Intel

ISC Manager Plugin Integration

Developer Guide

Table of Contents

[1 Revision History 2](#_Toc460157384)

[2 Introduction 3](#_Toc460157385)

[3 VNF Manager Plugin Integration 4](#_Toc460157386)

[4 Prerequisites 4](#_Toc460157387)

[4.1 Skill Requirements 4](#_Toc460157388)

[4.2 Software Requirements 4](#_Toc460157389)

[5 Downloading SDK 5](#_Toc460157390)

[6 Overview of SDK API’s 5](#_Toc460157391)

[6.1 Manager Information API (ApplianceManagerApi) 6](#_Toc460157392)

[6.2 Manager Notification APIs 6](#_Toc460157393)

[6.2.1 ManagerCallbackNotificationApi, ManagerWebSocketNotificationApi 6](#_Toc460157394)

[6.2.2 IscJobNotificationApi 7](#_Toc460157395)

[6.3 Manager Device API (ManagerDeviceApi) 7](#_Toc460157396)

[6.3.1 Functionality Applicable for Agentless deployments 7](#_Toc460157397)

[6.3.2 Functionality which is applicable for Agent based deployments only 8](#_Toc460157398)

[6.4 Manager Policy Mapping API (ManagerSecurityGroupInterfaceApi) 8](#_Toc460157399)

[6.5 Manager Security Grouping API (ManagerSecurityGroupApi) (If applicable) 9](#_Toc460157400)

[6.6 Manager Policy API (ManagerPolicyApi) 9](#_Toc460157401)

[6.7 Manager Domain API (ManagerDomainApi) 9](#_Toc460157402)

[7 REST APIs Exposed by ISC 9](#_Toc460157403)

[7.1 Notification API 9](#_Toc460157404)

[7.1.1 Request 10](#_Toc460157405)

[7.1.2 Response 10](#_Toc460157406)

[7.2 Propogate Manager file API(Agent Based deployment only) 10](#_Toc460157407)

[7.2.1 Request 11](#_Toc460157408)

[7.2.2 Response 11](#_Toc460157409)

[7.3 Query VM Information API 11](#_Toc460157410)

[7.3.1 Request 12](#_Toc460157411)

[7.3.2 Response 12](#_Toc460157412)

[8 Packaging 14](#_Toc460157413)

[8.1 Manager Plugin 14](#_Toc460157414)

[8.2 Appliance Packaging 14](#_Toc460157415)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Rev | Author | Description | Date |
| 0.1 | Arvind Nadendla | Initial Version. | 04/8/2016 |
| 0.2 | Arvind Nadendla | Add flag to support managers which do not provide Device grouping.  Refactored device API to Security Group api and Security Group Interface API.  Support agentless devices. | 6/2/2016 |
| 0.3 | Emanoel  Xavier | Support manager which do not support Policy Mapping. | 8/16/2016 |
| 0.4 | Arvind  Nadendla | Allow syncing security group member’s information with the manager.  Allow multiple files as part of bootstrap information. | 8/28/2016 |

# Introduction

Network Functions Virtualization (NFV) envisages the implementation of NFVs as software-only entities that run over the NFV Infrastructure (NFVI). Figure below illustrates the high-level NFV framework as referenced by ETSI. As such, three main working domains are identified in NFV:

* Virtualized Network Function (VNF), as the software implementation of a network function which is capable of running over the NFVI.
* NFV Infrastructure (NFVI), including the diversity of physical resources and how these can be virtualized. NFVI supports the execution of the VNFs.
* NFV Management and Orchestration, which covers the orchestration and lifecycle management of NFV’s that support the infrastructure virtualization. NFV Management and Orchestration focuses on all virtualization-specific management tasks necessary in the NFV framework



The NFV framework enables dynamic construction and management of VNF instances and the relationships between them regarding data, control, management, dependencies and other attributes. To this end, there are at least three architectural views of VNFs that are centered on different perspectives and contexts of a VNF. These perspectives include:

* a virtualization deployment/on-boarding perspective where the context can be a VM,
* a vendor-developed software package perspective where the context can be several inter-connected VMs and a deployment template that describes their attributes,
* An operator perspective where the context can be the operation and management of a VNF received in the form of a vendor software package.

# VNF Manager Plugin Integration

ISC (Intel Security Controller) Manager Plugin SDK implementation is required so that ISC can interact with third party VNF Manager Software and take advantage of ISC’s orchestration and management capabilities.

The SDK contains a bunch of interfaces which describe broad functionality. The plugin will provide the concrete implementations of this functionality by bridging the gaps between the manager and ISC concepts.

