VMware Smart Assurance Platform API User Guide

VMware Smart Assurance 10.0.0

vmware®

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

If you have comments about this documentation, submit your feedback to

docfeedback@vmware.com

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

Preface

As part of an effort to improve its product lines, VMware periodically releases revisions of its software and hardware. Therefore, some functions described in this document might not be supported by all versions of the software or hardware currently in use. The product release notes provide the most up-to-date information on product features.

Intended Audience

The VMware Smart Assurance DCF and EPS API implements the Application Programmer Interface (API) used to develop applications. It enables Software Development Kit (SDK) users to build systems that apply functionality to unique user requirements. These applications can be adapters that make the states of external devices and systems available to the VMware Smart Assurance system or remote client applications that extend Smart Assurance services to other client applications, such as Network Management systems.

This information is intended for anyone who wants to install and upgrade Smart Assurance products. The information is written for experienced system administrators who are familiar with virtual machine technology and datacenter operations.

This document is intended for software developers implementing subscription and adapter applications that involve the core event correlation, repository, and notification services of a Smart Assurance.

VMware Technical Publications Glossary

VMware Technical Publications provides a glossary of terms that might be unfamiliar to you. For definitions of terms as they are used in VMware technical documentation, go to http://www.vmware.com/support/pubs.

Overview

1

This chapter provides guidelines for building a basic application that uses the Java API to interact with Smart Assurance.

This chapter includes the following topics:

DCF API

The DCF API is the system component that provides access to the facilities of the VMware Smart Assurance product. It allows adapter and application developers to write software that extends and adapts functionality to their specific application requirements.

EPS API

Developers build adapters that forward external events and topology updates to the system. There are also adapters that receive event notifications and correlation results from the core system that integrate with other applications such as network management systems. Developers also use the EPS API to build remote applications, such as user interfaces, that utilize the core information directly. The developer designs and builds systems using these components and the information they exchange (events, topology updates and correlation results).

Using DCF API

2

This section includes instructions for building and running a client using the DCF API with an example.

This chapter includes the following topics:

Create Block

The Create Block program creates a new DCF block based on block object id.

```
/catalog/blocks/{blockId}/provision:
    post:
      tags:
      - "block"
      summary: ""
      description: "Create block: Create a new DCF block based on block object id"
      operationId: "createBlock"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "blockId"
        description: "Block object id"
```

```
        required: true
        type: "string"
      - in: body
        name: payload
        required: true
        schema:
          type: string
      responses:
        default:
          description: "Returns Block Config for the newly created block"
```

Example:

Request : curl --user admin:changeme -k -X POST -H 'Content-Type: application/json' https://127.0.0.1:8443/dcc/v1/catalog/blocks/smarts-metrics/provision

```
Response :
        {
          "blockID": "smarts-metrics",
          "message": "Block successfully deployed",
          "time": "2019-02-25 23:25:06.824015",
          "instanceId": "smarts-metricseaea8c69-b12a-4fa6-8389-6ea87dd997c5",
          "params": {
            "smarts-metrics": {
              "primary": {
                "collect": {
                  "sm": [
                    {
                      "host": "localhost",
                      "port": "12345",
                      "name": "INCHARGE-AM-PM",
                      "username": "admin",
                      "password":
"{FAF745B9C01D858B45D7F644CE55D30064E409A847AA2351B30B6FB74B3BA7FDEC5A9315ECA00C2F69D9C425039CA1F1}",
                      "type": "both",
                      "topologyPollingPeriod": "3600",
                      "metricPollingPeriod": "240",
                      "backendgroup": "group",
                      "thread": {
                        "poolsize": "3"
                      },
                      "dmconnection": {
                        "poolsize": "3"
                      }
                    }
                  ]
                }
              },
              "kafka": {
                "cluster": {
                  "node": [
                    {
                      "host": "localhost",
                      "port": "9092"
```

```
                    }
                  ],
                  "topic": "default-topic",
                  "auth": "true",
                  "username": "admin",
                  "password": "admin-secret"
                }
              }
            }
          }
        }
```

