**Comments and Suggested Revisions**

**Towards the SCOS Policy Section**

**of the 802.22.3 Standard Draft v1.0**

*Nilesh Khambekar*

Date: March 1st, 2017 [Comment version 1.0 ]

* + 1. Comments

Following are the comments on SCOS Policy section 5.3.6 in the standard version 1.0

1. There can policies towards SSDs, SSM, Sensing Data Manager, SCOS Platform Administration, and SCOS User Management.
2. SCOS policy high level architecture needs to be provided.
3. SCOS Policy structure and specification needs to be added.
4. SCOS Policy examples should be provided.
5. SCOS Policy application procedure should be described.
	* 1. Suggestions

Following draft attempts to address the above mentioned comments regarding SCOS policy section 5.3.6 in the standard draft v1.0

* + 1. SCOS Policy

A policy layer in the SSM at the northbound and southbound API will ensure that the SSM is operated within requirements of a local authority (national regulator, law enforcement, military, etc).

The policy on the southbound interface will determine, based on the USER type, (e.g. how a central authority can define what kinds of sensing can be done in what bands, what data governance rules there are, etc

-- resource allocation – what kinds of users are authorized to request resources from the sensor network and in which priority (i.e. if a sensing network is resource constrained, who gets first dibs on the sensors)

Policy on the northbound interface (SSM>DBstore) will have rules for how sensing data may be distibuted, and data storage policies. This would include which scan data takes priority if local storage in the store&forward buffer is running out due to a failed transmission link, and how long certain USER data is allowed to be stored in the local store&forward buffer.

Based on SCOS platform architecture, the SCOS policy is categorized based into following categories:

* Spectrum Sensing Manager (SSM) Policy
* Spectrum Sensing Device (SSD) Policy
* Sensing Data Manager (SDM) Policy \*\*SDM=Data-Put-manager
* SCOS Platform Administration (SPA) Policy \*\*Names/Acronyms temporary
* SCOS Platform User (SPU) Policy \*\*Names/Acronyms temporary

The SCOS policy is expressed using JSON. Following subsection provides details about the schema of SCOS policy.

* + - 1. The SCOS policy schema:

Each SCOS policy is associated with a policy-name, policy-namespace, policy-category, policy-scope, optional policy-description, and one or more statement(s).

Following figure shows the structure of a SCOS policy.

**SCOS Policy**

Top-level attributes

Statement

Statement

**Statement**

StatementId

Policy-Action

Actor

Resource

SCOS-Task

Condition-Block

Figure 2. High-level structure of the SCOS Policy

Example:

{

# "Version": "2017-02-15",

 "Policy": {

 "namespace": "OperatorFoo",

 "name": "Calibration-access-control",

 "description": "This policy added by FooAdmin On this date."

 "Policy-type": "SPA-Policy",

 "Policy-Action": "permit",

"Resource": "Foo:Sensors::\*"

"SCOS-Task": "Calibration-operation"

"Scope": "Sensor-management:"

 "Condition": {

 "equals" : {

 "Actor" : "FooAdminRole"

 }

 }

}

The default namespace is global. Specific-Namespaces could be used to restrict the application of policy within a certain context. Namespaces avoids name collisions and enables to identify actors or resources uniquely (when the names have been reused across namespaces).

### **SCOS Policy Statement**

Each statement specifies certain action. Different categories of SCOS-Policies are associated with different actions.

A statement may also have optional attributes that identify the context of applying the policy. These attributes allow specifying a fine-grained policy. Example context-attributes are Actor, Resource, SCOS-Task, time, frequency, and location.

A statement has an optional condition-block. The action is performed only when the condition(s) are matched.

### **Actor**

An actor is an entity that wishes to use the SCOS platform. An actor could be specified in terms of role, user, or a user-group.

### **User**

An user is an individual actor with specified name and access-credentials.

### **User-group**

A user-group is a logical collection of users. A user-group is specified with name and access-credentials.

### **Role**

A role is specified with a name. The role could be associated with specific SCOS services/functionality. A role could also be associated with privilege of various users of SCOS platform.

An user (or a user-group) is associated a role.

### **Resource**

SSD and Sensing data are two prime resources within SCOS platform.

### **Resource-group**

Multiple SSDs could be grouped together to jointly specify policies for using the SSDs. SSDs could possibly be grouped based on various attributes such as location, SSD-hardware-type, SSD-software-type.

### **Namespace**

Actors or resources are associated with a namespace. This avoids name collisions and enables to identify actors or resources uniquely (when the names have been reused across namespaces).