ISC also exposes REST endpoints which the manager can use to notify of changes on the manager, to quarantine a specific end point and other actions.



The SDK approach allows the security controller to be agnostic about the protocol/type of connection the manager prefers.

# Prerequisites

In order to make use this of this integration, you will need the following:

## Skill Requirements

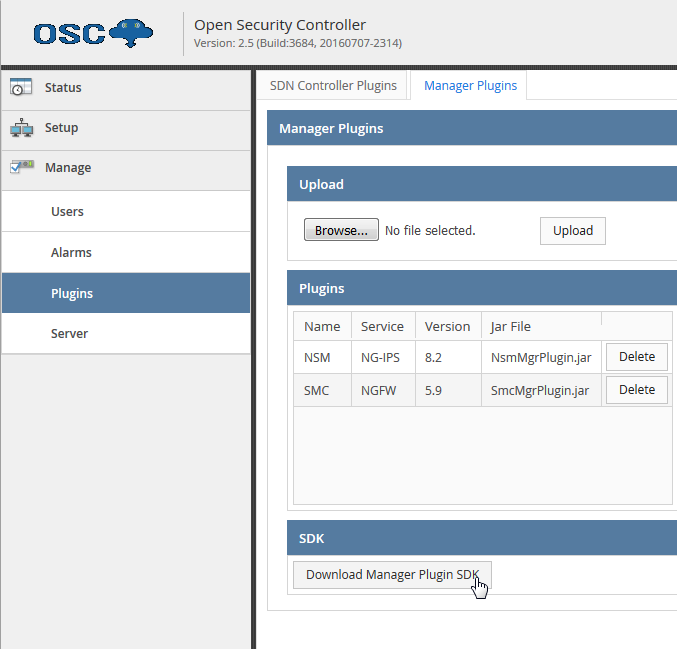
* Experience with Java (J2SE) development.
  + Experience with creating and using Jars.
  + Understanding of Java inheritance.
  + Should know how to import packages and libraries.
* Basic Networking knowledge:
  + Knowledge of IP address, DNS and NAT environment
  + Understanding of web communication.
* Experience implementing REST server and client side both.
  + Implementing web services using REST
  + Consuming REST APIs exposed by this SDK.

## Software Requirements

* Java(JDK)(1.7+)
* Eclipse (preferred) or any other JAVA IDE

# Downloading SDK

* You can download this SDK through ISC UI:
  + Login to ISC UI
  + Navigate to “Mange” menu item
    - Select “Plugins” and Go to “Manager Plugins” tab
    - Click on “Download Manager plugin SDK” button as shown in the image below:



# Overview of SDK API’s

**Agent Based**

In the following sections of the document we talk about agent based deployments which basically mean all the VNF deployed by OSC will have an OSC agent running on the VNF which responds to calls for agent status, Signature file propagation etc. This support is deprecated and will be removed in future versions of OSC

**Agentless**

For Agentless deployments, the assumption is that all the functionality for agent based deployment will be available through the manager. These API’s are under active development and subject to change.

The API’s exposed by the SDK can be broadly classified into the following

## Manager Information API (ApplianceManagerApi)

This is the basic entry point for all the interactions going to the manager. It details the manager information and capabilities supported by the manager.

The information here at a high level includes the following

* Name of the service
* Version of the manager plugin
* Authentication type supported by the manager for API requests. ISC supports the following authentication types:
  + Basic Auth uses username and password
  + Key\_Auth uses API Key
* Whether the manager supports Security Group constructs and is capable of Synchronizing it with ISC
* Whether the manager supports policy mapping.
* Whether Manager is capable of sending Notifications to ISC
* How ISC can subscribe to receive notifications from manager
* Retrieve the manager public key so it can be passed on to new VNF’s deployed by ISC
* Get the URL to the management console of the manager
* Manager Service Function Name
  + Information String. The service type (i.e. NGIPS/NGFW) that will be known to ISC and will show up as part of the service name in NSX.
* Manager NSX Service Function Name (IDS\_IPS, FIREWALL)
  + The service type that will be registered in NSX. See NSX developer guide for more details.

## Manager Notification APIs

### ManagerCallbackNotificationApi, ManagerWebSocketNotificationApi

The notification API’s allow ISC to subscribe to receive notifications from the manager for any changes made by the customer on the manager. For example, if a policy was created/updated or if a new domain was added/modified etc.

* ISC support the following subscription types:
  + Notifications over a Web Socket connection established by ISC
  + Notifications using a Callback URL which ISC can register

The plugin implementation can choose to use either of the mechanisms or choose to not implement any notification.