Get Block Objects

Get Block Objects program fetches block object id by capability.

```
/catalog/blocks/{capabilityType}/{metrics}:
    get:
      tags:
      - "block"
      summary: ""
      description: "Get block objects: Get block object id by capability"
      operationId: "getBlockObjects"
      produces:
      - "application/json"
      parameters:
      - name: "capabilityType"
        in: "path"
        description: "The type of data collector capability for the block"
        required: true
        type: "string"
      - name: "metrics"
        in: "path"
        description: "The capability type and metrics for this block"
        required: true
        type: "string"
      responses:
        200:
          description: "Block object id successfully returned"
        400:
          description: "Invalid Request"
        404:
          description: "Block object id not found"
```

Example:

```
Request : curl -k -u admin:changeme --request GET --url
https://127.0.0.1:8443/dcc/v1/catalog/blocks/smarts-metricsAPI/availability

Response : "smarts-metrics"
```

Get Instantiated Blocks

Get Instantiated Blocks program extracts information about instantiated blocks.

```
/catalog/blocks/instances:
    get:
      tags:
      - "block"
      summary: ""
      description: "Get instantiated blocks: Get information about instantiated blocks"
      operationId: "getInstantiatedBlocks"
      produces:
      - "application/json"
      responses:
        200:
          description: "Instantiated blocks returned"
        500:
          description: "Exception: Error in getting running blocks"
```

Example:

```
Request : curl –k –u admin:changeme ––request GET ––url
https://127.0.0.1:8443/dcc/v1/catalog/blocks/instances.
```

```
Response : {"smarts–metrics4cc747e0–da89–433e–9233–a6895b89755b":
{"block_object_id": "smarts–metrics", "deployment_status": "Running",
"dc_service_status": "STOPPED", "poll_count": 16}, "smarts–metrics035f8121–
aaed–4d66–83eb–ae3a7841f720": {"block_object_id": "smarts–metrics",
"deployment_status": "Running", "dc_service_status": "STOPPED", "poll_count": 16}}
```

Get Packages

Get Package program fetches information about packages.

```
/catalog/blocks/packages:
    get:
      tags:
        – "block"
      summary: ""
      description: "Get packages: Get information about packages"
      operationId: "getPackages"
      produces:
        – "application/json"
      responses:
        200:
          description: "List of packages"
        500:
          description: "Exception: Error in getting packages"
```

Example:

Request : curl –k ––user admin:changeme ––request GET ––url https://127.0.0.1:8443/dcc/v1/catalog/blocks/packages

Response : ["clearwater", "cloudify–orchestrator", "smarts–metrics", "smarts–notifs", "velocloud–sdwan", "vrops"]

Get Services

Get Services program extracts information about running services within the block.

```
/catalog/blocks/{instanceId}/services:
    get:
      tags:
      – "block"
      description: "Get services: Get information about running services within the block"
      operationId: "getServices"
      produces:
      – "application/json"
      parameters:
      – name: "instanceId"
        in: "path"
        description: "The instance id of the running block for which we want the services"
        required: true
        type: "string"
      responses:
        200:
          description: "All services returned"
        400:
          description: "Invalid Request"
        500:
          description: "Exception: In getting service details of block"
```

Example:

Request : curl –k ––user admin:changeme ––request GET ––url https://127.0.0.1:8443/dcc/v1/catalog/blocks/smarts–metrics70d9afac–516a–47f5–b38f–799fa530d55c/services

Response : {"id": "apg–collector–manager–smarts–metricseaea8c69–b12a–4fa6–8389–6ea87dd997c5", "module": "collector–manager", "instance": "smarts–metricseaea8c69–b12a–4fa6–8389–6ea87dd997c5", "status": "STOPPED", "last–modified": "–1", "startup–mode": "auto", "start–order": "96", "stop–order": "4"}

Start Service

Start Service program starts the block service.

```
/runtime/blocks/{instanceId}/service/start:
    post:
      tags:
      – "runtime"
```

```
      description: "Start service: Start the block service"
      operationId: "startService"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "instanceId"
        description: "UUID of the instantiated DCF block"
        required: true
        type: "string"
      responses:
        200:
          description: "Success"
        400:
          description: "Invalid Request"
        500:
          description: "Exception: In executing service request START on block"
```

Example:

Request : `curl –k ––user admin:changeme ––request POST ––url https://127.0.0.1:8443/dcc/v1/runtime/blocks/smarts–metrics70d9afac–516a–47f5–b38f–799fa530d55c/service/start`

`Response : [{"id": "apg–collector–manager–smarts–metricseaea8c69–b12a–4fa6–8389–6ea87dd997c5", "module": "collector–manager", "instance": "smarts–metricseaea8c69–b12a–4fa6–8389–6ea87dd997c5", "status": "STARTED", "last–modified": "1551155145000", "startup–mode": "auto", "start–order": "96", "stop–order": "4"}]`

Stop Service

Stop Service program stops the running block service.

```
/runtime/blocks/{instanceId}/service/stop:
    post:
      tags:
      - "runtime"
      description: "Stop service: stop the block service."
      operationId: "stopService"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "instanceId"
        description: "UUID of the instantiated DCF block"
        required: true
        type: "string"
      responses:
        200:
          description: "Success"
```

```
        400:
          description: "Invalid Request"
        500:
          description: "Exception: In executing service request STOP on block"
```

Example:

```
Request : curl –k ––user admin:changeme ––request POST ––url
https://127.0.0.1:8443/dcc/v1/runtime/blocks/smarts–metrics70d9afac–516a–47f5–
b38f–799fa530d55c/service/stop
```

```
Response : [{"id": "apg–collector–manager–smarts–metricseaea8c69–
b12a–4fa6–8389–6ea87dd997c5", "module": "collector–manager", "instance": "smarts–
metricseaea8c69–b12a–4fa6–8389–6ea87dd997c5", "status": "STOPPED", "last–modified":
"1551155147000", "startup–mode": "auto", "start–order": "96", "stop–order": "4"}]
```

Restart Service

Restart Service program restarts a specific block service.

```
/runtime/blocks/{instanceId}/service/restart:
    post:
      tags:
      – "runtime"
      description: "Restart service: Restart a specific block service"
      operationId: "restartService"
      produces:
      – "application/json"
      parameters:
      – in: "path"
        name: "instanceId"
        description: "UUID of the instantiated DCF block"
        required: true
        type: "string"
      responses:
        200:
          description: "Success"
        400:
          description: "Invalid Request"
        500:
          description: "Exception: In executing service request RESTART on block"
```

Example:

```
Request : curl –k ––user admin:changeme ––request POST ––url
https://127.0.0.1:8443/dcc/v1/runtime/blocks/smarts–metrics70d9afac–516a–47f5–
b38f–799fa530d55c/service/restart
```

Response : [{"id": "apg-collector-manager-cloudify-orchestratorffa3bb75-99f4-4613-8a5a-45e2cee0372c", "module": "collector-manager", "instance": "cloudify-orchestratorffa3bb75-99f4-4613-8a5a-45e2cee0372c", "status": "STARTED", "last-modified": "1551155175000", "startup-mode": "auto", "start-order": "96", "stop-order": "4"}]

Get Default Block Config

Get Default Block Config program extracts the configurations if no block is installed.

```
/catalog/blocks/{blockId}/config/default:
    get:
      tags:
      - "block"
      description: "Get default block config: Get configurations if no block is installed"
      operationId: "getDefaultBlockConfig"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "blockId"
        description: "Block ID"
        required: true
        type: "string"
      responses:
        200:
          description: "Returns default Block Config"
        400:
          description: "Invalid Request"
        500:
          description: "Exception: In getting module answers from block"
```

Example:

Request : curl -k -u admin:changeme --request GET --url https://127.0.0.1:8443/dcc/v1/catalog/blocks/smarts-metrics/config/default

```
Response :
        {
          "smarts-metrics": {
            "primary": {
              "collect": {
                "sm": [
                  {
                    "host": "localhost",
                    "port": "12345",
                    "name": "INCHARGE-AM-PM",
                    "username": "admin",
                    "password":
"{CCC8B6EFC8D913616264F4B8A0D25B5A52DBFAFE4F6EB5BB94B86D4B25BC2FA7F11E59A9A733BBA10002A20C5D60EE6A}",
                    "type": "both",
                    "topologyPollingPeriod": "3600",
```

```
                    "metricPollingPeriod": "240",
                    "backendgroup": "group",
                    "thread": {
                      "poolsize": "3"
                    },
                    "dmconnection": {
                      "poolsize": "3"
                    }
                  }
                ]
              }
            },
            "kafka": {
              "cluster": {
                "node": [
                  {
                    "host": "localhost",
                    "port": "9092"
                  }
                ],
                "topic": "default-topic",
                "authType": "SASL_PLAINTEXT",
                "username": "admin",
                "password":
"{B07EA85D8D516D4CA4201CFC8DEBEE022E6DE44C0AE6CB9652FC850F08859AFF5A93A33A85220F3E9E1719D3595D6AA7}"
              }
            }
          }
        }
```