### **SCOS-Task**

An SCOS-task represents a specific sensing-task within SCOS platform issued by a particular Actor on particular resources. Additionally, a policy may specify pre-defined SCOS operations in the SCOS-Task. These predefined SCOS-Tasks include: Scanning, Calibration, Storage, and Transmission.

### **Task-group**

Multiple tasks could be grouped together for convenience in specifying policies. For example, various tasks that can be performed towards sensor-management for a particular SSD operator could be grouped together and referred to in the SCOS policy. Similarly, sensing data management related tasks could be grouped together for precisely and conveniently specifying sensing-data-management related policies.

### **Conditions**

A condition is specified with a triplet of field(key), conditional-operator, and value. Condition is optional within a statement.

A condition evaluates whether a field meets certain criteria. Following table identifies various conditional operators.

|  |  |
| --- | --- |
| **Conditional-operator Name** | **Syntax** |
| equals | "equals" : "<value>" |
| Like |  "like" : "<value>"  |
| Contains |  "contains" : "<value>" |
| In | "in" : [ "<value1>","<value2>" ]  |
| Exists |  "exists" : "<bool>"  |
| LessThan |  "lessthan" : "<value>"  |
| GreaterThan | "greaterthan" : "<value>"  |
| LessThanEquals |  "lessthanequals" : "<value>"  |
| GreaterThanEquals | "greaterthanequals" : "<value>"  |

### **Logical Operators**

Logical operators enable to manipulate or combine multiple conditions. Following table specifies the logical operators.

|  |  |
| --- | --- |
| **Logical operator**  | **Syntax** |
| Not | “not”: {<condition>}  |
| AllOf |  "allOf" : [ {<condition>},{<condition>}]  |
| AnyOf |  "anyOf" : [ {<condition>},{<condition>}]  |

### **Aliases**

Aliases add convenience. Using aliases, multiple users can be combined together or multiple resources can be combined together to be referred in the SCOS policy . Furthermore, multiple tasks can be combined using task-groups.

Furthermore, locations could be specified using aliases to capture latitude, longitude, and altitude. A group of frequencies could also be combined using aliases. A group of time-slots also could be combined using aliases.

* + - * 1. SSM Policy Schema

Each SSM policy has following required fields: PolicyName, PolicyScope, PolicyType, and PolicyAction.

Optional fields include: Policy-Description, condition-block, Actor, Resource, and SCOS-Task.

With Policy-Action ‘set’, SSM attribute and value(s) could be specified.

* + - * 1. SSD Policy Schema

Each SSD policy has following required fields: PolicyName, PolicyScope, PolicyType, PolicyAction, and Resource.

Optional fields include: Policy-Description, condition-block, Actor and SCOS-Task.

Policy-Actions: Set, Permit, Deny, Calibrate, Scan,

With Policy-Action ‘set’, SSD attribute and value(s) could be specified.

* + - * 1. SDM Policy Schema

Each SDM policy has following required fields: PolicyName, PolicyScope, PolicyType, and PolicyAction, and SCOS-Task.

Optional fields include: Policy-Description, condition-block, Actor and Resource.

Policy-Actions: Set, Permit, Deny, Transmit-Sensing-Data, Store-Sensing-data, Discard-Sensing-data

With Policy-Action ‘set’, SDM attribute and value(s) could be specified.

* + - * 1. SPU Policy Schema

Each SPU policy has following required fields: PolicyName, PolicyScope, PolicyType, PolicyAction, and Actor.

Optional fields include: Policy-Description, condition-block, SCOS-Task, and Resource.

Policy-Actions: Set, Permit, Deny

With Policy-Action ‘set’, SPU attribute and value(s) could be specified.

* + - * 1. SPA Policy Schema

Each SPA policy has following required fields: PolicyName, PolicyScope, PolicyType, and PolicyAction.

Optional fields include: Policy-Description, condition-block, SCOS-Task, Actor, and Resource.

Policy-Actions: Set, Permit, Deny,

With Policy-Action ‘set’, SPA attribute and value(s) could be specified.

Policy file: It is envisioned in the first version of this standard that the SSM stores a policy file which is installed manually by Sensing Operator (through mechanism such as SSH and update pull, or remote SCP).

* + - 1. Policy Evaluation

Whenever an SCOS API needs to be executed, SSM needs to confirm if the action is permitted by evaluating related policies.

There exist three scopes for SCOS policies: Sensing management scope, Sensing-data management scope, and Sensor-management scope. Depending on the API, policies in the appropriate scope are looked up.

