStorage
    - Uses: Statistics, Configuration, Logs
    - Data Flow:Implicit: Data Flow
  - Resource <<Resource>>

- **Structure Sub-Model**
  - NamedComponents <<CodeComponents>>
    - Includes: s. document

- **Data Sub-Model**
  - UnitOfWork <<Action>>
    - Performs: s. document

- **Hierarchy**
Issues and Questions Today

• Note that the runtime model is in development today
  – First components (related to ApplicationSystem) in V2.8 preliminary

• Next -> The functional submodel
SLAs, SLOs and Policy

Service Level Agreements

Incorporated into

Service Level Objective

Service Level Metric

Information Model standards

Element Policy

Policy Management
(Management of Policies)

Policy-Based Management
(Automation)

manually

Enforces

Create, Update, Maintain

Contractual -- based on business process and requirements

Service Level Policy Specification

Common expression and metrics

Policy Refinement
• decompose service into elements

Model, schema, access methods, error detection, recovery, creation, deletion, modification, security, consolidation (synergies, conflicts), deployment
From Data to Configuration

Data Policy

- Business Requirements
  (Test Data, Business Continuance Data, Operations, Legal)

Service

- Data, Attributes
- Service, Capabilities

Refined

- Data, Attributes
- Service, Capabilities

Into

- Data, Attributes
- Service, Capabilities

Deployment

- Data, Attributes
- Service, Capabilities

Data, Attributes

Service, Capabilities

Element, Configuration

Deployment

Data, Attributes

Service, Capabilities

Element, Configuration

Deployment

Data, Attributes

Service, Capabilities

Element, Configuration

Deployment

Data, Attributes

Service, Capabilities

Element, Configuration

Deployment
Administrator creates Element Policy Instance, conveying desired behavior.

Provider enforces policy directly For the device
Administrator creates Service Level Policy Instance, conveying desired service levels

Policy Manager interprets service level policy as refined element policies

Policy manager creates, deploys element policies to support the achievement of specific service level objectives