Get Block Config

Get Block Config program fetches the configurations for an installed module and instance.

```
/runtime/blocks/{instanceId}/config/get:
    get:
      tags:
      - "runtime"
      description: "Get block config: Get configurations for an installed module and instance"
      operationId: "getBlockConfig"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "instanceId"
        description: "UUID of the instantiated DCF block"
        required: true
        type: "string"
      responses:
        200:
          description: "Returns Block Config"
```

```
        400:
          description: "Invalid Request"
        500:
          description: "Exception: In getting module answers from block"
```

Example:

Request : curl –k ––user admin:changeme ––request GET ––url https://127.0.0.1:8443/dcc/v1/runtime/blocks/smarts–metrics70d9afac–516a–47f5–b38f–799fa530d55c/config/get

```
Response :
        {
          "smarts–metrics": {
            "primary": {
              "collect": {
                "sm": [
                  {
                    "host": "localhost",
                    "port": "12345",
                    "name": "INCHARGE–AM–PM",
                    "username": "admin",
                    "password":
"{FAF745B9C01D858B45D7F644CE55D30064E409A847AA2351B30B6FB74B3BA7FDEC5A9315ECA00C2F69D9C425039CA1F1}",
                    "type": "both",
                    "topologyPollingPeriod": "3600",
                    "metricPollingPeriod": "240",
                    "backendgroup": "group",
                    "thread": {
                      "poolsize": "3"
                    },
                    "dmconnection": {
                      "poolsize": "3"
                    }
                  }
                ]
              }
            },
            "kafka": {
              "cluster": {
                "node": [
                  {
                    "host": "localhost",
                    "port": "9092"
                  }
                ],
                "topic": "default–topic",
                "auth": "true",
                "username": "admin",
                "password": "admin–secret"
              }
            }
          }
        }
```

Configure Block

Configure Block program updates the configurations for an installed module and instance.

```
/runtime/blocks/{instanceId}/config/set:
    post:
      tags:
      - "runtime"
      description: "Configure block: Update configurations for an installed module and instance"
      operationId: "configureBlock"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "instanceId"
        description: "UUID of the instantiated DCF block"
        required: true
        type: "string"
      - in: body
        name: payload
        required: true
        schema:
          type: string
      responses:
        200:
          description: "Success"
        400:
          description: "Invalid Request"
        500:
          description: "Exception: In updating module answers for the block"
```

Example:

```
Request : curl -k --user admin:changeme --request POST --url
https://127.0.0.1:8443/dcc/v1/runtime/blocks/smarts-metrics70d9afac-516a-47f5-
b38f-799fa530d55c/config/set --header 'content-type: application/json' --data '{
      "smarts-metrics": {
        "primary": {
          "collect": {
            "sm": [
              {
                "host": "localhost",
                "port": "12345",
                "name": "INCHARGE-AM-PM",
                "username": "admin",
                "password":
"{544756F82CDB79957BC6004EA85A6F97CA541AE9AB6CDF4F8D1A99BD2D341B6AF8A75F35A5DE5C59651AFF574DBC604E}",
                "type": "both",
                "topologyPollingPeriod": "3600",
                "metricPollingPeriod": "240",
                "backendgroup": "group",
                "thread": {
```

```
                  "poolsize": "3"
                },
                "dmconnection": {
                  "poolsize": "3"
                }
              }
            ]
          }
        },
        "kafka": {
          "cluster": {
            "node": [
              {
                "host": "localhost11",
                "port": "9092"
              }
            ],
            "topic": "default-topic",
            "authType": "SASL_PLAINTEXT",
            "username": "admin",
            "password":
"{28811D69F2FEEA51A491E8DF763912A5B692517875494C8B524D712CF3576658B1E0ED1CF87F00B0F88E4EFAA487A5D6}"
          }
        }
      }
    }'
```

```
Response : New configuration successfully set in block: smarts-metricseaea8c69-
b12a-4fa6-8389-6ea87dd997c5 for smarts-metrics service
```