The second step is ensure that the actor is authorized to perform tasks on the resource. A specific accept policy or default-accept policy should be match for the user, user-group, or role.

The final step is ensure if the resource permits the intended task. A specific accept policy or default-accept policy should be match for the resource, or resource-group.

Security Considerations

<Nilesh input>

* + - 1. SCOS Policy Examples
				1. SSD Policy

*Set sensitivity to -114 dBm task frequency UHFBand*

PolicyID: <generated>

PolicyName: SCOSMinSensitivityRule

Policy-Category: SSD-Policy

PolicyDescription: It applies to all SSDs within the SCOS operational region.

Policy-Action: Set

Sensitivity: Value

Frequency: value

Discuss: Should frequency be within the context-block or condition-block?

* + - * 1. SSM Policy

*Set scheduling minimum slot duration*

PolicyID: <generated>

PolicyName: SSMMinSensingSlotDuration

Policy-Category: SSM-Policy

PolicyDescription: It specifies the minimum slot duration for sensing task. The value is in seconds.

Policy-Action: Set

Min-sensing-slot-duration: Value (seconds)

*Note:* Fine-grained policy could be specified for a particular resource (SSDs) or sensing-tasks.

*Set sensing behavior for prioritized scan*

PolicyID: <generated>

PolicyName: SSM-Prioritized-Scan-Behavior

Policy-Category: SSM-Policy

PolicyDescription: It specifies whether existing scan should be suspended if a prioritized scan-request is received.

Policy-Action: Set

Prioritized-scan-enabled: true

Condition-block: if wait-time greaterthan value (in seconds)

*Note:* Condition-block is optional. Condition-block can be used to specify a condition when existing scans can be suspended.

* + - * 1. SDM Policy

*Set max-data-storage-duration at SDM*

PolicyID: <generated>

PolicyName: SDMMaxStorageConfig

Policy-Category: SDM-Policy

PolicyDescription: It specifies how long SDM can hold the sensing data.

Policy-Action: Set

Max-data-storage-duration: Value (seconds)

*Note:* Optionally specify task or SSDs or SDS. The value is in seconds.

*Discard sensing-data for <scan-task-L-band-User-Jim> if sensing-data is unqualified.*

PolicyID: <generated>

PolicyName: User-Jim-L-Band-Discard-Data-Policy

PolicyDescription: If sensing data does not meet the criteria specified in the sensing task, discard the data.

Policy-Category: SDM-Policy

Policy-Action: discard-data

SCOS-Task: scan-task-L-band-User-Jim

Condition-block: sensing-data-quality is ‘unqualified’.

*Note: The condition-block identifies when the operation is performed. Here, sensing-data has attribute sensing-data-quality. The condition is satisfied when the attribute’s value is unqualified. The sensing-task is identified pre-defined using name-alias. Optionally, the policy could be made more specific for certain time, and location attributes*.

* + - * 1. SPA Policy

*Enable SSM-Proxy device usage in SCOS system.*

PolicyID: <generated>

PolicyName: Enable-SSD-Proxy-Config

Policy-Category: SPA-Policy

PolicyDescription: Enable SSM-Proxy devices in the SCOS platform.

Policy-Action: Set

SSM-Proxy-Enabled: Boolean-Value

*Note: Optionally, the policy could be made more specific for certain frequency, time, and location attributes*.

* + - * 1. SPU Policy

*Deny scan operation for User-Foo in the military bands*

PolicyID: <generated>

PolicyName: MilitaryBandScanRestrictionPolicy

Policy-Category: SPU-Policy

PolicyDescription: Deny certain users/roles/groups to scan in certain bands.

SCOS-Task: scan

Actor: User-Foo

Policy-Action: Deny

frequency: X-band

*Note: The actors could be specified with user/role/user-group. The frequency bands could be pre-defined using name-aliases. Optionally, the policy could be made more specific for certain time, and location attributes*.

**SDS Policy**

*Send sensing-data for <scan-task-L-band-User-Jim> to data-store <FooStore3>*

PolicyID: <generated>

PolicyName: User-Jim-L-Band-DataStorePolicy

Policy-Category: SDS-Policy

PolicyDescription: Configure data store for a scan request.

Policy-Action: Transmit-sensing-data

SCOS-Task: scan-task-L-band-User-Jim

Resource: FooStore3

*Note: The data-store is specified with resource on which operation is done. The sensing-task is identified pre-defined using name-alias. Optionally, the policy could be made more specific for certain time, and location attributes*.