### IscJobNotificationApi

ISC can also provide notifications to the manager for some actions which the manager wants to track. ISC provides Job and Task notifications to the manager.

For example, if the manager wants to propogate a signature file to some devices and wants to track the status of that action (Job), then ISC would use the IscJobNotificationApi to notify the manager about the progress and success/failure of that job.

## Manager Device API (ManagerDeviceApi)

These API’s allow ISC to manage the Devices on the VNF manager.

The functionality provided by the API include:

* Whether Device grouping is supported
* Managing Device Containers/Device Groups(if applicable)
  + Creating Device Containers
  + Updating Device Containers
  + Retrieving Device Containers
    - List all device containers
    - Retrieve by ID
    - Retrieve by Name
  + Deleting Device Containers
* Managing Members within Device Containers
  + Creating a Device Member within a Device Container(in Agent Based only)
  + Updating Device Member information
  + Retrieving Device Container Members
    - List all Device Container Members
    - Retrieve by Device Member ID
    - Retrieve by Device Member Name
  + Deleting a Device Member
* Check if an appliance supports upgrading from one version to another(Upgrade or downgrade)

### Functionality Applicable for Agentless deployments

As part of deployment of the VNF, OSC provides some bootstrapping information to the appliance so it can communicate with its manager (for agentless) and with OSC for agent based deployments.

This is achieved by using the [configuration drive](http://docs.openstack.org/user-guide/cli_config_drive.html) functionality provided by openstack. The config drive attaches to the instance when it boots. The instance can mount this drive and read files from it to get information that is normally available through the metadata service.

It is intended to provide a minimal amount of launch-time personalization. The max size of the file contents is 10KB.

OSC allows you to specify a list of multiple files each with their own name and content.

For OSC, /<MOUNT\_POINT>/openstack/content/0000 will contain the first file content as a **base64** encoded string which when decoded will have the following information in a key value format.

managerIp=X.X.X.X

managerUser=agent

managerPassword=XXXXXX

virtualSystemId=353

applianceName=IPS-353-257

The above is just a sample. The manger should be able to provide us with the information which it expects to be present in the seed file so it can initialize itself and be able to talk to its manager.

* Provide bootstapping information to the device as an list of files
  + Provide bootstrapping information like
    - Name of the device
    - Manager IP
    - Manager Username
    - Manager password
    - Other Key value pair type properties
  + Information will be available in config drive under /MOUNT\_POINT/openstack/content/0000 as Base64 encoded string

### Functionality which is applicable for Agent based deployments only

* + Bootstrapping is handled by the controller agent present on the device member. No need to provide any information.

## Manager Policy Mapping API (ManagerSecurityGroupInterfaceApi)

* Managing Policy – Tag mapping within the context of a device container. This API does not be to be implemented it the manager does not support policy mapping.
  + Creating Mapping between policy Id and tag(vlan)
  + Updating the Mapping between policy Id and tag
  + Retrieving the mappings
    - List all Mapping
    - Retrieve Mapping by ID
    - Retrieve Mapping by Name
  + Deleting a Mapping

## Manager Security Grouping API (ManagerSecurityGroupApi) (If applicable)

* Synchronizing security groups between ISC and Manager
  + Creating group in manager referencing ISC group id and members within the security group
  + Updating the group name and members within the group
  + Retrieving the groups
    - List all groups
    - Retrieve group by ID
  + Deleting a group

## Manager Policy API (ManagerPolicyApi)

The manager Policy API’s are to allow ISC to synchronize with the policies exposed by the manager

The functionality includes:

* Retrieving Policy information(Name and ID) based on Policy ID and Domain ID
* List all Policies belonging to a specific Domain

## Manager Domain API (ManagerDomainApi)

The manager Domain API’s are to allow ISC to synchronize with the security domains exposed by the manager. This API does not need implementation if the manager does not support policy mapping.

The functionality includes:

* Retrieving Domain information(Name and ID) based on Domain ID
* List all Domains

# REST APIs Exposed by ISC

The following are Rest API’s exposed by ISC to support functionality.

Security

* Basic Auth

Request/Response Content type

* application/xml
* application/json

## Notification API

Notifies ISC about registered changes in Manager

/api/manager/v1/notification

POST

The relevant manager connector is derived from the IP address of the HTTP client the notification request is reported by and responds to the notification accordingly.