Delete Block

Delete Block program deletes a running DCF block.

```
/catalog/blocks/{instanceId}:
    delete:
      tags:
      - "block"
      description: "Delete block: delete a running DCF block"
      operationId: "deleteBlock"
      produces:
      - "application/json"
      parameters:
      - in: "path"
        name: "instanceId"
        description: "UUID of the instantiated DCF block"
        required: true
        type: "string"
      responses:
        200:
          description: "Success"
```

```
        400:
          description: "Invalid Request"
        500:
          description: "Exception in deleting the block"
```

Example:

```
Request : curl –k –u admin:changeme ––request DELETE ––url
https://127.0.0.1:8443/dcc/v1/catalog/blocks/smarts–metrics035f8121–aaed–4d66–83eb–
ae3a7841f720
```

```
Response : {"instanceId": "smarts–metrics035f8121–aaed–4d66–83eb–ae3a7841f720",
"message": "Block instance successfully deleted", "time": "2019–04–09
05:42:21.204886"}
```

Using EPS API

3

This chapter includes the following topics:

RightClick Operation

Acknowledge Request

The Acknowledge Request operation acknowledges the notification.

Request Example:

```
{
"AuditTrailText": "User action done",
"Names": [
"NOTIFICATION-Interface_IF-10.107.116.241/200_Down",
" NOTIFICATION-Interface_IF-10.107.116.243/6_Down"
],
"User": "admin"
}
```

Response Example:

```
{
"Failed": [
"NOTIFICATION-Interface_IF-10.107.116.241/200_Down"
]
}
```

Output: Response Code is 200 OK.

ReleaseOwnership Request

Release Ownership Request operation releases the ownership of the Notification.

Request Example:

```
{
"AuditTrailText": "User action done",
"Names": [
"NOTIFICATION-Interface_IF-10.107.116.241/200_Down",
" NOTIFICATION-Interface_IF-10.107.116.243/6_Down"
],
"User": "admin"
}
```

Response Example:

```
{
"Failed": [
"NOTIFICATION-Interface_IF-10.107.116.241/200_Down"
]
}
```

Output: Response Code is 200 OK.

TakeOwnership Request

TakeOwnership Request operation describes about the taking ownership of the notification.

Request Example:

```
{
"AuditTrailText": "User action done",
"Names": [
"NOTIFICATION-Interface_IF-10.107.116.241/200_Down",
" NOTIFICATION-Interface_IF-10.107.116.243/6_Down"
],
"User": "admin"
}
```

Response Example:

```
{
"Failed": [
"NOTIFICATION-Interface_IF-10.107.116.241/200_Down"
]
}
```

Output: Response Code is 200 OK.

UnAcknowledge Request

UnAcknowledge Request operation unacknowledges the notification.

Request Example:

```
{
"AuditTrailText": "User action done",
"Names": [
"NOTIFICATION-Interface_IF-10.107.116.241/200_Down",
" NOTIFICATION-Interface_IF-10.107.116.243/6_Down"
],
"User": "admin"
}
```

Response Example:

```
{
"Failed": [
"NOTIFICATION-Interface_IF-10.107.116.241/200_Down"
]
}
```

Output: Response Code is 200 OK.

Login and Logout Operation

This section describes how to perform login and logout operations using APIs.

Login

This section describes Login operation of EPS with the help of APIs.

Using GET API to Login (GET /login/{username} Login):

This API reads UserProfile criteria from SAM and store it in Cache.

Request Example:

```
string
(path)
```

Response Example:

```
{
"bool": {}
}
```

Output: Response code is 200 OK.

Using GET API for loginUserDetails (GET /loginUserDetails):

This API returns UserProfile criteria store it in Cache.

Request Example:

```
string
(header)
```

Response Example:

```
{
"bool": {}
}
```

Output: Response code is 200 OK.

Logout

This API deletes the UserProfile stored in Redis Cache.

Request Example:

```
string
(path)
```

Response Example:

```
{
"Result": "Logout Successful"
}
```

Output: Response code is 200 OK.

Configuration

The API "POST /refresh", refreshes the configuration of Eventstore Service.

Request Example:

```
{
"doc": {
"kafkaTopic": "sam_notification",
"logLevel": "TRACE"
}
}
```

Response Example:

```
{
"Result": "OK"
}
```

Output: Response code is 200 OK.

Server Tools

This API provides the information about the available Server Tools of the notification.

Request Example:

```
{
"name": "NOTIFICATION-Interface_IF-10.107.116.241/200_Down"
}
```

Response Example:

```
[
{
"DisplayName": "Sample – Open Trouble Ticket",
"Name": "ICS-ServerProgramAction-Sample – Open Trouble Ticket"
}
]
```

Output: Response code is 200OK.

ServerTools Action Request

This API describes about Server Tool Action for the notification.

Request Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<common.Arguments>
<actionID>ICS-ServerProgramAction-Sample – Open Trouble Ticket</actionID>
<parameters>
<actionParam>
<paramName>TicketID</paramName>
<paramValue>541</paramValue>
</actionParam>
</parameters>
<target>ICS_Notification::NOTIFICATION-Router_10.107.117.20_Unresponsive</target>
</common.Arguments>
```

Response Example:


```
{
"ActionReturn": [
"0",
" ****************************************\nPinging from Source Domain = INCHARGE-AMPM\
nDevice = ICIM_UnitaryComputerSystem \"2C-36-F8-B5-A2-21\"\nIPAddr = MR-E-NON_SINGLETON_SETSet\
n****************************************\nPing to MR-E-NON_SINGLETON_SET-Set
succeeded\nPing request could not find host MR-E-NON_SINGLETON_SET-Set. Please check the name and
try
again.\n****************************************\n****************************************\n
\n"
]
}
```


Output: Response code is 200 OK.

Models

common.ActionReturnRes


```
{
ActionReturn [
example: List [ "0", " ****************************************\nPinging from Source Domain =
INCHARGE-AM-PM\nDevice = ICIM_UnitaryComputerSystem \"2C-36-F8-B5-A2-21\"\nIPAddr = MR-ENON_
SINGLETON_SET-Set\n****************************************\nPing to MR-E-NON_SINGLETON_SET-Set
succeeded\nPing request could not find host MR-E-NON_SINGLETON_SET-Set. Please check the name and try
again.\n****************************************\n****************************************\n\n"
]string]
}
```


common.Arguments


```
{
actionID string
example: ICS-ServerProgramAction-Sample – Open Trouble Ticket
parameters {
actionParam {
paramName string
example: TicketID
paramValue string
example: 541
}
}
target string
example: ICS_Notification::NOTIFICATION-Router_10.107.117.20_Unresponsive
}
```

common.ConfigRequest

```
{
doc {
kafkaTopic string
example: sam_notification
logLevel string
example: TRACE
}
```

common.Document

```
{
kafkaTopic string
example: sam_notification
logLevel string
example: TRACE
}
```

common.FailedMessage

```
{
Failed [
example: List [ "NOTIFICATION-Interface_IF-10.107.116.241/200_Down" ]string]
}
```

common.LoginSuccess

```
{
bool {
}
}
```

common.LogoutSuccess

```
{
Result string
example: Logout Successful
}
```

common.RefreshSuccess

```
{
Result string
example: OK
}
```

common.Request

```
{
AuditTrailText string
example: User action done
Names [
example: List [ "NOTIFICATION-Interface_IF-10.107.116.241/200_Down", " NOTIFICATIONInterface_
IF-10.107.116.243/6_Down" ]string]
User string
example: admin
}
```

common.ServerToolRequest

```
{
name string
example: NOTIFICATION-Interface_IF-10.107.116.241/200_Down
}
```

common.ServerTools

```
{
DisplayName string
example: Sample – Open Trouble Ticket
Name string
example: ICS-ServerProgramAction-Sample – Open Trouble Ticket
}
```