### Request

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Located in | Required | Description | Default | Schema |
| body | body | yes |  | - | Notification |

Notification

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Required | Description |
| eventNotification | EventNotification | optional | - |

EventNotification

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Required | Description |
| eventObject | string | required | Notification Object Identifier. This will be interpreted by the Manager Plugin |
| eventType | string | required | Notification Object Type. This will be interpreted by the Manager Plugin |

### Response

|  |  |  |
| --- | --- | --- |
| Status Code | Reason | Response Model |
| 200 | Successful operation | BaseJobResponse |
| 400 | In case of any error | ErrorCodeDto |

## Propogate Manager file API(Agent Based deployment only)

Propagate a Manager File to Appliance Instances

/api/manager/v1/propagateMgrFile/vs/{virtualSystemName}

PUT

Provided virtualSystemName must be of an existing Virtual Security System (VSS).

MgrFile request contains Manager File information and list of Appliance Instances to propagate file to. If Appliance Instances is ommited, all instances is assumed.

If successful, returns the File Propagation Job Id.

Each Appliance Instance file will be propagated and persisted in the CPA directoy and the process-mgr-file.py will be called to notify the appliance to process the file.

### Request

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Located in | Required | Description | Default | Schema |
| virtualSystemName | path | yes | Virtual System name to which file propagation is requested | - | string |
| body | body | yes | The File to propogate | - | [MgrFile](file:///F:\svn\vmidc\01\trunk\server\vmiDCServer\target\webapp\api-doc\api.html#/definitions/MgrFile) |

MgrFile

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Required | Description |
| mgrFile | array[string] | required | A stream of bytes represent the content of the file. |
| mgrFileName | string | required | The filename will be used when file is persisted. |
| applianceInstances | array[string] | optional | list of dai IDs, null or empty list will indicate ALL option |

### Response

|  |  |  |
| --- | --- | --- |
| Status Code | Reason | Response Model |
| 200 | Corresponding File Propagation Job started. Id in response is expected to be empty | [BaseJobResponse](file:///F:\svn\vmidc\01\trunk\server\vmiDCServer\target\webapp\api-doc\api.html#/definitions/BaseJobResponse) |
| 400 | In case of any error | [ErrorCodeDto](file:///F:\svn\vmidc\01\trunk\server\vmiDCServer\target\webapp\api-doc\api.html#/definitions/ErrorCodeDto) |

## Query VM Information API

Allows the manger to Query Virtual Machine information based on VM UUID, IP, MAC or Flow 6-field-tuple.

/api/manager/v1/queryVmInfo

POST

Query VM information based on VM UUID, IP, MAC or Flow 6-field-tuple. Request can include all search criteria.

If found, the respond will include the VM information based on the information provided for query.

For example, if IP is provided, response will include a map entry where the key is the IP and the value is the VM information.

### Request

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Located in | Required | Description | Default | Schema |
| body | body | yes |  | - | [QueryVmInfoRequest](file:///F:\svn\vmidc\01\trunk\server\vmiDCServer\target\webapp\api-doc\api.html#/definitions/QueryVmInfoRequest) |

QueryVmInfoRequest

Description

Contains a list of IP and/or MAC and/or VM-UUID and/or a map of unique-request-identifier key and flow value

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Required | Description |
| applianceInstanceName | string | optional | - |
| ipAddress | array[string] | optional | List of IP Addresses for which VM information is queried |
| macAddress | array[string] | optional | List of MAC Addresses for which VM information is queried |
| vmUuid | array[string] | optional | List of VM UUID for which VM information is queried |
| flow | object | optional | A map of string key and FlowInfo (5-tuple + timestamp) value for which VM information is queried. Key value must be a unique. FlowInfo structure comrise of 6 elements:  sourceIpAddress, sourcePort, destinationIpAddress, destinationPort, protocolId and flowTimestamp |

### Response

|  |  |  |
| --- | --- | --- |
| Status Code | Reason | Response Model |
| 200 | Successful operation | [QueryVmInfoResponse](file:///F:\svn\vmidc\01\trunk\server\vmiDCServer\target\webapp\api-doc\api.html#/definitions/QueryVmInfoResponse) |
| 400 | In case of any error | [ErrorCodeDto](file:///F:\svn\vmidc\01\trunk\server\vmiDCServer\target\webapp\api-doc\api.html#/definitions/ErrorCodeDto) |

QueryVmInfoResponse

Contains a map of key/values. Key can be IP, MAC, VM-UUID or unique-request-identifier in case of Flow based query

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Type | Required | Description | Example | Allowable Values |
| vmInfo | object | optional | A map containing the query identifier key (IP, MAC, VM UUID) and the value holding the VM info |  |  |
| flowVmInfo | object | optional | A map containing a flow based request unique identifier key and the value holding holding the VM info |  |  |

# Packaging

## Manager Plugin

Once Manager Plugin Jar file is created, it is required to package it with metadata to properly import it to ISC. Follow the below steps to package ISC Manager Plugin:

* Create plugin metadata file (file name must be called meta.json)
  + Plugin name. Must match value return by ApplianceManagerApi.getName()
  + Plugin Jar file name
  + ApplianceManagerApi implementer’s class name
  + Minimum ISC version requirement to support this plugin
  + Sample

{

"metaDataVersion": "1.0",

"pluginName": "sample",

"jarFilename": "sampleMgrPlugin.jar",

"applianceManagerApiClass": "com.sample.MyApplianceManagerApi",

"minIscVersion": {

"major": 2,

"minor": 0,

"build": 2370

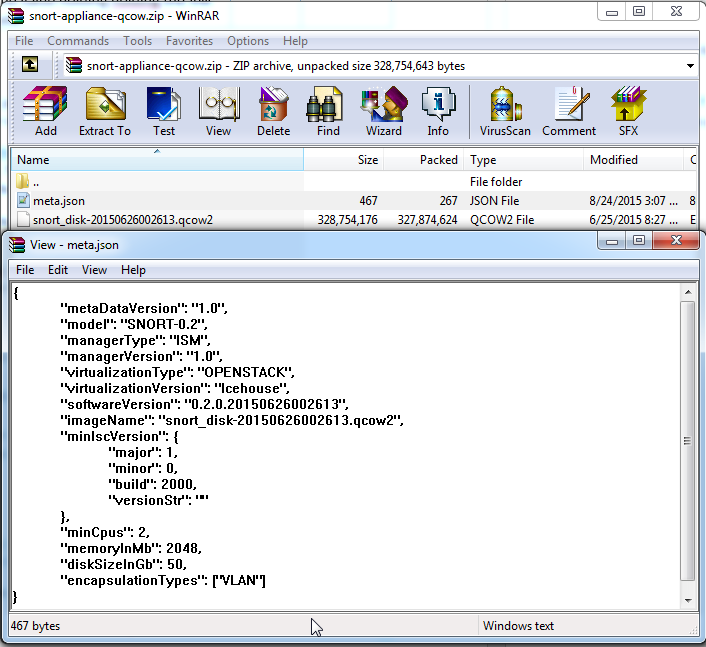
}

}

* Package Manager plugin jar along with the metadata file in zip format.

## Appliance Packaging

OSC expects the appliance to be packaged as a zip along with descriptor file in json format to be able to import this appliance into OSC. An example of this is the following which describes an Appliance for Openstack virtualization platform.



Follow the below steps to package Appliance:

* Create metadata file (file name must be called meta.json)
  + Model – The Model/Name of the device which shows on OSC UI
  + Manager type – This should match plugin name(from the manager plugin meta.json)
  + Manager Version – Version of the manager appliance works with
  + Virtualization Type – Can be OPENSTACK/VMWARE
  + Virtualization Version
  + The Version of the appliance which shows in OSC UI
  + The Name of appliance packaged with the zip file
  + Minimum OSC version requirement to support this plugin
  + CPU/Memory/disk required for the image
  + Encapsulation type – the type of encapsulation to use to provide policy mapping data
  + Additional Nic For Inspection – true to indicate appliance needs separate Ingress and egress inspection ports as opposed to a single inspection port which handles both ingress and egress traffic
  + Image Properties
    - Key-value pairs of properties
    - Used to provide hints in the virtualization environment
    - For Openstack, corresponds to glance image properties
    - For example, hw\_disk\_bus is a property in openstack which allows users to override default disk models configured in nova.
  + Config Properties
    - Key-value pairs of properties
    - Used to provide information to the device during bootstrapping phase
    - For openstack, this information will be available in the config drive
  + Sample

{

"metaDataVersion": "1.1",

"model": "IPS-VM100-VSS",

"managerType": "ISM",

"managerVersion": "1.0",

"virtualizationType": "OPENSTACK",

"virtualizationVersion": "Icehouse",

"softwareVersion": "0.2",

"imageName": "snort\_disk-20150626002613.qcow2",

"minIscVersion": {

"major": 2,

"minor": 5,

"build": 3651,

"versionStr": ""

},

"minCpus": 4,

"memoryInMb": 6144,

"diskSizeInGb": 50,

"encapsulationTypes": ["VLAN"],

"additionalNicForInspection": true,

"imageProperties": {

"hw\_disk\_bus": "ide"

},

"configProperties": {

"applianceModel": "IPS-VM100-VSS",

"applianceSoftwareVersion": "0.2"

}

}

* Package appliance along with zip file.